International Technical Support Organization

Lotus Solutions for the Enterprise, Volume 4
Lotus Notes and the MQSeries Enterprise Integrator

October 1997
Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, “Special Notices” on page 319.

First Edition (October 1997)

This edition applies to MQSeries Enterprise Integrator for Lotus Notes Release 1.0.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE  Building 80-E2
650 Harry Road
San Jose, California 95120-6099

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

© Copyright International Business Machines Corporation 1997. All rights reserved.
Note to U.S. Government Users —Documentation related to restricted rights —Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figures</td>
<td>vii</td>
</tr>
<tr>
<td>Tables</td>
<td>ix</td>
</tr>
<tr>
<td>Preface</td>
<td>xi</td>
</tr>
<tr>
<td>How This Redbook is Organized</td>
<td>xii</td>
</tr>
<tr>
<td>The Team That Wrote This Redbook</td>
<td>xii</td>
</tr>
<tr>
<td>Lotus Solutions for the Enterprise Collection</td>
<td>xiv</td>
</tr>
<tr>
<td>Comments Welcome</td>
<td>xv</td>
</tr>
<tr>
<td>Chapter 1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Business Environment</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Existing Solutions</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Today’s Requirements</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Lotus Notes Solutions</td>
<td>4</td>
</tr>
<tr>
<td>1.5 MQEI</td>
<td>4</td>
</tr>
<tr>
<td>Chapter 2. System Components</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Lotus Notes</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 Notes Databases</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 Documents</td>
<td>7</td>
</tr>
<tr>
<td>2.1.3 Forms</td>
<td>8</td>
</tr>
<tr>
<td>2.1.4 Views and Folders</td>
<td>8</td>
</tr>
<tr>
<td>2.1.5 Navigators</td>
<td>8</td>
</tr>
<tr>
<td>2.1.6 Agents</td>
<td>8</td>
</tr>
<tr>
<td>2.1.7 Application Development</td>
<td>8</td>
</tr>
<tr>
<td>2.1.8 Lotus Notes Domino Server</td>
<td>9</td>
</tr>
<tr>
<td>2.1.9 Domino.Connect</td>
<td>11</td>
</tr>
<tr>
<td>2.2 MQEI</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 System Structure</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2 Application Access</td>
<td>14</td>
</tr>
<tr>
<td>2.2.3 Security</td>
<td>15</td>
</tr>
<tr>
<td>2.3 MQSeries</td>
<td>15</td>
</tr>
<tr>
<td>2.3.1 Messages</td>
<td>18</td>
</tr>
<tr>
<td>2.3.2 Queue Managers</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3 Message Queues</td>
<td>18</td>
</tr>
<tr>
<td>2.3.4 Channels</td>
<td>19</td>
</tr>
<tr>
<td>2.3.5 Triggering</td>
<td>19</td>
</tr>
<tr>
<td>2.3.6 Distributed MQSeries</td>
<td>21</td>
</tr>
<tr>
<td>2.3.7 MQSeries Bridges</td>
<td>23</td>
</tr>
<tr>
<td>2.4 CICS</td>
<td>24</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1997
2.4.1 Communication Area ................................ 25
2.4.2 Distributed Program Link .......................... 25
2.4.3 Basic Mapping Support ............................. 25
2.4.4 CICS Clients ..................................... 26
2.4.5 MQSeries-CICS/ESA DPL Bridge .................... 27
2.5 IMS .............................................. 29
2.5.1 Input and Output Messages ......................... 29
2.5.2 Open Transaction Manager Access .................... 30
2.5.3 MQSeries-IMS Bridge ............................. 30
2.6 Components of the ITSO MQEI Sample Application .... 32

Chapter 3. Description of MQEI ............................ 37
3.1 LotusScript Extension .................................. 37
  3.1.1 EISession Object ................................ 38
  3.1.2 EIService Object ................................ 39
  3.1.3 EIMessage Object ............................... 40
  3.1.4 EISendOptions Object ........................... 40
  3.1.5 EIReceiveOptions Object ......................... 40
3.2 Definition Database .................................... 40
3.3 Security Database .................................... 41
3.4 MQEI Utilities ....................................... 42
  3.4.1 CICS BMS Map Conversion Utility ................. 42
  3.4.2 MQEI Trace Utility .............................. 42
  3.4.3 MQEI Code Level Utility ......................... 42

Chapter 4. Design Issues ................................. 43
4.1 Application Models .................................... 43
  4.1.1 Two Tier Model ................................ 43
  4.1.2 Three Tier Model ............................... 45
  4.1.3 User Types ................................... 46
  4.1.4 Design Decision Table .......................... 51
4.2 Security ........................................... 52
  4.2.1 Internet ....................................... 53
  4.2.2 Lotus Notes .................................... 54
  4.2.3 Middleware ..................................... 54
  4.2.4 Enterprise Applications ......................... 54
  4.2.5 How MQEI Handles Enterprise Application Security . 54
  4.2.6 Security Considerations for Lotus Notes Clients . 56
4.3 Replication ......................................... 56
4.4 Recovery and Restart ................................ 58
4.5 MQEI and Other Bridge Solutions ..................... 60
  4.5.1 Link and Link Extra Technologies ............... 60
  4.5.2 MQSeries Link LotusScript Extension ............ 61
  4.5.3 Data Access and Copy .......................... 63
Figures

1. MQEI System Structure .................................. 13
2. Message-Driven Processing ............................... 16
3. MQSeries Triggering ...................................... 21
4. MQSeries Client .......................................... 22
5. MQSeries Distributed Queuing ............................ 23
6. MQSeries-CICS/ESA DPL Bridge .......................... 28
7. Application Flow with the MQSeries-IMS Bridge ......... 31
8. Components of the ITSO MQEI Sample Application .......... 33
9. Test Environment ......................................... 35
10. MQEI Objects ........................................... 38
11. Definition Database Interface ............................ 41
12. Security Database Interface .............................. 42
13. Two Tier Model ......................................... 44
14. Three Tier Model ........................................ 45
15. Single Connection ....................................... 57
16. Multiple Connections ................................... 58
17. MQEI Security Definition for a Lotus Notes User ......... 67
18. MQEI Security Definition for the Lotus Notes Server .... 67
19. Sample Application ITSO Form ........................... 69
20. Lotus Notes Client Access (Two Tier Model) .............. 72
21. Scheduled Agent Definition ............................... 79
22. Web User Access (Three Tier Model) ..................... 81
23. MQEI CICS DPL Direct Service Process Flow ............ 89
24. CICS DPL Direct Service Definition ..................... 90
25. CICS DPL Direct Service Message Definition ............ 91
26. LotusScript Steps with CICS DPL Direct Service ......... 93
27. CICS 3270 Direct Service Process Flow .................. 95
28. CICS 3270 Direct Service Definition ..................... 96
29. CICS 3270 Direct Service Messages Definition .......... 97
30. LotusScript Steps with CICS 3270 Direct Service ......... 100
31. Native MQSeries Service Process Flow .................. 103
32. Native MQSeries Service Definition ..................... 105
33. Enhanced Form for Delayed Replies ..................... 107
34. CICS DPL via MQSeries Service Process Flow ........... 111
35. CICS DPL via MQSeries Service Definition ............. 113
36. IMS via MQSeries Service Process Flow .................. 117
37. IMS via MQSeries Service Definition .................... 120
38. IMS via MQSeries Service Message Definition .......... 121
39. Lotus Notes Client Agent for CICS DPL Direct Service .... 127
40. Lotus Notes Client Agent for CICS 3270 Direct Service .... 136
41. Lotus Notes Client Agent for CICS DPL via MQSeries Service 151

© Copyright IBM Corp. 1997
42. Lotus Notes Client Agent for Native MQSeries Service .......... 162
43. Lotus Notes Client Agent for IMS via MQSeries Service .......... 173
44. Scheduled Lotus Notes Agent for Native MQSeries Service .......... 184
45. Web Client Agent for CICS DPL Direct Service .................. 197
46. Web Client Agent for CICS 3270 Direct Service .................. 207
47. Web Client Agent for Native MQSeries Service ................. 222
48. Web Client Agent for CICS DPL via MQSeries Service ............ 233
49. Web Client Agent for IMS via MQSeries Service ................. 244
50. VSAMSERV Source ........................................ 255
51. AMQSCIH1 Header File ...................................... 264
52. AMQSCIC1 CICS MQSeries Program .......................... 265
53. TECHPROG Source ........................................... 282
54. TECHMS BMS Source ........................................ 296
55. PCT Entry for MQ1 Transaction .................................. 305
56. PCT Entry for TECH Transaction .................................. 306
57. PCT Entry for CKTI Transaction .................................. 306
58. PPT Entry for AMQLTMC0 Program ................................ 307
59. PPT Entry for AMQSCIC1 Program ................................ 307
60. PPT Entry for TECHPROG Program ................................ 308
61. PPT Entry for VSAMSERV Program ................................ 308
62. FCT Entry for TECHBASE File ................................... 309
63. FCT Entry for TECHALT Alternate Index File .................... 309
64. CICS/ESA Resource Definitions for the MQSeries - CICS DPL Bridge ........................................ 310
65. Server Connection Channel Definition ............................. 313
66. Initiation Queue Definition for CICS .............................. 313
67. Process Definition to Start CICS TECH Transaction ................ 313
68. Local Queue Definition for ELINPUTQUEUE Input Queue ........ 314
69. Local Queue Definition for ELOUTPUTQUEUE Output Queue .... 314
70. Remote Output Queue Definitions for Queue Located on MVS ...... 314
71. IMS OTMA Queue Located on MVS ................................ 315
72. Transmission Queue Definition to QMGR CSQ1 on MVS ............ 315
73. Sender Channel Definition on OS/2 to MVS ........................ 315
74. Receiver Channel Definition on OS/2 from QMGR MQ01 on MVS .... 315
75. Remote Reply Queue Definitions for Queue Located on OS/2 ...... 316
76. Transmission Queue Definition to QMGR EITEST on OS/2 ........ 316
77. Sender Channel Definition on MVS to OS/2 ........................ 316
78. Receiver Channel Definition on MVS from QMGR EITEST on OS/2 .. 316
79. MQSeries DPL Bridge Request Queue Definition ................... 317
80. IMS OTMA Queue Definition ..................................... 317
### Tables

1. Three Tier and Two Tier Decision Criteria .......................... 51
2. Security Implementation for Web and Lotus Notes Clients .......... 52
3. Link, Link Extra, and MQEI Technologies ............................. 60
4. MQLSX and MQEI Technologies ........................................... 63
Most businesses today depend for their survival on the smooth operation of their core business applications but are being challenged by competitive pressures to extend and adapt these applications to meet new ways of working as they restructure their business processes. Radical modification of the existing applications is often impractical, yet the risks and costs associated with moving to a new technology base for such applications can be very high.

Now there is a better way. Domino provides a variety of integration techniques that allow application developers to meet the challenge of new requirements by leveraging the power of Lotus Domino while at the same time capitalizing on existing software investments. The Domino features fall into two categories:

1. Data-oriented tools that can replicate data between Lotus Notes and relational databases or provide database query and update facilities
2. Tools to link or integrate Lotus Notes applications with transactional or other applications that run on non-Notes systems

This redbook describes the IBM MQSeries Enterprise Integrator for Lotus Notes (MQEI). The MQEI makes the task of developing and managing an extended enterprise application more straightforward. It enhances the Lotus application development language with special extensions and databases that enable you to integrate Lotus Notes applications with existing MQSeries, IMS, and CICS applications.

This redbook is the result of a practical implementation of an extended enterprise application. We examine the elements required to integrate Domino and Lotus Notes applications with our existing enterprise applications by using MQEI. We discuss design issues that we considered during the implementation. We describe in detail how we implemented the application on the intranet, using a Lotus Notes client, and on the Internet, using a Web browser. We also compare the MQEI with other ways of connecting Lotus Notes to enterprise applications.

This book is written to help you understand the various aspects of the MQEI. The book covers the subject from the conceptual view to an actual Lotus Notes sample application developed using the MQEI.

We expect there to be three different audiences for this book: information system architects, Lotus Notes developers, and enterprise developers who want to connect Lotus Notes to their enterprise back-end systems.
How This Redbook is Organized

This redbook is organized as follows:

- Chapter 1, “Introduction” on page 1
  In this chapter we briefly discuss the business environment, the existing solutions, and today’s requirements. We introduce how Lotus Notes and transaction systems, integrated using the MQEI, can be used to meet today’s requirements.

- Chapter 2, “System Components” on page 7
  The components described are Lotus Notes, MQSeries, CICS, IMS, and the MQEI itself. It is intended for architects and developers with a background in either enterprise systems or Lotus Notes, who need to acquire a basic knowledge of the other components in order to understand the discussion of the MQEI.

- Chapter 3, “Description of MQEI” on page 37
  In this chapter we describe the MQEI classes, definition database, security database, and utilities. Developers are provided with an overview of how to use the MQEI.

- Chapter 4, “Design Issues” on page 43
  In this chapter we discuss design issues to consider when using the MQEI to integrate Lotus Notes and enterprise applications. We talk about the design considerations for mobile, Internet, and Lotus Notes client users. We also contrast the MQEI with the other enterprise integration features. Finally, we discuss security, replication, and recovery and restart. This chapter is intended primarily for architects who design applications that integrate Lotus Notes and the enterprise.

- Chapter 5, “ITSO MQEI Sample Application Implementation” on page 65
  In this chapter we describe in detail the ITSO MQEI sample application that we used to test the various connection capabilities of the MQEI.

The Team That Wrote This Redbook

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

Christophe Toulemonde is a Senior ITSO Specialist for client/server and network computing at the Application Development and Data Management ITSO, San Jose Center. He writes extensively and teaches IBM classes worldwide on all areas of client/server and network computing. Before

Lotus Notes and the MQSeries Enterprise Integrator
joining the ITSO, Christophe worked as Technical Manager in an IBM subsidiary, Datablue, in France. You can reach him by e-mail at toulemon@us.ibm.com.

Raymond Quon is a Programmer and Analyst in the Lotus Notes Development Division of Aardvark Consulting in Canada. He has eight years of experience in the mainframe field and three years of experience in the Lotus Notes field. Raymond holds a degree in Computer Information Systems from the British Columbia Institute of Technology. His areas of expertise include Lotus Notes and CICS. You can reach him by e-mail at Ray.Quon@Aardvark.com.

David Reed is an Advisory Programmer in IBM Global Services, United States. He has 18 years of experience in the computer programming field. David has worked at IBM for 15 years. His areas of expertise include application design and programming in a CICS and DB2 environment. You can reach him by e-mail at reedcd@vnet.ibm.com.

Rüdiger Severin is a Lotus Notes Application Developer and Consultant in IBM Global Services, Germany. He holds a degree in Computer Science. Rüdiger has six years of experience in application design and programming in various system environments. You can reach him by e-mail at severin@de.ibm.com.

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

Ed Fletcher
IBM Hursley

Pierangelo Giusta
IBM Italy

Justine Grose
IBM Lotus EMEA Marketing Center

Steve Hanson
IBM Hursley

Suzie Wendler
Advanced Technical Support - North America

Gordon Wilson
ETM Team, IBM EMEA Transaction Systems
Lotus Solutions for the Enterprise Collection

This book is the fourth volume of the Lotus Solutions for the Enterprise Collection. This collection of four books covers the integration of Lotus Notes applications with enterprise data and applications. It contains:

- **Volume 1 - Lotus Notes: An Enterprise Application Platform, SG24-4837**
  This book explains the integration of Lotus Notes applications with traditional enterprise applications at the architecture level. It is designed to help managers and technical professionals evaluate the impact of implementing such an interface between Lotus Notes and transaction and database management systems.

- **Volume 2 - Lotus Notes and Database 2 Common Server, SG24-4918 (to be published in 1997)**
  This book explains how to integrate Lotus Notes applications with data managed by the IBM DATABASE 2 (DB2) family and other database management systems.
  It also describes how to set up communications between Lotus Notes and DB2 common server and provides some practical examples to demonstrate data access.

- **Volume 3 - Using the IBM CICS Gateway for Lotus Notes, SG24-4512**
  This book describes the IBM CICS gateway for Lotus Notes, which enables a Lotus Notes application to access resources on any CICS server.
  Although the book focuses on the functions of the IBM CICS gateway for Lotus Notes, it also describes how to set up communications between Lotus Notes and CICS/ESA and provides some practical examples to demonstrate online CICS data access.

- **Volume 4 - Lotus Notes and the MQSeries Enterprise Integrator, SG24-2217**
  This book describes the use of the MQSeries Enterprise Integrator to connect Lotus Notes to any transaction system. It explains the concept of the extended transaction model supported by this product.
The book focuses on the functions of the MQEI. It also describes how to set up communications between Lotus Notes and several back-end transaction servers.

Comments Welcome

Your comments are important to us!

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

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 341 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:
  - For Internet users: http://www.redbooks.ibm.com
  - For IBM Intranet users: http://w3.itso.ibm.com
- Send us a note at the following address:
  redbook@vnet.ibm.com
Chapter 1. Introduction

In this chapter we briefly discuss the business environment, existing solutions, and today's requirements and explain how Lotus Notes and transaction systems integrated through the IBM MQSeries Enterprise Integrator for Lotus Notes (MQEI) can be used to meet today's requirements.

1.1 Business Environment

Every business today is investing in sophisticated and advanced computing architectures and platforms, exploiting robust databases and operating systems in distributed, client/server networks. The Internet is the most visible example of this new technology. Among the benefits provided by the technology are improved access to data in usable form, superior price-to-performance ratios based on scalable server technology, and ease of use based on well-designed graphical user interfaces and client software.

A large portion of the data that is critical to many of an enterprise's business processes is stored and managed by transaction systems. Moreover, these systems manage high volumes of transactions, the bread and butter of any company. One of the most formidable challenges that information systems managers face is the need to marry new client/server systems with these transaction systems without sacrificing data integrity, performance, or ease of use and without consuming extraordinary resources in expensive middleware and dedicated development and maintenance staff.

1.2 Existing Solutions

Current attempts to integrate modern systems and transaction systems represent a trade-off that inevitably favors one characteristic (for example, performance) over another (for example, ease of use):

- **Terminal emulation.** This solution requires the user to understand the syntax and become familiar with the user interface of a disparate system, creating a need for increased user training. In addition, the transfer of information usually requires some form of manual intervention, which introduces inefficiencies and inaccuracies. Also, the communications support for emulation of the older application must reside on the workstation.

- **Staged data.** By making a query that retrieves and stores a large amount of data on the client/server system, users have direct access to data in a familiar and intuitive environment. However, this data immediately falls out of synchronization with the host system, leading to
data integrity and transaction integrity issues that can often present larger and more intractable problems.

- **Direct access to data.** Direct access to data ensures that the client/server system is working with the most recent data and provides the user with a single point of reference. However, such access requires significant investment in middleware and application development. More important, such data access ignores the larger issue of transaction integrity, sacrificing the security, recovery, logging and synchronization that are the hallmarks of transaction systems.

In the latter two cases, the focus is on access to transaction data rather than integration with transaction applications.

The integration effort is further complicated by the need to use products and solutions from multiple vendors, which inevitably leads to integration, support, enhancement, and maintenance problems.

### 1.3 Today’s Requirements

A solution that effectively integrates transaction systems with new client/server systems must preserve and leverage the strengths of each platform with minimal trade-offs in function and ease of use:

- **Support for transaction processing monitor system.** Transaction processing applications rely on the system-level services provided by the transaction processing monitor, such as rollback, backup and recovery facilities, and logging and auditing functions.

  Recovery is repositioning, after a failure, each system component to a known stable state such that no important information is lost.

  In a transaction processing environment, for a given instance of a transaction, updates to system resources, usually databases or files, are not actually done until the transaction ends or the application requests that the updates be completed. This mechanism is known as a *syncpoint*.

  If the transaction abends or detects a problem and requests that the updates not be done, the updates are not completed. This mechanism is known as a *rollback*.

  To perform a file or database recovery, operations personnel restore an image—the backup—of the file or database and apply on that image all modifications (kept in the log files) that occur between the last backup of the file or database and the failure. File or database recovery in a transaction processing environment is mainly a system function activated by operations personnel. A complete integration solution must
take advantage of this reliable infrastructure and should not require the introduction of any additional infrastructure.

- **Transparency.** Having to switch between two types of systems imposes a tedious and error-prone process on end users. The integrated solution should present the user with a single, intuitive environment that is consistent with other client applications.

- **Reliability.** Information systems managers, quality assurance staff, and users depend on the completion of a transaction. The integrated solution must ensure that a transaction is processed or alert the appropriate staff of failed transactions.

- **Cross-platform support at all levels of processing.** Client/server systems are characterized by a plethora of operating system platforms, including Windows, OS/2, Macintosh, and UNIX. The integration solution must support the full breadth of client, server, and host platforms.

- **Mobile support.** Many client/server applications use replication technology to support mobile and remote users. Replication is a function that synchronizes client and server copies of the same database. Mobile applications that presume periods of latency are fundamentally at odds with the real-time orientation of transaction systems. This complicated problem adds a daunting level of complexity to the bridging of transaction and client/server systems.

- **Cost effectiveness.** Information systems managers will balance the risk and cost of continuing with current, incomplete integration solutions with the cost of acquiring, installing and maintaining a new solution. A complete solution therefore must represent a compelling cost-benefit proposition.

- **Support for standards.** The information systems organization depends on the ability to invest in upgrades to systems software without introducing incompatibilities with existing systems. Therefore, an integration solution should be based on published standards and application programming interfaces (APIs) that can be expected to be supported by future platform changes.

- **Accessibility.** To leverage network computing, an enterprise’s application systems must be accessible to authorized users any time, from any place. The integrated solution therefore must support multiple forms of connectivity, for example, public and private networks, intranets, and the Internet.
1.4 Lotus Notes Solutions

With millions of licenses issued, Lotus Notes is one of the leading client/server systems today. Lotus Notes is an environment for the development and deployment of groupware applications. Groupware applications provide groups of users with communications, collaboration, and coordination capabilities. Many organizations have identified the opportunity to integrate Lotus Notes as a partner with host-based transaction processing systems. The benefit of extending access to existing host transaction applications to Lotus Notes is to provide users with a single point of access to those systems and to client/server applications, as well to enable developers to build applications that use both the structured data of transaction systems and the unstructured data of groupware.

In fact, for many users, Lotus Notes has become the computing home in which they spend the majority of their time. Lotus Notes, with its complete integration with the Internet, provides for groupware applications, client/server mail, integration with desktop productivity tools. Its integration with third-party databases makes it a robust, familiar environment in which to launch and run a wide array of business applications. Lotus and IBM's Hursley Lab have worked together to develop a set of technologies that integrate Lotus Notes with transaction processing systems, allowing Lotus Notes to act as the home for transaction systems.

The integration technologies remove the constraints of partial solutions, providing an information system organization with robust, transaction-oriented solutions for integrating their application systems with Lotus Notes. These components leverage existing transaction processing systems, requiring no changes in technology or business processes to conduct backup and recovery, logging and auditing, system measurement, workload balancing, or performance monitoring. System security also remains unchanged.

No less important is the ability of Lotus Notes to extend access to Lotus Notes and transaction systems to the ultimate end user over the Internet. This access is made possible with the advent of Domino, a new server technology that transforms Lotus Notes into an Internet applications server, allowing any Web client to participate in Lotus Notes applications securely.

1.5 MQEI

Mobile support and cross-platform support are inherent features of Lotus Notes. Transaction processing support is provided by such products as CICS and IMS. Lotus Domino Server allows accessibility to your application through intranets or the Internet. Applications built using the capabilities of
both Lotus Notes and transaction processing systems facilitate the integration of workflow, mobile support, and Internet access with the existing capabilities of the transaction systems that run a company’s business.

The MQEI makes the task of developing and managing an extended enterprise application straightforward.

The MQEI includes:

- A set of classes that provide a Notes programmer with a simple interface to communicate with enterprise applications. These classes provide a common set of verbs, hiding details of any transaction management system being used by the enterprise application.
- Built-in capability to access unmodified enterprise applications running under CICS and IMS, as well as native MQSeries applications on other enterprise server platforms.
- Enhanced message-building facilities. Notes programmers can build and interpret messages by reading and writing named fields within the message, without having to know precise details of the message format. If appropriate, the system will provide default values for fields that programmers do not set explicitly.
- Integration of the Identification and Authentication services of Lotus Notes with the access controls provided by CICS and IMS.

These features extend Lotus Notes to include straightforward access to enterprise applications, in a manner that is consistent with its application development environment. They provide access to enterprise applications from application logic either running in a client, or running as agents on a server. In the client case, the MQEI support and associated MQSeries client (or server) or CICS client must be installed on the Lotus Notes client machine.

The programming model is simpler than the model provided by the standard MQSeries Link LotusScript extension (MQLSX) as the MQEI uses a Lotus Notes database to hold object-definition information, reducing the amount of inline programming required. Additional bridges and drivers provide access to unmodified enterprise applications.
Chapter 2. System Components

In this chapter, we introduce Lotus Notes, the MQEI, MQSeries, CICS, and IMS.

2.1 Lotus Notes

Lotus Notes is a groupware product that enables people to share and manipulate common information. Lotus Notes applications are implemented with the client/server computing model, that is, users access information stored in Lotus Notes servers by means of Lotus Notes clients. Lotus Notes clients can run applications while they are connected to the Lotus Notes server, or they can run the applications in disconnected (stand-alone) mode.

Lotus Notes is very suitable for applications where documents must be processed by multiple users. Typically, Lotus Notes is excellent for applications that require document flow and where the nature of the data, that is, the content of the documents, is rather unstructured. Also, Lotus Notes is very good for the development and execution of client/server applications and applications that require electronic messaging.

2.1.1 Notes Databases

Lotus Notes information is stored in Lotus Notes databases, which are collections of documents. Lotus Notes databases usually reside on servers, allowing multiple users to access data concurrently. Users, however, can also have Lotus Notes databases on their local workstations. Because these local databases can be replica copies of the server databases, users can work on the documents while not connected to the Lotus Notes server. The data exchange between client databases and the equivalent server databases as well as between any server database and equivalent databases on other servers is done through the replication function of Lotus Notes. Database replication is one of the valuable functions of Lotus Notes.

2.1.2 Documents

The primary commodity in Lotus Notes is a semistructured, multimedia document. A document is the fundamental unit of information. Lotus Notes uses this unique structure to organize information. The document contains fields that hold data. A document's design makes the information easy to read and convenient to use.
2.1.3 Forms
Forms allow users to create documents that store the application's data. A form can contain fields to display data, text and graphics to make the form attractive and easy-to-use, actions and hotspots that automate tasks, or pop-ups that give users extra information. Most databases have several forms, each of which serves a particular purpose.

2.1.4 Views and Folders
Views and folders display document summaries in rows and columns so users can find the documents they want to read. They can include data extracted from document fields, calculation results, or totals and averages across documents. To read documents in a database, a user opens a view, then opens the document, represented by one row. Documents that appear in a view are determined by a selection formula which is similar in function to a SQL select statement. A folder works in most ways like a view, except that users do not use a selection formula to determine which documents appear in a folder. Instead, users drag documents into a folder to make them appear.

2.1.5 Navigators
Navigators represent a graphical way for users to find documents, switch between different folders and views, change to a different navigator, or take actions. Actions could be drop a document into a folder or run a piece of code. Navigators make it possible to easily implement more sophisticated applications such as workflow and messaging.

2.1.6 Agents
Agents are used to automate tasks with Lotus Notes. An agent consists of a document selection formula, a trigger, and one or more actions. Agents can be triggered by a predetermined time (hourly, daily, weekly, monthly) or event (new mail arrives, documents are changed, or users run the agent from a menu). Agents perform routine tasks in the background such as sending out of office messages, archiving older documents, or searching the Web for updated information in areas of interest. You can program agents, using LotusScript or the Lotus Notes formula language.

An agent could be compared to an MVS batch job that is scheduled to run at a certain time or a CICS transaction that is started through CICS Interval Control to execute at a particular time of day.

2.1.7 Application Development
Lotus Notes has several ways to create functions within the Notes application: simple actions, formulas and @Functions, and LotusScript.
2.1.7.1 Simple Actions
Simple actions are useful, precoded routines. They allow you to define a sequence of actions without requiring any programming knowledge. They are ideal for the end user who wants to automate some routine tasks. In addition, they provide significant power to the developer and eliminate manual coding. Examples of simple actions are reply to a sender of mail or run an agent.

2.1.7.2 Formulas and @Functions
Notes formulas are expressions that have programlike attributes. For example, you can assign values to variables and use a limited control logic. The formula language interface to Notes is provided through calls to @Functions.

@Functions are used to perform specific tasks within Notes. For example, @Created displays the create date of a document.

2.1.7.3 LotusScript
LotusScript is an object-based programming language that is a superset of the BASIC language. You can write sophisticated scripts by using conditions, branches, subroutines, while loops, and the like. Its interface to Lotus Notes is through predefined object classes. Lotus Notes oversees the compilation and loading of user scripts and automatically includes the Notes class definitions.

You can extend LotusScript by writing your own classes or you can add additional classes such as those that are part of the MQEI. These additional classes are called LotusScript extensions (LSXs).

LSXs, including MQEI, can be used with other Lotus products, such as SmartSuite, that support LotusScript.

2.1.8 Lotus Notes Domino Server
Lotus Notes Domino server is the main approach to integrating Lotus Notes with the Web. It allows any Web browsers to be clients of a Lotus Notes server. It is an add-in task that runs on the server. It provides dynamic translation from Lotus Notes documents to Hypertext Markup Language (HTML) format.

With Domino, a user browsing the Web from any Web browser can:
- Access Lotus Notes securely
- Access dynamic data and applications based on time, database queries, and/or user identity
- Execute Lotus Notes agents on a Domino server
- Create, edit, and delete documents in a Lotus Notes database
- Search a Lotus Notes database
- View content in a Lotus Notes database, using the navigational capabilities of Lotus Notes

Domino’s target is to meet user needs not only for static publishing but also for the new class of highly interactive Internet and intranet applications that customers are now demanding. Domino transforms Lotus Notes into an interactive Web application server, allowing any Web client to participate in Lotus Notes applications securely. Domino supports all internet standards: HTML, Secure Socket Layer (SSL), Java, and Hypertext Transfer Protocol (HTTP).

Existing Lotus Notes applications, which previously were accessible only through a Lotus Notes client, can now be accessed by using a Web browser on the Internet or intranet. Domino converts on the fly Lotus Notes databases to HTML, making all Lotus Notes objects standard Web pages. This feature of Lotus Notes broadens the reach of applications, because they can be accessed through any Web browser and by anyone on the Internet. Thus the scope of Lotus Notes applications is as broad as the Internet itself.

Some of the features of Domino are:

- **Forms support**
  Domino can create forms that users can read and fill out with a Web browser.

- **Discussion databases**
  Domino discussion databases automatically become Web-based discussion databases.

- **Access control list (ACL) support**
  Domino eases security administration by using Lotus Notes’ ACLs to manage servers, databases, views, pages, and fields within pages.

- **Replication**
  Lotus Notes Domino is the only Web server that supports bidirectional replication, which enables a webmaster to synchronize all pages on Web servers.

Creating a Domino application is almost the same as creating a Lotus Notes application. Actually the Domino server can publish many Lotus Notes
applications. As HTML forms, Lotus Notes forms are used for collecting Web user input. After the user has completed the form, it is saved in the Lotus Notes database. Lotus Notes field transaction and validation formulas also work on the Web browser. Domino also translates Lotus Notes views, which can be collapsed and expanded within the Web browser. Lotus Notes navigators are also available from a Web browser, because Domino converts them into image maps with links. Security is handled as in a Lotus Notes environment through ACLs. You can add Web users to the ACL in the same way you add Lotus Notes users to it. The same levels of permission are available, and you can therefore control not only who has permission to view an individual database but also who can read, edit, or delete documents.

2.1.9 Domino.Connect

Domino.Connect integrates the power of Notes and Domino with relational databases and transaction systems. It enables you to extend your critical business applications including your Notes application to an intranet and the Internet. With Domino.Connect you can integrate traditional applications, legacy systems, and relational databases with e-mail, workflow, groupware, and the Internet.

Domino.Connect enables Lotus Notes clients and Web browsers to access enterprise data and applications, including enterprise resource planning application systems such as SAP R/3; traditional transaction processing systems such as CICS, MQSeries, and IMS; and relational database systems such as DB2, Oracle, and Informix. Using Domino.Connect, you can incorporate information shared in databases, enterprise resource planning systems, and transaction systems in your Domino-based Web sites and applications. Applications built with Domino and Domino.Connect support multiple levels of secure, distributed access, built-in application and database integration with complete messaging, calendaring and scheduling, as well as replication services to support the most strategic, dynamic, and interactive Internet and intranet applications.

Domino.Connect provides application developers with the tools to enable bidirectional data exchange between Domino applications and back-end databases and transaction systems. You can create applications that use a Web browser or Notes client user interface to seamlessly update databases or create complex transactions with the back-end system. Domino.Connect is easy to use because of its broad support of the most popular enterprise systems.

Domino.Connect provides you with three ways of bringing a company to the Internet or intranet:
• Relational database connectivity
  Customers can access and update data in non-Notes database management systems (DBMSs). Domino.Connect includes:
  − Lotus NotesPump 2.0
    A data distribution server that offers high-volume data exchange and synchronization with DB2, Oracle, Lotus Notes, ASCII/text, Sybase, and open database connectivity (ODBC) databases
  − LotusScript Data Object (LS:DO)
    Provides real-time data access from Domino through ODBC access to external databases
  − Oracle LSX
    Provides real-time data access through Oracle’s native protocols
  − DB2 LSX
    Provides real-time data access through native call level interfaces (CLIs)

• Transaction systems connectivity
  Runs mission-critical applications and accesses data within the context of the business rules in today’s high-volume production and legacy systems. Domino.Connect applications can access more than 18 different platforms and systems, including CICS and IMS, through the MQLSX and the MQEI

• Application systems connectivity
  Brings popular enterprise applications to the Web. Domino.Connect features direct access to SAP R/3 through a LotusScript BASIC plug-in based on both remote function calls and SAP R/3’s Business API. This plug-in enables bidirectional data exchange and is capable of both synchronous and asynchronous activity.

2.2 MQEI

In this section we describe the MQEI system structure, the various ways in which the MQEI integrates with enterprise systems, and MQEI security.

2.2.1 System Structure

Figure 1 on page 13 illustrates the MQEI system structure.
A new plug-in or LSX provides the LotusScript programmer with the MQEI interface classes. The two key classes from this interface are the EIService and EIMessage classes. The EIService class provides a universal interface suitable for driving a variety of styles of back-end enterprise applications. It can support single-shot or multiple-interaction conversations and can be used in either synchronous or asynchronous mode. The LotusScript programmer instantiates an object from this class in order to communicate with a back-end enterprise application. Individual communications with this application are then effected through the EIService objects. The EIMessage objects encapsulate the parameter (field) flows to and from the enterprise application.

Objects from the EIService and EIMessage classes derive their characteristics from tables held in the MQEI Definition database. This database provides a library of predefined objects; the definition for an EIService object contains information such as the type of transaction system on which it runs, and the communications mechanism used to access it.
The LotusScript programmer does not have to be aware of this information, as it does not have to be coded in the LotusScript application itself.

MQEI includes tools to create and manage the MQEI Definition database; the database itself may be built by the Notes programmer, or by a systems integrator who is familiar with the back-end enterprise applications and has access to them. As it is a regular Lotus Notes database, the MQEI Definition database can be replicated from a central location to all systems that require it.

The MQEI Definition database also contains definitions for the EIMessage objects required. An EIMessage object consists of a set of typed, named fields. These fields are presented to the LotusScript programmer as properties of the EIMessage object. The LotusScript programmer can read and write fields in the message, using simple assignment statements, knowing nothing more than the names of the fields in question. This is directly analogous to the interface that Lotus Notes itself provides to the individual fields within a document.

The MQEI Definition database also holds mapping information, stating how the individual fields identified in a particular EIMessage definition are to be mapped into data structures understood by existing applications. Like the information used to control the operation of the EIService object, this information is stored inside the database and does not have to be coded in the LotusScript program.

2.2.2 Application Access

When the LotusScript application constructs a message, using the EIMessage class, and submits it through an EIService object, MQEI invokes a service to access the transaction itself. MQEI includes a variety of services. The choice of service is governed by the EIService object definition. The first services to be provided are:

- Native MQSeries service, which interfaces to MQSeries-aware enterprise applications.
  
  MQSeries is required for this service.

- IMS via MQSeries service, which allows access to unchanged IMS 3270-oriented transactions through the MQSeries-IMS bridge capability built into MQSeries for MVS/ESA V1.1.4
  
  MQSeries is required for this service.

- CICS DPL via MQSeries service, which provides access to CICS programs based on distributed program link (DPL). These programs are accessed through MQSeries using the newly provided
MQSeries-CICS/ESA DPL bridge, a CICS/ESA application that responds to incoming MQSeries messages and invokes unmodified CICS DPL-based programs.

MQSeries is required for this service.

- CICS DPL direct service, which provides access to CICS DPL-based programs through the CICS client through the CICS external call interface (ECI).

  A CICS Client is required for this service.

- CICS 3270 direct service, which provides access to CICS 3270-oriented transactions through the CICS external presentation interface (EPI).

  A CICS Client is required for this service.

### 2.2.3 Security

Access to enterprise applications may be restricted to certain password-protected user IDs defined and managed on the enterprise system. In the case of IMS and CICS/ESA transactions, these user IDs are managed by RACF or an equivalent security manager. Notes has its own extensive security mechanism that uses both passwords and public key technology. MQEI brings both together through its MQEI Security database, a Notes database containing the host user IDs and authenticators (passwords) an end user will need to run back-end transactions. As the authenticator itself is sensitive information, it can be encrypted, and access to it can be restricted using Notes security.

When an MQEI application wants to communicate with an enterprise application, through an EIService object, the underlying service extracts the user ID and password required from the MQEI Security database and uses them to authenticate the end user with the enterprise system. This authentication is performed by the MQSeries-IMS bridge, the MQSeries-CICS/ESA DPL bridge, or CICS. This authentication is transparent to the MQEI application.

### 2.3 MQSeries

MQSeries products enable applications to use message queuing to participate in message-driven processing.

Message queuing is a method of program-to-program communication. It allows programs to send and receive information without having a direct connection established between them. Programs communicate by putting messages on message queues and by taking messages from messages queues (see Figure 2 on page 16).
The most important characteristics of message queuing are:

- Time-independent (asynchronous) communication
  The exchange of messages between the sending and the receiving program is time independent. The sending program can continue processing without having to wait for an acknowledgment of the receipt of the message. MQSeries holds the message in the queue until it is processed.

- Connectionless communication
  Sending and receiving programs only use queues to communicate. All activity associated with such communication—maintaining queues, maintaining the relationships between programs and queues, handling network restarts, and moving messages around the network—is the responsibility of MQSeries.

- Parallel processing
  MQSeries allows a one-to-one relationship between communicating programs, but it can also support application structures and message-flow patterns that are much more complex, such as many-to-one, one-to-many, or any combination of those relationships.

MQSeries is a family of products for cross-network communication. It is available, with others being added on a regular basis, on the following platforms:

- Host
  - IBM MVS/ESA Server and Client enabled

---

MQSeries products enable programs to talk to each other across a network of unlike components—processors, operating systems, subsystems, and communication protocols—through a simple and consistent common API, the message queue interface (MQI).

Messages and queues are basic to MQSeries. Programs communicate by sending each other data in messages rather than by calling each other.
directly. Messages are placed on queues in storage, so that programs can run independently of each other, at different speeds and times, in different locations, and without having a logical connection between them.

2.3.1 Messages
A message is a string of bits and bytes that has meaning to one or more application programs. The data is classified into application data, which is the data that has a meaning to the application, and control information, which is the data that specifies the properties of the message. The message queueing service uses control information to decide how the message should be processed.

Messages can also contain information that tells the application processing the message where to send a reply. This is done by storing in the message the queue manager and message queue to which the reply is to be sent.

2.3.2 Queue Managers
A queue manager is the system service that provides the message queueing facilities used by applications. Applications can issue the message queueing calls with a local queue manager.

Each queue manager is known by its name, which must be unique within a network of interconnected queue managers so that each queue manager is unambiguously identified.

2.3.3 Message Queues
A message queue is a named object in which messages accumulate and from which messages are later removed. The queue belongs to a particular queue manager, which is responsible for the maintenance of that queue. The queue is identified by its name, which must be unique within the queue manager.

A message queue orders messages as a queue, that is, when messages are added to the queue, they are added at the end, and when messages are removed from the queue, they are removed from the front. However, it is also possible to read the messages in the queue other than in arrival order.

Queues can order messages by either arrival time or priority. In the priority queues, messages are added to the queue in front of lower-priority messages.

The physical management of message queues is entirely the responsibility of the queue manager, and the application program is unaware of such details.
Message queues can be either local, belonging to the local queue manager, or remote, belonging to a queue manager other than the local queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues.

2.3.4 Channels

A channel is a communication link used by distributed applications. There are two types of channels in MQSeries:

- Message channels, which are unidirectional. They transfer messages from one queue manager to another.
- MQSeries channels, which are bidirectional. They transfer MQSeries calls from an MQSeries client to a queue manager, and responses from a queue manager to an MQSeries client.

Programs are not aware of channels. However, the design of an MQSeries environment will have to include channels.

2.3.5 Triggering

The queue manager defines certain conditions as constituting "trigger events." If triggering is enabled for a queue and a trigger event occurs, the queue manager sends a trigger message to a queue called an initiation queue. The presence of the trigger message on the initiation queue indicates that a trigger event has occurred.

The program that processes the initiation queue is called a trigger-monitor application, and its function is to read the trigger message and take appropriate action based on the information in the trigger message. Typically such an action would be to start some other application to process the queue that caused the trigger message to be generated. From the point of view of the queue manager, there is nothing special about the trigger-monitor application: it is simply another application that reads messages from a queue (the initiation queue).

If triggering is enabled for a queue, that queue must have associated with it a process definition object. This object contains information about the application that is to be started to process the message which caused the trigger event. When the queue manager generates the trigger message, it extracts this information from the relevant process definition and places it in the trigger message, for use by the trigger-monitor application. The name of the process definition associated with a queue is given by the ProcessName attribute of the queue definition. Each queue can specify a different process definition, or several queues can share the same process definition.
Triggering involves:

- **Application queue**
  An application queue is a local queue. When it has triggering set to on and the conditions are met, trigger messages are written to the initiation queue.

- **Process definition**
  Each application queue has a process definition associated with it that holds details of the application that will get messages from the application queue.

- **Trigger event**
  A trigger event causes a trigger message to be generated by the queue manager. This event is usually a message arriving on an application queue, but it can also occur at other times. MQSeries has a range of options to allow you to control the conditions that cause a trigger event.

- **Trigger message**
  The queue manager creates a trigger message when it recognizes a trigger event. It copies into the trigger message information about the application to be started. This information comes from the application queue definition and the process definition associated with the application queue.

- **Initiation queue**
  An initiation queue is a local queue on which the queue manager puts trigger messages. A queue manager can own more than one initiation queue, and each initiation queue is associated with one or more application queues.

- **Trigger monitor**
  A trigger monitor is a continuously running program that serves one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. The trigger monitor uses the information in the trigger message to issue a command to start the application that is to retrieve the messages arriving on the application queue, passing it information contained in the trigger message header, which includes the name of the application queue.

Figure 3 on page 21 illustrates MQSeries triggering.
Figure 3. MQSeries Triggering

Triggering involves the following steps:

1. Program A puts a message to an application message queue.
2. MQSeries uses the process definition for the message queue to format a trigger message.
3. MQSeries writes the trigger message to the message queue’s initiation queue.
4. The trigger monitor gets the trigger message from the initiation queue.
5. The trigger monitor starts program B to process the message from the application message queue.
6. Program B gets the message and processes it.

2.3.6 Distributed MQSeries

In this section we describe MQSeries clients and MQSeries distributed queuing.

2.3.6.1 MQSeries Clients

An MQSeries client is an independently installable component of the MQSeries product that enables you to run MQSeries applications, where the MQSeries client, by means of a communications protocol, can interact with
one or more MQSeries servers and connect to their queue managers (see Figure 4 on page 22).

Figure 4. MQSeries Client

An application that you want to run in the MQSeries client environment must first be linked with the relevant client library. When an MQSeries call is issued by the application, the MQSeries client code directs the request to a queue manager on an MQSeries server, where it is processed and a reply sent back to the MQSeries client. From an application's point of view there is no difference between running the application on an MQSeries client or an MQSeries server.

Using MQSeries clients is a cost-effective way of implementing MQSeries messaging and queuing. You can have an MQSeries application running on one machine and the queue manager running on a different machine. The benefits of this approach are:

- No need for a full MQSeries implementation on the client machine
- Reduced hardware requirements on the client system
- Reduced system administration requirements
- The MQSeries application can connect to multiple queue managers on different systems

2.3.6.2 MQSeries Distributed Queuing
Distributed queuing is the part of MQSeries that allows applications to put messages on a queue of another queue manager that may be located on another MQSeries server (see Figure 5 on page 23).
Figure 5. MQSeries Distributed Queuing

Figure 5 demonstrates distributed queuing. Program A puts a message using the MQSeries MQPUT command to logical queue A on queue manager QM1. However, queue A is not physically on QM1; it is physically located on queue manager QM2. Program B puts a message to logical queue B on queue manager QM2, which is physically located on queue manager QM1. MQSeries automatically routes the message to the correct queue manager and queue, assuming that you have defined to the local queue manager where the queue is actually located and have specified channels to transfer messages between queue managers.

2.3.7 MQSeries Bridges

Bridges are available to connect MQSeries to existing environments such as CICS, IMS, or SAP. See 2.4.5, “MQSeries-CICS/ESA DPL Bridge” on page 27 and 2.5.3, “MQSeries-IMS Bridge” on page 30.
2.4 CICS

The CICS family of products is a set of transaction processing products that run in a variety of operating systems. CICS provides an environment for the development and execution of business applications.

The business applications running in a CICS system consist of a set of CICS transactions, which are often executed by many users at the same time. A CICS transaction consists of one or more CICS programs. CICS provides functions that allow users to concurrently execute those transactions as well as functions to ensure the consistency and integrity of data that those transactions access. These properties of a transaction system are known as the ACID properties:

- **Atomicity**
  A transaction’s changes to state are atomic: either all happen or none happens.

- **Consistency**
  A transaction is a correct transformation of state. Enterprise data represents the state of the enterprise and any change must keep the data consistent.

- **Isolation**
  Even though transactions execute concurrently, each transaction appears to run by itself, either before or after the others.

- **Durability**
  Once a transaction complete successfully (commits), its changes survive failures of any kind.

Another important concept of transaction processing systems is the concept of unit of work. A CICS transaction usually consists of one or more units of work; that is, a recoverable sequence of operations within an application process. A unit of work is the basic building block used by the online transaction processing (OLTP) system to ensure that the data is in a consistent state. Any reading or writing to the file or the database is done within a unit of work. A point of consistency (or commit point) is a time when all recoverable data that an application accesses is consistent with related data.

For example, a bank transaction might involve the transfer of funds from a savings account to a checking account. After the application subtracts an amount from the savings account, the two accounts are inconsistent; they are not consistent again until the amount is added to the checking account.
When both steps are complete, the point of consistency is reached, and the changes are committed and made available to other applications.

Today's existing CICS applications were developed to support such formal business processes of enterprises as order handling and purchasing.

CICS transactions typically make up the business-critical applications of an enterprise.

We now explain the components of CICS that you need to understand if you are going to use the MQEI to connect with CICS.

2.4.1 Communication Area

The CICS communication area, the COMMAREA, is the data area that can be passed to CICS programs when the programs are called by another program. The calling program could be a CICS program using the DPL API or the ECI API.

The invoked program receives the data as a parameter. The program must contain a definition of a data area to allow access to the passed data.

2.4.2 Distributed Program Link

DPL enables a CICS application program to link to a program on a remote CICS system and pass the CICS program a COMMAREA. The linked-to program executes and returns control to the calling program. It can be thought of as a form of a remote procedure call (RPC). The linked-to system can run any of the CICS family of products.

A server program can itself issue a DPL and act as a client program with respect to the program to which it links.

If DPL is performed between CICS on a workstation such as CICS for OS/2 or CICS for AIX and either a CICS/ESA or a CICS/400 system, COMMAREA data can be automatically converted between ASCII and EBCDIC if required.

2.4.3 Basic Mapping Support

Basic mapping support (BMS) is an interface between CICS application programs and terminal devices.

A CICS application program uses BMS and non-BMS commands to perform input and output. BMS has most of the facilities you will need and is easier to use than the non-BMS commands. Nevertheless, you might sometimes need to use non-BMS commands.
BMS lets you separate the tasks of display design and CICS application programming. It interprets generalized device-independent application program output commands and generates device-dependent data streams for specific output devices. It also transforms incoming data streams into a form acceptable to application programs. BMS learns about the format of the data stream for the terminal from the terminal control table terminal entry (TCTTE) for the task, not from the application program.

BMS commands are quite simple because all of the low-level formatting information is held separately in maps. Therefore your application programs are easier to write and less affected by changes to the system or its devices. You can change the layout of the information on the terminal device just by changing the maps.

2.4.4 CICS Clients

CICS clients are currently available for DOS, Windows 3.1, Windows 95, Windows NT, Macintosh, OS/2, AIX, Solaris, SINIX, HP-UX, and Digital UNIX. From a development perspective, they support two APIs, the EPI and ECI, that enable you to develop applications that communicate with all the CICS servers.

2.4.4.1 External Presentation Interface

The CICS EPI enables existing CICS applications to use client applications such as graphical user interface (GUIs) on workstation-based CICS clients without the need to change CICS applications.

The CICS server application sends and receives 3270 data streams (for example, a CICS BMS transaction) to and from the client application as though it were conversing with a 3270 terminal. The client application captures this data and processes it as desired, for example, displaying the data with a non-3270 presentation product such as a GUI.

2.4.4.2 External Call Interface

The CICS ECI enables CICS applications to be designed with the business logic on the CICS server and the presentation logic on the workstation-based CICS clients.

The ECI is a remote call from a workstation's application to a CICS program on a server. ECI enables a non-CICS client application to call a CICS application synchronously (that is, the calling program waits for a response from the linked-to program) or asynchronously (that is, the two programs continue to execute independently) as a subroutine. The client application communicates with the CICS server program, using the COMMAREA.

At the CICS server, the ECI looks like a DPL from a partner CICS system.
Using the ECI, any program on the workstation can access facilities on any CICS server. The ECI provides maximum flexibility for the client/server environment because you can change the client without affecting the server and vice versa.

2.4.5 MQSeries-CICS/ESA DPL Bridge

The MQSeries-CICS/ESA DPL bridge enables an application, not running in a CICS environment, to cause a CICS program to run and provide a response back to the application.\(^2\) The application making the initial request can be run from any environment that can put a message onto an MQSeries queue, such as a Lotus Notes application using MQEI. Once the request message is on an MQSeries queue, it must be transmitted to the MVS/ESA system where the bridge is running.

The bridge enables your non-CICS application to:
- Issue a CICS DPL request through the MQI
- Run one or more CICS programs within a unit of work
- Get feedback from the CICS program that it can use

The MQSeries-CICS/ESA DPL bridge runs in the same MVS/ESA image as MQSeries and CICS. It consists of:
- A single MQSeries-CICS/ESA DPL bridge monitor task
- An MQSeries-CICS/ESA DPL bridge task for each unit of work

To run, the bridge requires an MQSeries queue to hold the requests sent by your non-CICS application. Also, if the calling application wants responses, it requires at least one MQSeries queue to hold them.

The MQSeries-CICS/ESA DPL bridge requires:
- MQSeries for MVS/ESA Version 1.1.4 or later
- CICS for MVS/ESA Version 3.3 or later

The MQSeries-CICS/ESA DPL bridge requires both MQSeries and CICS to run in the same MVS image. The MQSeries request queue must be local; however, the response queue can be local or remote.

Figure 6 on page 28 shows the steps taken to process a single message.

The process consists of these steps:

1. A message, with a request to run a CICS program, is put on the request queue.
2. The CICS DPL monitor task, which constantly browses the queue, recognizes that a start unit of work message is waiting.
3. Relevant authentication checks are made. If authorization is granted, a bridge task is started with the appropriate authority.
4. The CICS DPL bridge task removes the message from the request queue, builds a COMMAREA from the data in the message, and issues an EXEC CICS LINK for the program requested in the message.
5. The program returns the response in the COMMAREA used by the request.
6. The MQSeries-CICS/ESA DPL bridge task reads the COMMAREA, creates a message, and puts it on the ReplyToQueue specified in the request message. The MQSeries-CICS/ESA DPL bridge task ends.

When you run the MQSeries-CICS/ESA DPL bridge, you can specify the level of security you want CICS to use when accessing resources. If required, the user ID is taken from the MQSeries Message Descriptor (MQMD). The options that include password (or passticket) validation require a
MQSeries-CICS/ESA DPL bridge header (MQCIH). Needless to say, the more validation that takes place, the greater the impact on the performance of your systems. However, you may have more than one monitor task running per CICS system, so you can run each of them with different levels of authority to suit your needs.

2.5 IMS

Information Management System (IMS) is a program that makes accurate, consistent, timely, and critical information available to many end users. It consists of a database system, IMS Database Manager (IMS DB), and a data communication system, IMS Transaction Manager (IMS TM). Both run under MVS/ESA. Together they create a complete OLTP environment providing continuous availability and data integrity.

IMS DB processes concurrent database calls, providing excellent performance for a wide variety of applications ranging from those with moderate volumes and complex data structures to those with extremely high volumes and simple data structures. It provides database support for both IMS TM and CICS transactions.

IMS TM provides high-volume, high-performance, high-capacity, low-cost transaction processing for both IMS DB and DB2 databases. IMS TM supports very large numbers of terminal sessions at extremely high transaction volumes. It enables you to use terminals or other devices or subsystems (such as Lotus Notes applications using MQEI) to enter transactions that initiate application programs which access IMS DB or DB2 databases and return results.

You can define a variety of online processing options. For example, you can define transactions for high-volume data-entry applications, others for interactive applications such as a Lotus Notes application, and still others to support predefined queries. IMS TM supports a wide variety of terminals and devices. It also enables you to develop a wide range of high-volume, rapid-response applications, and to geographically disperse your data processing locations, while keeping centralized control of your database.

2.5.1 Input and Output Messages

IMS TM provides business solutions for cooperative processing, distributed database processing, and continuous operation. It enhances system management, simplifies network administration, and manages and secures the IMS terminal network. It also routes messages from terminal to terminal, from application to application, and between application programs and terminals. Finally, it queues input and output messages and schedules
messages by associating programs with the transactions they are to process.

Of the input messages, two are of interest for our environment:

- A transaction, which is processed by an IMS application program. Its transaction code determines the application program that is the destination of the message.
- A command, which directs IMS to perform a function such as displaying the status of an IMS system resource. A command is recognized with a slash (/) in the first character of the input message.

After an IMS application has completed processing a transaction, it places any output messages on a queue either for the user or application from which the transaction came, or for an alternative destination.

### 2.5.2 Open Transaction Manager Access

IMS TM provides multiple facilities to help balance the workload and improve the system's throughput. One of them, of special interest for our environment, is called IMS Open Transaction Manager Access (OTMA). OTMA is a new component of IMS Version 5, providing an access path and an interface specification for sending and receiving transactions and data from IMS. It enables host-based communications servers, such as MQSeries, to access IMS TM applications through the MVS Cross Systems Coupling Facility (XCF).

From a client application, using OTMA, you can:

- Run existing IMS application programs without modification
- Issue most IMS commands and receive responses as a result of those commands
- Indicate that no security checking is to be done for messages, thereby minimizing security-processing overhead.

### 2.5.3 MQSeries-IMS Bridge

Since the availability of MQSeries for MVS Version 1.1.4, IMS applications can run unchanged, using the IMS programming interface to get to MQSeries queues. MQSeries provides a bridge function, which enables an MQSeries application to deliver messages to the IMS queues where they are handled with the traditional IMS programming interface—get unique (GU) and insert (ISRT) calls. This bridge is an OTMA client.

Figure 7 on page 31 shows how the MQSeries-IMS bridge function works.
When an IMS transaction is started, any data entered on a 3270 screen is made available to the application through the IMS GU call. The application sends its response back to the terminal through the IMS ISRT call.

An MQSeries application can cause the same IMS transaction to be scheduled by using the MQSeries-IMS bridge, and the MQSeries message data becomes input to the IMS GU call. The data returned by the ISRT call is put into the reply-to queue, where the MQSeries application can retrieve it, using a standard MQGET call.

After the user application puts a message to a queue, the process is as follows (step numbers refer to Figure 7):

1. The MQSeries-IMS bridge code, running in the queue manager address space, issues an MQGET of the message. The MQSeries-IMS bridge uses the OTMA interface to pass the information to IMS and waits for an acknowledgment.

Steps 2 and 3 are independent and can occur in parallel:

2. The MQSeries-IMS bridge commits the get of the message.

3. IMS schedules the IMS application that processes the transaction in a message processing region (MPR).
The IMS application issues a GU call to obtain the information passed to it.

The IMS application processes the message and issues an ISRT call to send the reply back to the IMS queue.

IMS sends the reply to the MQSeries-IMS bridge (running in the queue manager address space), using the OTMA interface.

The MQSeries-IMS bridge uses the MQPUT1 call to put the message to the reply-to queue.

Then, the user application program issues an MQGET to retrieve the message and continues its processing.

Note: IMS Version 5 is required to run the MQSeries-IMS bridge. With IMS Version 4, you can still connect Lotus Notes applications to IMS transactions using the IMS explicit API support of MQSeries programming interfaces. The Native MQSeries service should be used here.

### 2.6 Components of the ITSO MQEI Sample Application

To demonstrate the use of the MQEI we implemented a sample application that uses MQEI and several back-end applications. Our intent was not to demonstrate how to develop a Lotus Notes application or an enterprise application but to show how the MQEI can be used to integrate the two types of applications. Our sample includes a Lotus Notes application and a CICS application. For the most part we used the samples provided with the CICS gateway for Lotus Notes, part of the IBM CICS Client. It does not include the IMS application as it is part of the installation verification procedure shipped with IMS.

The ITSO MQEI sample application allows a Lotus Notes or Web user to display, add, update, and delete information. We implemented the same user interface for both the Lotus Notes client and the Web client.

The components used to test the CICS DPL direct service, the CICS 3270 direct service, the Native MQSeries service, and the CICS DPL via MQSeries service are:

- A Lotus Notes front-end application that uses the MQEI
- CICS COBOL programs and BMS
  - A COBOL program, VSAMSERV, which contains the business logic to create, update, delete, or query account information
  - A CICS COBOL front-end program, TECHPROG, which handles the map processing and then links to program VSAMSERV.
  - A CICS BMS file
These components are shipped with IBM CICS gateway for Lotus Notes, part of the IBM CICS Client.

- A CICS C language program, AMQSCIC1, which contains MQSeries calls, reads the messages coming from Lotus Notes, and links to program VSAMSERV.

All sources of the CICS programs are listed in Appendix D, “CICS Program Listings” on page 255.

The components used to test the IMS via MQSeries service are:

- A Lotus Notes front-end application that uses the MQEI.
- The Phone Book Application, which contains the business logic to add, change, delete, and display phone information. This application is part of the INSTALL/IVP facility of IMS/ESA Version 5 (refer to IMS/ESA V5 Install Volume 1, SC26-8023-00, for more information).

Figure 8 shows the components used in our sample application.
We used the following MQEI services in our sample:

- **Lotus Notes to CICS**
  - CICS DPL direct service
    The Lotus Notes application calls CICS program VSAMSERV with a COMMAREA. VSAMSERV processes the request and returns an updated COMMAREA to Lotus Notes. Lotus Notes updates a document.
  - CICS 3270 direct service
    A Lotus Notes application sends a 3270 data stream to CICS transaction TECH. CICS transaction TECH invokes program TECHPROG, which processes the 3270 data stream just as if the input came from a CICS terminal. TECHPROG calls program VSAMSERV with a COMMAREA. VSAMSERV processes the request and returns an updated COMMAREA to TECHPROG, which sends a BMS map. Instead of the map being displayed on a terminal, a 3270 data stream is sent to Lotus Notes. Lotus Notes processes the 3270 data stream and updates a document.
  - Native MQSeries service
    The Lotus Notes application writes to a MQSeries message queue. A CICS transaction, MQ1, is started through MQSeries triggering. CICS program AMQCIC1 then executes to get the messages from the MQSeries message queue and calls program VSAMSERV to process the message. VSAMSERV then returns to AMQCIC1, which puts a reply back to a MQSeries message queue. Lotus Notes gets the message and updates a document.

We used CICS for our sample application, but we could have used any platform for which there is an MQSeries product.

- **CICS DPL via MQSeries service**
  Lotus Notes writes to an MQSeries remote message queue, which is located on MVS where the CICS/ESA region is located. The MQSeries-CICS/ESA DPL bridge monitor starts a CICS DPL bridge task, which gets the message and calls program VSAMSERV to process the message. VSAMSERV then returns to the MQSeries-CICS/ESA DPL bridge task, which puts a reply back to a remote MQSeries message queue which is located on OS/2. Lotus Notes gets the message and updates a document.

- **Lotus Notes to IMS**
  - IMS via MQSeries service
Lotus Notes writes to an MQSeries remote message queue, which is located on MVS where the IMS region is located. The MQSeries-IMS bridge gets the message and uses the OTMA interface to pass the information to the IMS Phone Book Application. The application processes the message and returns to the MQSeries-IMS bridge task, which puts a reply back to a remote MQSeries message queue located on OS/2. Lotus Notes gets the message and updates a document.

Figure 9 shows our test environment.

Our test environment was as follows:

- Servers
  - Windows NT Workstation 4.0 running Lotus Notes Domino Server Version 4.5
  - OS/2 Warp 4.0, our local back-end server, running:
    - IBM Transaction Server for OS/2 Warp Version 4
    - MQSeries for OS/2 Version 2.0.1
  - MVS/ESA system, our remote back-end server, running:
- MQSeries for MVS/ESA Version 1.1.3
- CICS/ESA Version 4
- IMS/ESA Version 5

- Clients
  - Windows NT Workstation 4.0 running Lotus Notes Client Version 4.5, Netscape Navigator 3.01 for Windows NT, CICS and MQSeries client for Windows NT
  - OS/2 Warp 4.0 running Lotus Notes Client Version 4.5, Netscape Navigator 2.02 for OS/2, CICS and MQSeries client for OS/2
Chapter 3. Description of MQEI

MQEI is a convenient way of accessing your organization’s enterprise applications through a Lotus Notes interface running on your workstation, or from a Web browser if you are using a Lotus Notes Domino server.

Enterprise applications such as CICS or IMS are typically reliable, high-volume, high-performance applications that you use to run your business. The user interfaces for these enterprise applications are likely to vary from system to system. MQEI enables you to integrate your enterprise applications by using a set of objects that provide a common API.

In this chapter we describe the MQEI components and show you how they work.

The MQEI is an add-on to Lotus Notes that enables a Lotus Notes application to communicate with an enterprise service. The MQEI is a LotusScript extension and uses two special Lotus Notes databases.

3.1 LotusScript Extension

The LotusScript language is extended with five new objects that control communications and access to the enterprise service (see Figure 10 on page 38).
3.1.1 EISession Object

The EISession object is the root object for the MQEI. It defines the MQEI environment and controls all MQEI resources. There must always be one and only one EISession object defined per LotusScript instance. If you attempt to create another EISession object, you are returned a reference to the original object. The EISession object controls access to all MQEI resources. It is responsible for the creation of and access to the following MQEI objects:

- EIService
- EIMessage
- EIReceiveOptions
- EISendOptions

On the basis of the path defined in the system environment variable, the EISession object reads an initialization file (mqei.ini), which must be present for the MQEI environment definitions. The initialization file contains environment-specific information such as:

- Basic system information including the local character set and encoding
- Name and location of the MQEI Definition database (see 3.2, “Definition Database” on page 40)
• Name and location of the MQEI Security database (see 3.3, “Security Database” on page 41)

The EI_Session object is responsible for opening and closing the MQEI Definition and the MQEI Security databases.

### 3.1.2 EIIService Object

The EIIService object represents the enterprise service to which messages are sent and received. An EIIService object enables your LotusScript program to communicate with a non-Notes application or service on the same or a remote server. It provides a common API regardless of the nature of the enterprise service involved. An enterprise service can be:

- An MQSeries native application
- An IMS transaction using the MQSeries-IMS bridge
- A CICS DPL program using the MQSeries-CICS/ESA DPL bridge
- A CICS DPL program using the CICS client interface
- A CICS 3270 program using the CICS client interface

An instance of an EIIService object corresponds to a single enterprise service, however, a single service can consist of several programs or transactions. The EIIService object is responsible for:

- Establishing the connection to the enterprise system on which the service resides
- Performing authentication of security IDs with the enterprise service
- Transmitting EIIMessage objects in the appropriate format and receiving any replies
- Closing the connection

To create an EIIService object the MQEI Definition database is read to get the details of the corresponding MQEI Service definition. Among the information held about a service is the SystemName Property. If there is an MQEI Security database, the MQEI reads it using your Notes name and SystemName as the key to locate an MQEI Security definition:

- If no MQEI Security definition is found, the UserId Property is set to EIUI_NOT_APPLICABLE and the Authenticator Property (password) is set to blanks.
- If an MQEI Security definition is found, the UserId and Authenticator information is extracted and put in the appropriate EIIService properties.
3.1.3 EIMessage Object
An EIMessage object represents the message that you send to or receive from an enterprise service.

The EIMessage object allows your LotusScript program to build and interpret the message on a field-by-field basis using field names. An EIMessage object contains only user application data, not message headers such as those required by MQSeries bridges, and as such is enterprise service independent. Thus, for example, a given EIMessage object could represent a message that is sent to more than one enterprise service, assuming that the format of the user application data is the same.

To create an EIMessage object the MQEI Definition database is read to get the details of the corresponding MQEI Message definition.

There is no limit on the number of messages that can be transmitted: the number depends entirely on the enterprise service.

Your LotusScript program controls how the EIMessage is sent and received by using the EISendOptions object and the EIReceiveOptions object. The EIService, EISendOptions, and EIReceiveOptions objects provide the transmission function of the MQEI API.

3.1.4 EISendOptions Object
An EISendOptions object is used to control how an EIMessage object is sent to an enterprise service through the EIService. It gives you the flexibility to specify options in your LotusScript program, such as message priority and enterprise unit of work control.

3.1.5 EIReceiveOptions Object
An EIReceiveOptions object is used to control how an EIMessage object is received from an enterprise service through the EIService object and returned to the LotusScript program. It gives you the flexibility to specify options in your LotusScript program, such as whether to block or poll, or whether to receive a new message or return the existing message in the MQEI buffer.

3.2 Definition Database
The MQEI Definition database is a Lotus Notes database containing Lotus Notes documents that define the various message formats and enterprise service definitions. It acts as a repository for message definitions and enterprise service definitions. It is a Notes database, with a Notes interface.
that enables you to work with MQEI Message definitions and MQEI Service definitions. Figure 11 on page 41 shows the Definition database interface.

![Figure 11. Definition Database Interface](image)

You can administer the Definition database documents in the same way you would administer any other Notes databases. After creating new or modifying existing message definitions, you have to build them to improve performance. The EIMessage and EIService objects reference this database when they are created.

### 3.3 Security Database

The MQEI Security database is a Lotus Notes database containing Lotus Notes documents that define the enterprise security definitions for enterprise system users. It acts as a repository for security definitions. It is used to hold user IDs and authenticators for the enterprise systems you want to access. It is a Notes database, with a Notes interface that enables you to work with MQEI Security definitions. Figure 12 on page 42 shows the Security database interface.
You can administer Security database documents in the same way you would administer any other Notes databases. The EIService object references this database when it is created.

### 3.4 MQEI Utilities

Packaged with the MQEI are three utilities to aid in development and debugging of your application. In this section we describe those utilities.

#### 3.4.1 CICS BMS Map Conversion Utility

The CICS BMS map conversion utility enables you to convert CICS BMS map files into message and field definitions in the MQEI Definition database. These messages and field definitions should in turn be built to improve performance.

#### 3.4.2 MQEI Trace Utility

The MQEI trace utility helps the service organization identify what is happening when you have a problem. It shows the paths taken when you run your MQEI script.

#### 3.4.3 MQEI Code Level Utility

The MQEI code level utility helps the service organization verify the level of code you have installed.
In this chapter we describe the following design issues, which we had to consider during our implementation of Lotus Notes and MQEI:

- Application models
- Security
- Replication
- Recovery and restart
- MQEI and other bridge solutions

4.1 Application Models

When designing your Lotus Notes application that uses the MQEI to connect to an enterprise system, you can use either a two tier or three tier model. The pros and cons as well as the restrictions of the different models depend on the type of user you want to support in your application (see 4.1.3, “User Types” on page 46). In this section we provide a generic introduction to the two tier and three tier model of the MQEI application.

The main difference between the two tier and three tier model is the place where the LotusScript code runs. The LotusScript code uses the MQEI classes to communicate with the enterprise system. The code could run on either the Lotus Notes server or client.

4.1.1 Two Tier Model

When running the LotusScript code on the client, only two parties are involved in the front-end and back-end communication: the client and the enterprise system. It is therefore referred to as a two tier model (see Figure 13 on page 44).
Figure 13. Two Tier Model

The sequence of actions is:

1. The user enters a request for the enterprise system. (The term "request" is used here for any action on the enterprise system.)

2. The LotusScript code running on the client uses the MQEI classes to build a connection to an enterprise system.

3. The MQEI classes convert the entered request data such that it is usable by the enterprise system. The request is sent to the enterprise system.

4. The LotusScript code uses the MQEI classes to receive the reply from the enterprise system and to convert the reply data such that it is usable by Lotus Notes. The reply could be data that is queried, a return code, or any other reply from the enterprise system.

5. The reply data is displayed to the user.

Prerequisite for this implementation, where the MQEI code runs locally on the client system, is the installation of an MQSeries or CICS client that is connected to a server counterpart on the enterprise system.
4.1.2 Three Tier Model

A three tier model is an implementation with the agent running on the Lotus Notes server. The three parties involved are the client, the Lotus Notes server, and the enterprise system (see Figure 14).

![Three Tier Model Diagram]

The sequence of actions is the same as that of the two tier model, supplemented by two steps which are highlighted below:

1. The user enters a request for the enterprise system.
2. **The request data is transferred from the client to the server.** How this transfer takes place depends on the user type (see 4.1.3, “User Types” on page 46).
3. The LotusScript code running on the server uses the MQEI classes to build a connection to an enterprise system.
4. The MQEI classes convert the entered request data such that it is usable by the enterprise system. The request is sent to the enterprise system.
5. The LotusScript code uses the MQEI classes to receive the reply from the enterprise system and to convert the reply data such that it is usable.
by Lotus Notes. The reply could be data that is queried, a return code, or any other reply from the enterprise system.

6. The reply data is transferred from the server to the client.
7. The reply data is displayed to the user.

The three tier model does not require a CICS or MQSeries client to be installed on the client, nor does it require the MQEI on the client. Instead a CICS or MQSeries client (or server) and MQEI have to be installed only on the Lotus Notes server because that is the only place where the LotusScript code that uses the MQEI classes runs.

4.1.3 User Types

When you design your application with the MQEI, you should first think about the user types you want to support. Are there just static Lotus Notes clients or do you want to support mobile users as well? Do you want to design an application that is accessible from the Internet or the intranet? Do you want to have one application for different types of users?

In this section we describe the characteristics of the different types of users. We discuss the impact on MQEI usage, the restrictions you must be aware of, and the relationship between different user types and the two tier or three tier model.

4.1.3.1 Lotus Notes Client

The native environment for Lotus Notes applications is a static network with one or more Lotus Notes servers, each with one or more Lotus Notes clients.

Many Lotus Notes applications are usually designed such that the LotusScript code (or simple actions or formulas) that defines the application logic is assigned to different user interface events, such as clicking a button. These code parts, which are used for all interactive or online activities, run locally on the Lotus Notes client.

Lotus Notes supports scheduled agents that can be used for batch or offline processing. Therefore an agent can be defined to run on a certain schedule (for example, hourly or daily) or when a certain "non-user-interface" event occurs, that is, when certain documents are created, modified, or pasted. In the latter case, the agent is not activated immediately after the event, but with a delay of up to a couple of minutes because the server task that checks for these events is scheduled itself. Agents of both types, scheduled and event driven, run on the system where the database in which the agent is defined is physically stored. In a static Lotus Notes client/server environment (without mobile users), this system is usually the server.
So, there are two ways of implementing application logic in Lotus Notes:

- If an immediate response is critical to the application, an online implementation using a two tier model is required. In this implementation, the code is started immediately on certain user interface events and always runs on the client machine.

- If an immediate response from the enterprise application is not necessary, an offline implementation with a three tier model can be used. A server agent is started as defined in a schedule or with some delay after an event has occurred. In this case, the code always runs on the server.³

However, if no online connection is needed and you want to have a two tier model anyway, you can create an agent that processes all request documents and runs locally. You have to start this agent manually through the Action menu in Lotus Notes. Scheduled agents can just be defined to run on Lotus Notes servers. This batch processing may be useful as a backup solution. If the network or enterprise system is down, you might want to have a feature for delayed processing. Because not being permanently connected is characteristic of a mobile user, we discuss this feature in 4.1.3.2, “Mobile.”

Choosing an implementation of the MQEI application also has an impact on the MQEI security mechanism. In a two tier model, the logon to the enterprise system is based on the Security document in the MQEI Security database. MQEI uses the document assigned to the user currently logged on to the Lotus Notes client. In a three tier model using a scheduled agent, MQEI uses the Security document of the server on which the agent is running (see 4.2, “Security” on page 52).

### 4.1.3.2 Mobile

One of the main features of Lotus Notes is the support of mobile users, that is, users who are only occasionally connected to a Lotus Notes server. The reasons for occasional connection might be that the user is working at home or at a customer’s site instead of the company office.

Lotus Notes supports mobile users with a flexible, two-way, and secure database replication mechanism. Mobile users store on their machine replicas of databases from the Lotus Notes server. They need a temporary connection to the server established through a modem or the LAN.

---

³ Up to Lotus Notes 4.5, server agents can only be started by Lotus Notes clients on a schedule or with some delay. Support for the immediate activation of a server agent is planned in future releases of Lotus Notes.
Support of mobile users is not transparent in a Lotus Notes application using MQEI. The application has to be designed to react differently to the two different user types: mobile and permanently connected users. Instead of using the MQEI immediately, a mobile user’s request must be stored until a connection to the enterprise system is established. An agent must then be started that sends all unprocessed requests to the server. This offline processing is similar to batch processing.

Offline processing can be used only if immediate replies from the enterprise system are not critical to the application. Sequences of dependent actions, such as querying for a customer number and using that number to assign sales orders to the customer, are not applicable for mobile users. In such a case, you must redesign the application to avoid dependencies. For example, you can define different forms to create different documents as steps in a sequence, where each form contains a computed field that is used for later reassembly of the sequence. A specific agent for that sequence of actions could send one document (for example, the customer query) to the enterprise system and use the reply to update the next document in the sequence (set the customer number for the sales order). However, this approach might become complex, particularly when error handling is involved.

Lotus Notes applications supporting offline processing can use the two tier or the three tier model. In a two tier model, the agent runs on the mobile Lotus Notes client, whereas in a three tier model it runs on the server.

In a two tier model, the sequence of actions is:

1. The mobile user enters a request for the enterprise system. The request is stored as a Lotus Notes document if no connection is established.

2. When the connection is available, the user activates the agent by using the Lotus Notes Action menu or by using an action button that triggers an appropriate formula.

3. The agent uses the MQEI classes to:
   - Build up a connection to the enterprise system
   - Convert the entered request data and send it to the enterprise system
   - Receive the reply from the enterprise system and convert it

4. The agent either adds the reply data to the existing document or creates a new document.

Using a loop, this agent processes all stored request documents (see 5.1.4, “Using Scheduled Agents for a Three Tier Model” on page 78).
To support this two tier implementation, the mobile workstation must have an MQSeries or CICS client and the capability to establish a connection with the enterprise back-end system. Therefore mobile users cannot use this implementation when they connect to a Lotus Notes server through the Lotus Notes dial-up server connection. Instead they must have a connection to the enterprise LAN.

With a three tier model, the agent runs on the Lotus Notes server. The three parties involved are the mobile Lotus Notes client, the Lotus Notes server, and the enterprise system.

The sequence of actions for a three tier implementation is:

1. The mobile user enters a request for the enterprise system. The request is stored as a Lotus Notes document if no connection is established.
2. When a temporary connection is available, the document is transferred to the Lotus Notes server (see step 3).
3. An agent is started on the Lotus Notes server. This could be either a scheduled agent or an agent that is triggered when documents are created. Either way, there is a delay after the documents are transferred.
4. The agent uses the MQEI classes to:
   - Build up a connection to the enterprise system.
   - Convert the entered request data and send it to the enterprise system
   - Receive the reply from the enterprise system and convert it
5. The agent either adds the reply data to the existing document or creates a new document.
6. The document is transferred to the mobile Lotus Notes client.

The transferring of a request and reply document could be implemented either by mailing the document to a specific database on the server or more simply by using the replication concept. To use replication, two replicas of the same database must exist—one on the Lotus Notes server and one on the mobile client. Replication results in transferring the request documents to the Lotus Notes server, where an agent processes them. Afterward a second replication has to take place to transfer the document with the reply data back to the mobile Lotus Notes client. Because server agents cannot be started immediately, the time between the two replications must be at least a couple of minutes.
To support this three tier implementation, the mobile workstation needs only have the Lotus Notes client. A CICS or MQSeries client (or server) and MQEI have to be installed only on the Lotus Notes server where the agent using the MQEI classes runs.

Using MQEI classes in scheduled Lotus Notes agents has some impact on security. When an EIService object is created, MQEI fetches the enterprise user name and password (authenticator) from the Security document of the user that started the agent. In the case of a scheduled agent, the server name is used instead (see 4.2, “Security” on page 52).

4.1.3.3 Web User

With Domino, Lotus Notes is now Internet-enabled. Internet users access Lotus Notes databases with a Web browser.

The implementation of a Web-enabled MQEI application is always based on the three tier model. Three parties are always involved in the front-end and back-end communication: the Web client, the Lotus Notes Domino server, and the enterprise system. You cannot have a two tier model, where a Web client is connected to the enterprise system using the MQEI.

With a three tier model in a native Lotus Notes environment, you can only use scheduled agents or agents that are activated with some delay. In contrast to a Lotus Notes environment, in a Web environment, the Domino server supports the immediate activation of server agents when an event comes in from the Web client. Therefore, as soon as the user clicks on the button on the HTML page, the request is sent to the Domino server, and the agent is activated immediately. You can have an online connection to the enterprise system (see 5.1.5, “Implementation of Web User Access (Three Tier Model)” on page 80).

In a Web agent that is started when submitting an HTML page (representing a Lotus Notes form), you can use the MQEI to communicate with the enterprise system. The agent therefore accesses the request data entered and sends it to the enterprise system.

The sequence of actions for a Web-enabled application is:

1. The Web user enters a request for the enterprise system in a Web form (the HTML representation of a Lotus Notes form, which is automatically created by the Domino server). The request is sent to the server and stored as a Lotus Notes document.
2. A Web agent is started immediately on the Lotus Notes Domino server.
3. The Web agent uses the MQEI classes to:
- Build up a connection to the enterprise system
- Convert the entered request data and send it to the enterprise system
- Receive the reply from the enterprise system and convert it

4. The agent either adds the reply data to the existing Lotus Notes document or creates a new document.

5. The Lotus Notes document is converted to an HTML page and sent to the Web client.

The processing of delayed replies is not as comfortable as in the case of a Lotus Notes client (mobile or permanent). Processing delayed replies might be required if the enterprise system or network connection is temporarily down. In that case you might want to design the application such that requests are stored and processed later. But a delayed reply cannot be sent by simply sending it to the mailbox of the requestor, or by sending a memo with a doclink to the reply. Web users must either check a database for the reply to their requests on their own initiative or provide their e-mail addresses when sending the request. The e-mail address method could be used by an agent that sends delayed replies to the requestor, but in that case greater administrative effort would be required to convert Lotus Notes documents to SMTP documents.

The security handling of MQEI used in Web agents is similar to that used in scheduled agents. When the MQEI classes are used to establish a connection to the enterprise system, MQEI looks up the Security document (in the MQEI Security database) for the Lotus Notes server where the Web agent is running. The user ID and password that are defined in that security document are used to log on to the enterprise system. Consequently, every Web user is logged on at the enterprise system with the same user ID (see 4.2, “Security” on page 52).

4.1.4 Design Decision Table

Table 1 summarizes the criteria to take into account when creating an application using MQEI.

<table>
<thead>
<tr>
<th>Table 1 (Page 1 of 2). Three Tier and Two Tier Decision Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Three Tier</strong></td>
</tr>
<tr>
<td>CICS or MQSeries client (or server)</td>
</tr>
<tr>
<td>MQEI installation</td>
</tr>
<tr>
<td>Administrative tasks</td>
</tr>
</tbody>
</table>

Chapter 4. Design Issues 51
### Table 1. Three Tier and Two Tier Decision Criteria

<table>
<thead>
<tr>
<th></th>
<th>Three Tier</th>
<th>Two Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resource requirements</strong></td>
<td>On the server</td>
<td>On each client</td>
</tr>
<tr>
<td><strong>Logon to enterprise system</strong></td>
<td>Based on the server Security document (no user identification possible on the enterprise system)</td>
<td>Based on each Lotus Notes user Security document (user identification possible on the enterprise system)</td>
</tr>
<tr>
<td><strong>Application requirements for Lotus Notes users</strong></td>
<td>Offline implementation support only, using a server agent. Server agents triggered by a Lotus Notes client application are activated with a delay.</td>
<td>Online implementation support</td>
</tr>
<tr>
<td><strong>Application requirements for Web users</strong></td>
<td>Online implementation support, using a server agent. Server agents triggered by a Web browser application are activated immediately.</td>
<td>Not applicable</td>
</tr>
<tr>
<td><strong>Lotus Notes mobile user support</strong></td>
<td>Can use a Lotus Notes dial-up connection</td>
<td>Must use an enterprise LAN connection</td>
</tr>
</tbody>
</table>

### 4.2 Security

In this section we address the security issues to consider while designing your Lotus Notes application to access your enterprise services. We discuss how security is implemented in each environment and how MQEI handles security between Lotus Notes and the different connections to the enterprise services. Table 2 shows the different security implementation for the Web and Lotus Notes clients.

### Table 2. Security Implementation for Web and Lotus Notes Clients

<table>
<thead>
<tr>
<th>Environment</th>
<th>Web Client</th>
<th>Lotus Notes Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>• SSL</td>
<td>• Lotus Notes certificates</td>
</tr>
<tr>
<td></td>
<td>• Web user ID and password or Anonymous</td>
<td>• Authentication by the Lotus Notes Domino server</td>
</tr>
<tr>
<td></td>
<td>• Authentication by the Lotus Notes Domino server</td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td>Web Client</td>
<td>Lotus Notes Client</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lotus Notes Domino</td>
<td>• Database security (ACL)</td>
<td>▪ MOEI reads the MOEI Security definition for the Lotus Notes user that contains an enterprise service user ID and password. This definition is used to set the EIService UserId and Authenticator properties. Alternatively the LotusScript program can prompt the user for this information and set EIService UserId and Authenticator itself.</td>
</tr>
<tr>
<td>Server</td>
<td>• View and form security</td>
<td>▪ MOEI reads the MOEI Security definition for the server that contains an enterprise service user ID and password. This definition is used to set the EIService UserId and Authenticator properties.</td>
</tr>
<tr>
<td>MQEI</td>
<td>▪ MOEI reads the MQEI Security definition for the server that contains an enterprise service user ID and password. This definition is used to set the EIService UserId and Authenticator properties. Alternatively users can enter their enterprise user IDs and passwords in an HTML page. The agent, then, sets the EIService UserId and Authenticator. This is not fully secure.</td>
<td>▪ MQEI reads the MQEI Security definition for the Lotus Notes user that contains an enterprise service user ID and password. This definition is used to set the EIService UserId and Authenticator properties.</td>
</tr>
<tr>
<td>Middleware</td>
<td>• CICS Client</td>
<td>▪ MQSeries Client</td>
</tr>
<tr>
<td></td>
<td>▪ ECII</td>
<td>▪ Send on each SendMessage call from EIService UserId and Authenticator properties</td>
</tr>
<tr>
<td></td>
<td>▪ /u and /p options of the CICSCLI command</td>
<td>▪ /u and /p options of the CICSCLI command. This option must be used when your CICS client is attached to a CICS/ESA server.</td>
</tr>
<tr>
<td></td>
<td>▪ Send on each SendMessage call from EIService</td>
<td>▪ Explicitly performed in LotusScript using the CICS CESN transaction (or your local equivalent)</td>
</tr>
<tr>
<td></td>
<td>UserId and Authenticator properties</td>
<td>▪ MQSeries Client</td>
</tr>
<tr>
<td>Enterprise services</td>
<td>• MQSeries Client</td>
<td>▪ Send on each SendMessage call from EIService UserId and Authenticator properties</td>
</tr>
<tr>
<td></td>
<td>▪ Enterprise services security mechanisms such as RACF</td>
<td></td>
</tr>
</tbody>
</table>

### 4.2.1 Internet

The Web server handles the security of all Web browsers connected to it. In our case, the Web server is the Domino server. It has the same security
levels as the native Lotus Notes server. You can control how Web clients access resources on the server by:

- Allowing or not allowing anonymous Web user access
- Allowing or not allowing Web clients to browse databases on the Domino server
- Using or not using SSL security
  SSL is a security protocol that provides private communication between the Domino server and Web client.

4.2.2 Lotus Notes
Lotus Notes has several levels of security for accessing the server, database, and the contents of the database:

- Server-level security - Controls access to the Notes server
- Database-level security - Controls access to the Notes database
- Document-level security - Controls access to the Notes document
- Field-level security - Controls access to the fields within the Notes document

For more information about Lotus Notes security, refer to the "About Notes Database Security" section in the *Lotus Notes Application Developer's Guide*.

4.2.3 Middleware
Middleware is the software that connects your Lotus Notes application to the enterprise services. MQSeries and MQEI are middleware. In the case of MQSeries, security can be implemented on the queues; only authorized user IDs and passwords can access the queues.

4.2.4 Enterprise Applications
To access the enterprise application from an enterprise client, users must enter their user IDs and passwords. The security system on the enterprise system verifies the user ID and password and validates the user access rights to the enterprise applications. Each enterprise service has different security requirements, and different methods of performing authentication.

4.2.5 How MQEI Handles Enterprise Application Security
To access the enterprise application from your Lotus Notes application, MQEI translates your Lotus Notes name to your enterprise user ID and password. The MQEI provides two properties of the EIService class to enable this access:
• Userld property
  This is your user ID on the enterprise system. It may be different for each service you use.
• Authenticator property
  This is the password used to identify the user ID on the enterprise system.

There are two ways to set the Userld and Authenticator properties:

1. In LotusScript, after the EIService object has been created, you set Userld and Authenticator properties to the user ID and password you want to use to log on or sign on to the enterprise system described by the EIService.

2. MQEI uses the Lotus Notes user ID to reference the MQEI Security database to look up the associated security document for that user. Once the security document is found, MQEI reads the user ID and password for the enterprise system, using them to verify the user rights to the enterprise system.

We used three types of connections to the enterprise system, each of which has an impact on security:

• Lotus Notes user
  Any application running within a Lotus Notes client runs under the identification of the Lotus Notes user. Therefore, you must define a Security document for each user that needs to connect to an enterprise application.

• Scheduled agent
  When the agent manager runs a scheduled agent, it runs under the server's name, but it is given the same ACL rights as the agent's owner (the person who last saved the agent). If the scheduled agent needs to connect to an enterprise application, you must define a Security document for the server.

• Web application
  When a Domino application invokes an agent, it runs under the server's name. The agent has the authority of its owner when accessing the MQEI databases.4

---

4 From Lotus Notes Domino server 4.6, you can choose whether a Web agent runs with the authority of the agent owner or the Web user. If you select the latter option, you must ensure that the Web user and/or Anonymous has Reader access to the MQEI Security database.
So, for both scheduled agents and Web applications, you must:
- Ensure that the agent owner has Reader access to the MQEI Security database
- Ensure that the server has Author access to the MQEI Security database
- Create MQEI Security definitions (one per enterprise system) for the server's name

4.2.6 Security Considerations for Lotus Notes Clients

If you are developing an application that caters only to Lotus Notes clients, you must decide how you are going to communicate with your enterprise services—offline or online. The ramifications are:

- Online

  If you have to access the enterprise services online, you must install the enterprise system software client (for example, CICS Client) and the MQEI software on each Lotus Notes client, making it a so-called fat client. If the Security database is used, all users must have a security document with their enterprise system user ID and password. (The MQEI Security database must be kept synchronized with user IDs and passwords on the enterprise system.)

- Offline

  If you have to access the enterprise services offline, you can develop an agent that will run at a scheduled time to access the services and update your documents. The enterprise system software client and the MQEI software must be installed only on the Lotus Notes server, and there must be only one entry in the MQEI Security database for the agent to access the enterprise services.

4.3 Replication

One of the powerful features of Lotus Notes is its ability to replicate databases. Replication enables users on a variety of networks in a variety of locations and remote Lotus Notes users to access the same information. With this in mind, you should look at the architecture of your network and identify where the connections to your enterprise system will be. Here are the options and issues you will encounter while designing your application with MQEI:

- Single enterprise system connection (see Figure 15 on page 57)

  One Lotus Notes server serves as the gateway to your enterprise system. All unprocessed documents from other Lotus Notes servers or Lotus Notes clients are replicated to a central server that communicates
with your enterprise system, a scheduled agent processes all of the documents, and the next replication distributes them back to the other servers. This agent can run only on the gateway. It must not be replicated.

---

*Figure 15. Single Connection*

- Multiple enterprise system connections (see Figure 16 on page 58)

Multiple Lotus Notes servers or Lotus Notes clients communicate with the enterprise system and replicate among each other. In this case only processed documents should be replicated, using a replication formula. If an unprocessed document is replicated, it is likely that the document will be processed twice from the enterprise system viewpoint, once by one server and once by another server. This may also cause replication conflicts.
4.4 Recovery and Restart

Recovery, in its broadest sense, means repositioning, after a failure, each system component to a known stable state such that no important information is lost. To perform a system or application recovery, operations personnel restore an image—the backup—of that system or application and apply on that image all modifications—kept in the log files—that occur between the backup and the failure. In an enterprise application environment, recovery has mainly been a system function activated by operation personnel.

In a Lotus Notes environment, at least one replica copy of each database is maintained, but this cannot prevent a corrupted database portion from being
replicated. Because Lotus Notes does not provide a logging function, backup is done by either developing in the application itself some logging functions or using the system backup, which covers Lotus Notes databases as a portion of the whole system.

When a Lotus Notes application is integrated with an enterprise application, for backup and recovery purposes, both applications have to apply their own procedure to ensure the recoverable status of each environment. There is no coordination between the system backup implemented in Lotus Notes and the application backup implemented in the enterprise applications.

A Lotus Notes client or Domino server does not support transactional concepts, so it cannot be used to coordinate units of works on distributed systems. Each enterprise unit of work that your MQEI application initiates runs independently of others.

These facts have to be taken into account during application design as they have several implications:

- The scope of the units of work should be limited to one environment.
- The relationship between Lotus Notes data and enterprise application data should not require referential integrity. Also, if the unit of work is to cross both environments, access in one environment should be limited to read only.
- A Lotus Notes process can initiate a unit of work in the enterprise application, and an enterprise application's unit of work can initiate a Lotus Notes process. As the unit of work is independent of the Lotus Notes process, each environment can maintain data integrity.
- A Lotus Notes application can directly access enterprise application data, using data access methods such as LS:DO, DB2 LSX, or Oracle LSX. In this case, the application should access information for browsing only, not for updating. Thus a Lotus Notes environment cannot jeopardize data integrity in an enterprise application environment.
- If the results of an enterprise application's transaction are stored in a Lotus Notes document, they may be out of date if the input data of the transaction changes or a recovery of the enterprise application occurs. This should be taken into account especially when you use the replication feature of Lotus Notes.

Nevertheless, MQEI does provide some capability of controlling the enterprise units of work. If the enterprise service you want to communicate with supports transactional concepts, any message you send to the service will participate in a logical unit of work on the enterprise system such as CICS and IMS. Your LotusScript program can send multiple messages to
CICS or IMS and have them included in the same unit of work, using the UnitOfWork Property of the EISendOptions Class. See the IBM MQSeries Enterprise Integrator for Lotus Notes User Guide online for a detailed discussion on how this works.

4.5 MQEI and Other Bridge Solutions

In this section we provide a brief description of the other bridge solution available to access enterprise applications and compare them with the MQEI.

4.5.1 Link and Link Extra Technologies

Available since Lotus Notes 3.3 and therefore not using the LotusScript environment, MQSeries link for Lotus Notes, CICS gateway for Lotus Notes, MQSeries link extra for Lotus Notes, and CICS link extra for Lotus Notes implement the same link technology.

The link supports an application flow from Lotus Notes to the enterprise application. The link runs as an add-in task running on a Lotus Notes server. It uses the standard Lotus Notes API and is form based. The Lotus Notes application is coded to perform an @Mailsend function, transferring parts of the Notes document to a mail-in database associated with the link add-in task. The link add-in task constructs the MQSeries messages or the CICS ECI calls to connect to the enterprise application.

The link extra technology enables an application on any supported enterprise system to initiate the sending of data to the Lotus Notes user. It provides enterprise applications with a means of sending data to a Lotus Notes environment. The link extra is an independent process: a triggered task running on a Lotus Notes server.

The application process must support an asynchronous model. In such a model, each partner is independent. The Lotus Notes application starts a connection to the enterprise application and returns to its own process. It does not have to wait for the results of the enterprise application. Later on, when the enterprise application has completed, both partners have to resynchronize.

Table 3 compares the link, link extra, and MQEI technologies.

<table>
<thead>
<tr>
<th></th>
<th>Link</th>
<th>Link Extra</th>
<th>MQEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisite</td>
<td>Lotus Notes 3.3 or higher</td>
<td>Lotus Notes 3.3 or higher</td>
<td>Lotus Notes 4 or higher</td>
</tr>
</tbody>
</table>

Table 3 (Page 1 of 2). Link, Link Extra, and MQEI Technologies

60 Lotus Notes and the MQSeries Enterprise Integrator
4.5.2 MQSeries Link LotusScript Extension

MQLSX enables a Notes application to interact with non-Notes applications throughout the enterprise through MQSeries. It provides direct integration between Lotus Notes and MQSeries.

The MQLSX is an API that you call from LotusScript to access the MQI. This integration between Lotus Notes and MQSeries software extends the scope of Notes to include transactions and data that are part of other environments.

The MQLSX provides classes, events, and methods for MQSeries for use from within a LotusScript program.

The MQLSX code does not make calls to Lotus Notes. It is up to the application developer to handle what is updated in Lotus Notes, splitting the messages received from MQSeries into fields, and adding them to new or existing documents.
The MQLSX is designed to:

- Provide an infrastructure to enable you to develop applications that integrate your Lotus Notes environment with your traditional transaction system applications and their data
- Give you access to all the functions and features of the MQSeries API, permitting full interconnectivity with other MQSeries platforms
- Conform to the normal conventions expected in an LSX

The MQLSX can be installed on a Lotus Notes client or a Lotus Notes server. If installed on the client, the client user can request information from the target system, without going through the Lotus Notes server, giving a near synchronous response. If installed on the Lotus Notes Server, the applications are used by multiple clients through agents.

MQEI is positioned as a higher level interface that hides some of the complexity of MQSeries. However, there will always be those who need access to the lower level APIs of the MQLSX, for example, a message queue browse instead of a destructive read. Also, the MQEI has some performance overhead, which might need to be avoided in some situations.

Here are some differences between the MQLSX and MQEI:

- Using MQLSX you cannot access CICS DPL programs and CICS 3270 transactions directly through the CICS client.
- Your LotusScript program must know the MQSeries queue manager and queue names. There is no such definition database as the MQEI Definition database.
- Your LotusScript program must know the message format.
- Using MQLSX you must understand the enterprise system to which you connect, whereas MQEI provides a higher level of abstraction.
- You need to build manually in your LotusScript program the MQSeries control blocks such as the MQMD and the MQIIH. MQEI builds them automatically.
- There is no integrated support for security in MQLSX.
- The MQLSX gives you access to all functions of the MQSeries MQI. If your enterprise application makes use of some of the less used parts of the MQI, you may need to use the MQLSX.

Table 4 on page 63 compares the MQLSX and MQEI technologies.
Table 4. MQLSX and MQEI Technologies

<table>
<thead>
<tr>
<th></th>
<th>MQLSX</th>
<th>MQEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise application</td>
<td>• MQSeries application</td>
<td>• Native MQSeries application</td>
</tr>
<tr>
<td></td>
<td>• IMS through the MQSeries-IMS bridge</td>
<td>• IMS through the MQSeries-IMS bridge</td>
</tr>
<tr>
<td></td>
<td>• CICS DPL through the MQSeries-CICS/ESA DPL bridge</td>
<td>• CICS DPL through the MQSeries-CICS/ESA DPL bridge</td>
</tr>
<tr>
<td></td>
<td>• SAP R/3 through MQSeries</td>
<td>• CICS DPL direct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CICS 3270 direct</td>
</tr>
<tr>
<td>Programming on Lotus Notes</td>
<td>Full LotusScript environment with all the functions of the MQSeries MQI</td>
<td>Full LotusScript environment with some functions of the MQSeries MQI</td>
</tr>
<tr>
<td>Security</td>
<td>To be implemented in the LotusScript program</td>
<td>Implemented through the Security database</td>
</tr>
<tr>
<td>Messages</td>
<td>To be implemented in the LotusScript program</td>
<td>Implemented through the Definition database</td>
</tr>
<tr>
<td>Abstraction for the enterprise system</td>
<td>You need to understand the enterprise system to which you connect.</td>
<td>MQEI provides a higher level of abstraction</td>
</tr>
</tbody>
</table>

4.5.3 Data Access and Copy

Because MQEI accesses enterprise applications only, you may need to integrate your Lotus Notes application with either the corporate data or local copies of it. Below we describe some data access and copy technologies.

4.5.3.1 Data Access

Depending on the data source your application needs to access, three technologies are available:

- The LS:DO provides full read and write access to external ODBC data sources using LotusScript.

  The LS:DO consists of a set of three classes: ODBCConnection, ODBCQuery, and ODBCResultSet. These classes come complete with a powerful set of properties and methods and full SQL capabilities. The LS:DO is good for optimizing data entry with on-the-fly database lookups, immediate updates, input validation, and avoiding duplicate entries. LS:DO can be used to transfer low volume data. Using its cursor processing feature, you can handle result sets.
- The DB2 LSX plug-in provides native access to DB2 through the DB2 CLI. Based on the LS:DO, this plug-in is a set of three classes that access DB2 data directly as well as enable you to use DB2 extended functions, such as support for DB2 user-defined types and large objects.

- The Oracle LSX plug-in allows you to create Lotus Notes applications that read and write Oracle databases and systems by using the native Oracle Call Interface (OCI) API.

4.5.3.2 Data Copy
Lotus NotesPump is a server task that enables high-performance, scalable copy of data between relational DBMSs and Notes databases. Using NotesPump, developers, database administrators, and information technology professionals can establish frequently scheduled, event-driven, or ad-hoc bidirectional exchanges of data among corporate data store systems, including IBM DB2, Oracle, Sybase, and ODBC sources, and Notes databases.
Chapter 5. ITSO MQEI Sample Application Implementation

In this chapter we describe the implementation of the ITSO MQEI sample application that we created during the residency. This sample application is based on one simple user interface to add, read, update, and delete information in a file. We used the same user interface for both the Web user and Lotus Notes user. This should provide you with a better way of comparing the two access types.

We used one user interface for the different MQEI services: CICS DPL direct service, CICS 3270 direct service, Native MQSeries service, CICS DPL via MQSeries service, and IMS via MQSeries service. We have divided this chapter into one section for each type of MQEI service, preceded by a section on service-independent implementation considerations. You might want to restrict your reading to 5.1, “Service-Independent Implementation” and to the section that discusses the MQEI service you want to use.

5.1 Service-Independent Implementation

This section focuses on the implementation of those parts of our sample application that are not affected by the differences between the MQEI services we used.

We describe how we set up the MQEI in a Web-enabled client/server environment and how we implemented Web support as well as support of Lotus Notes clients (mobile support and scheduled agents). The description is intended to show you how to implement your own MQEI application.

5.1.1 MQEI Setup

The ITSO MQEI sample application consists of two core parts: one for the support of Lotus Notes clients and one for Web user support. To provide an online connection to the enterprise system, we implemented a two tier model for the Lotus Notes clients and a three tier model for Web users (see 4.1.3, “User Types” on page 46). Consequently we had to install the MQEI run-time library on each Lotus Notes client and on the server.

Depending on the tier model you want to use, you have to install the MQEI run-time library on either the Lotus Notes server or the Lotus Notes client.

5.1.1.1 MQEI Initialization File

The MQEI code runs on the client machine in a two tier model implementation or on the server machine in a three tier model implementation. So, to minimize the maintenance effort, we stored the
MQEI Security database and the MQEI Definition database on the server. We provided the database locations in the MQEI initialization file (mqei.ini), using the following syntax:

```
[DefinitionDatabase]
DatabaseName = MQEI\mqeidata.nsf
DatabaseType = LotusNotes
ServerName = CHINA

[SecurityDatabase]
DatabaseName = MQEI\mqeisecu.nsf
DatabaseType = LotusNotes
ServerName = CHINA
```

with DatabaseName : the subpath from the Notes data directory (MQEI) and the database filename (mqeidata.nsf, mqeisecu.nsf)

ServerName : the name of the Domino server where your MQEI databases reside (CHINA)

### 5.1.1.2 MQEI Security Database

In the MQEI Security database we created an MQEI Security definition for each Lotus Notes user who had access to the sample application. We also needed to create an MQEI Security definition for the Lotus Notes server to support connection through the Web. The Lotus Notes server MQEI Security definition is also necessary for the support of Web agents (see 4.2, “Security” on page 52 and scheduled agents). Figure 17 on page 67 shows the MQEI Security definition for a Lotus Notes user.
Figure 17. MQEI Security Definition for a Lotus Notes User

Figure 18 shows the MQEI Security definition for the Lotus Notes server used by all Web users.

Figure 18. MQEI Security Definition for the Lotus Notes Server
5.1.1.3 MQEI Definition Database
In the MQEI Definition Database we created entries to define the enterprise services with which we want to communicate and the enterprise messages we want to send and receive. There are four object definitions in this database:

- A Service definition, which defines the properties of the enterprise service
- A Message definition, which defines the structure of the EIMessage that is sent to or received from an enterprise service
- A Field definition, which describes a field within a message
- A Field Type definition, which acts as a template for one or more Field definitions.

For a detailed description of these definitions, refer to the section that discusses the MQEI service you want to use.

5.1.2 Sample Database
We implemented the sample application in one database. We used only one form to create documents and redisplay them when they were updated. We implemented several agents depending on the access type (from the Web or a Lotus Notes client) and the service (CICS DPL direct service, CICS 3270 direct service, Native MQSeries service, CICS DPL via MQSeries service, and IMS via MQSeries service)

5.1.2.1 ITSO Form
In our sample database we defined two forms, ITSO Form and IMS Form, for Web and Lotus Notes users. ITSO Form is used for all the services that access CICS. IMS Form is used for the service that access IMS. Both forms are developed with the same logic. Below we describe the ITSO Form.

In the ITSO Form (see Figure 19 on page 69), the user has to select the action (read, add, update, and delete) and the service type that he or she wants to use. Selection of service type is implemented for demonstration purposes only. You will have just one hardcoded service type. Depending on the action selected, the user has to enter values into different fields:

- Add: All fields except the account number, which is set by the enterprise system
- Read: Either the user name or the account number for which to query
- Update: All fields. The record to be updated is defined by the account number
- Delete: account number
We could have designed the form to be more user friendly, by hiding the fields that are not used, but our objective was to focus on the use of the MQEI and keep the rest of the application simple.

Figure 19. Sample Application ITSO Form

We used the form to make the same application available for both Lotus Notes and Web users. The translation of forms to HTML pages is a standard feature of the Lotus Notes Domino server.

If you want to develop an application for Lotus Notes users only, you can use dialog boxes instead of a form. The boxes can be designed as easily as forms, when they are based on layout regions, and thus they can force the
Lotus Notes client user to input data before proceeding with other work. Be aware that there is no way of converting Lotus Notes dialog boxes to HTML.

5.1.2.2 Documents
All documents that are created with the ITSO Form are stored in the database. We used a view with all documents as a logging facility for all transactions.

The LotusScript code we implemented accesses the document, uses the MQEI classes to send the request data to the enterprise system, receives the reply, and updates the same document. Additionally we used a status flag specifying whether the document is processed or unprocessed. We implemented this status flag as a computed text field in the form.

The status flag is used to realize a kind of batch processing. If an error occurred when trying to connect to an enterprise system, the document with the request is saved with the flag unprocessed. We assume that the reason for such an error, is that the enterprise system or the network is down, or that the request is done by a mobile user who is not connected to the network. These stored request documents can then be processed later. An agent therefore selects documents that have the unprocessed flag, communicates the request data in the form to the enterprise system, and updates the document with the reply data.

5.1.2.3 Agents
In our sample application the MQEI classes are used only in agents. Depending on the access type (Web user or Lotus Notes client) and on the MQEI service type selected, a different agent is started when the Submit button is clicked. The different access types are discussed in 5.1.3, “Implementation of Lotus Notes Client Access (Two Tier Model)” on page 71 and 5.1.5, “Implementation of Web User Access (Three Tier Model)” on page 80. The impact of the different service types is described in 5.2, “CICS DPL Direct Service Sample Implementation” on page 88 through 5.6, “IMS via MQSeries Service Implementation” on page 117.

We developed the following agents:

- For the Lotus Notes client
  - Client agent (CICS DPL ECI)
  - Client agent (CICS 3270 EPI)
  - Client agent (Native MQSeries)
  - Client agent (CICS DPL via MQSeries)
  - Client agent (IMS via MQSeries)

- For the Web client
- Web agent (CICS DPL ECI)
- Web agent (CICS 3270 EPI)
- Web agent (Native MQSeries)
- Web agent (CICS DPL via MQSeries)
- Web agent (IMS via MQSeries)

  - For scheduling
    - Scheduled agent (Native MQSeries)

5.1.3 Implementation of Lotus Notes Client Access (Two Tier Model)

Figure 20 on page 72 highlights the sequence of operations when a Lotus Notes client accesses the enterprise system through MQEI.
We describe the sequence of operation below. The steps that we implemented are highlighted.

For a complete list of Lotus Notes agents, see Appendix B, “Lotus Notes Agents” on page 127.

1. Creating a new document

We defined an action button in the main view of our database. The button uses the following formula to create a new document using the request form:
If the user clicks on the action button, a "user interface document" is created (an object of the LotusScript class, NotesUIDocument). This is the object that is presented on the Lotus Notes user interface through the form. It is not yet stored in the database (see 4.1.3.3, "Web User" on page 50).

2. **User enters request data in the form and submits the request**

We defined a submit button in the form that uses the following formula to save the document:

```
@Command([FileSave])
```

We saved the document because we implemented the agent that processes the request such that we read the request data from the back-end document (an object of the NotesDocument class) and not from the front-end document (NotesUIDocument class). We used the back-end document so that we could use the agent, with little enhancement, as a scheduled agent as well (see 5.1.4, "Using Scheduled Agents for a Three Tier Model" on page 78).

If in your application you prefer to access the user interface document, you do not have to save the document.

3. **Starting the agent**

The formula underlying the click event of the submit button starts the agent that processes the request. Because we used different agents for the different MQEI Services, but just one form, we implemented the following @IF statement to run a specific agent for each service selection:

```
@If(
    ServiceSelection="Native MQSeries";
    @Command([ToolsRunMacro];"Client Agent (Native MQSeries)");
    ServiceSelection="CICS DPL ECI";
    @Command([ToolsRunMacro];"Client Agent (CICS DPL ECI)");
    ServiceSelection="CICS 3270 EPI";
    @Command([ToolsRunMacro];"Client Agent (CICS 3270 EPI)");
    ServiceSelection="CICS DPL via MQSeries";
    @Command([ToolsRunMacro];"Client Agent (CICS DPL via MQSeries)");
    NULL)
```
Agent runs locally

Because the command that starts the agent is executed on the client, the agent itself runs on the client as well, even if the database where it is stored, is located on the server. Therefore the agent code is transmitted from the server to the run-time storage of the client.

Creating the MQEI objects

The first step in our agent is to create the MQEI objects of EIService, EIMessage, EISendOptions, and EIReceiveOptions classes.

    '************************************************
    '** Message, Service and file name of sample text.
    '************************************************
Const DBMESSAGE = "VSAMSERVCOMMAREA" ' Name of Message and
Const DBSERVICE = "CICSECI" ' Name of Database Service as
defined in Definition Database

    .......

    '************************************************
    '** Define Enterprise Integrator objects.
    '************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MyMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

Set MySession = New EISession
If MySession.ReasonCode <> EIRC_NONE ' Check for errors
    Set MyService = MySession.CreateService(DBSERVICE)
On Event EIError From MyService Call EIServiceErrorHandler
    ....
Set MyMessage = MySession.CreateMessage(DBMESSAGE)
On Event EIError From MyMessage Call EIMessageErrorHandler
    ....
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EIReceiveOptionsErrorHandler
    ....
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorHandler
    ....

Retrieving Service and Message definitions
When an object of the EIService class is created, the MQEI run-time component reads the service definition document in the Definition database to set the EIService Attributes. The message and field definitions are fetched when an EIMessage object is created.

- Retrieving user ID and password for the enterprise system logon.

When an object of the EIService class is created, the MQEI run-time component reads the MQEI Security definition of the user who runs the code (the equivalent property is the UserName property of the NotesSession class). The entries in that MQEI Security definition are used as EIService properties.

- Connecting to the enterprise system

The next step in the agent is the call of the Connect method of the EIService object. The MQEI component establishes a connection with the MQ or CICS server on the enterprise system, using the CICS or MQSeries client (or server) on the Lotus Notes client:

```vbscript
Call MyService.Connect() ' Connect to enterprise services

' Check for warnings
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
    Case EIRC_CONNECTED:......
```

- Mapping user input in Lotus Notes document to request message

The agent accesses the request document, reads the user input, and sets the field values in the EIMessage object. To access the Lotus Notes document, the agent accesses the document context property of the session, using the following script:

```vbscript
Dim Session As New NotesSession
Dim Doc As NotesDocument

Set Doc = Session.DocumentContext

We implemented the reading of item values in the NotesDocument, using the following script:
MyMessage.SURNAME = Doc.GetItemValue("I_SURNAME")(0)
MyMessage.FIRST_NAME = Doc.GetItemValue("I_FIRST_NAME")(0)
MyMessage.ACCOUNT = Doc.GetItemValue("I_ACCOUNT")(0)

with MyMessage : an object of class EIMessage
  SURNAME, FIRSTNAME, ACCOUNT : fieldnames as defined in definition database
  I_SURNAME, I_FIRST_NAME, I_ACCOUNT : names of the items

To access the user interface document (NotesUIDocument) instead of the back-end document (NotesDocument) in your application, you have to use the CurrentDocument property of the NotesUIWorkspace class. To read the input values, you have to use the FieldGetText method of the NotesUIDocument.

7. Sending request to enterprise system

We used the following script to send the request to the enterprise system:

Call MyService.SendMessage (MyMessage, MySendOptions)

Calling the Send method of the EIService object causes the EIMessage object to be sent to the enterprise system. The MQEI component sends a message to the MQSeries or CICS client on the Lotus Notes client. How this is done depends on the MQEI Service type. The MQSeries or CICS client sends the data to the MQ or CICS server of the enterprise system.

8. Receiving reply from enterprise system

We used the following script to receive the reply from the enterprise system:

MyReceiveOptions.Identifier = MySendOptions.Identifier
  ' Only read messages that belong to us
Do
  ' Attempt to receive the message
  Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
  ....
Loop

Calling the Receive method of the EIService object fetches the reply from the enterprise system and puts it into the EIMessage object. The
MQEI component fetches a message from the local MQSeries or CICS client. How this is done depends on the MQEI service type. The MQSeries or CICS client fetches the data from the MQ or CICS server of the enterprise system.

The EIReceiveOptions WaitInterval and WaitType properties of a EIReceiveOptions object can be used to make the call block or not:

- A blocking call waits for the specified wait interval and return a "no message available" reason code if no message arrives.
- A nonblocking call must be included in a loop until the message arrives. An escape method must also be included in the code to prevent an infinite loop.

Some service-specific considerations are pointed out in the appropriate sections of this chapter. For more detail refer to the MQEI documentation.

Mapping reply from enterprise system to Lotus Notes document

The output values of the received EIMessage object are read and the values are stored in the document, using the following script:

```plaintext
Doc.I_SURNAME=Trim(MyMessage.SURNAME)
Doc.I_FIRST_NAME=Trim(MyMessage.FIRST_NAME)
Doc.I_ACCOUNT=Trim(MyMessage.ACCOUNT)
```

with MyMessage : the received message, object of class EIMessage
SURNAMEn-FIRSTNAMEnACCOUNT : fieldnames as defined in definition database
I_SURNAMEnFIRST_NAMEnACCOUNT : names of the document items

It is always necessary to save the NotesDocument object (using the Save method) to commit the changes of item values of a document.

If you access the NotesUIDocument instead of the NotesDocument object, you do not have to save it. To set field values with the received data, use the FieldSetText method of the NotesUIDocument.

Redisplaying user interface document

Until now the agent acted just on the NotesDocument on the server. To redisplay the user interface document on the Lotus Notes client, the agent has to call the Reload method of the NotesUIDocument.

```plaintext
Call Doc.Save(True, True) ' Save back-end document
Call Uidoc.reload ' Reload UI document
```

Updating the user interface document
The user interface document accesses the back-end document and uses the document item values to update its fields.

5.1.4 Using Scheduled Agents for a Three Tier Model

The agent that is used for the two tier implementation in a native Lotus Notes environment (see 5.1.3, “Implementation of Lotus Notes Client Access (Two Tier Model)” on page 71) can be easily enhanced to be used in a three tier model with batch processing. A three tier model is especially useful if you want to support mobile users, or if you have other reasons to implement batch processing (refer to section 4.1.3, “User Types” on page 46). You just have to enhance the agent with the following features:

- Instead of activating the agent by a click on a button, you can define the agent as a scheduled agent.
- Instead of processing the request in one document (the current), the agent does a loop over all unprocessed request documents.

First, you need to know the status of the documents to be able to select only the unprocessed documents. In our sample, a form contains a field with a status flag that has a default value of Unprocessed. After the request document is updated with the reply data, the flag is set to Processed.

To define the agent as a scheduled agent, using the agent design pane, you select when this agent should run and on which document.

Figure 21 on page 79 shows the definition of our scheduled agent using the Native MQSeries service.
In our example, we choose to run the agent on an hourly basis. Because the scheduled agent runs on the server, and not on the client, we use a three tier model.

We enhanced the agent to process only unprocessed documents by defining a selection criterion based on that status flag. This is shown on the Search Builder window (See Figure 21).

We accessed the collection of documents that result from the search by using the following code fragment in the LotusScript code of the agent:
Dim Session As New NotesSession  
Dim Dc As NotesDocumentCollection  
Set Dc = Session.CurrentDatabase.UnprocessedDocuments  

Instead of processing one request document as the agent in the two tier implementation does, this scheduled agent has to do a loop over all elements of the document collection. To be more efficient, the loop should just include the reading of a document, the sending of the request, the receipt of the reply, and the updating of the document (steps 6 through 10 of 5.1.3, “Implementation of Lotus Notes Client Access (Two Tier Model)” on page 71). The connection should be established (and closed) outside the loop.

The loop can be implemented as in the following code fragment (step numbers refer to 5.1.3, “Implementation of Lotus Notes Client Access (Two Tier Model)” on page 71):

Dim Doc As NotesDocument  
Set Doc = Dc.FirstDocument  
Do Until Doc Is Nothing  
   ' Mapping user input to request message  
   step 6  
   ' Send request to enterprise system  
   step 7  
   ' Receive reply from enterprise system  
   step 8  
   ' Mapping reply to document  
   step 9  
   ' Save document  
   step 10  
   Set Doc = Dc.GetNextDocument(Doc)  
Loop

You must be aware that, in a scheduled agent, you cannot use any user interface object such as NotesUIDocument or NotesUIWorkspace. Even the declaration of such object variables causes errors on the Lotus Notes server.

5.1.5 Implementation of Web User Access (Three Tier Model)

Figure 22 on page 81 highlights the sequence of operations when a Web client access the enterprise system through MQEI.

80 Lotus Notes and the MQSeries Enterprise Integrator
We describe the sequence of operations below. The steps that we implemented are highlighted.

For a complete list of each Web agent, see Appendix C, “Web Agents” on page 197.

- Accessing the Domino server and browsing through the databases
  The HTTP server task provides standard features to display database lists, views, navigators, and other elements that a Web user can use interactively. Depending on the security setup of the Lotus Notes Domino server, Web users might have to identify themselves when accessing the server or a specific Lotus Notes element such as a
database. They do that by entering their Notes user names and password in a sign-on window (see 4.2, “Security” on page 52). By clicking on hotspots of the HTML pages, they send commands to the HTTP task of the Domino server. The HTTP task fetches the requested Lotus Notes element, converts it to an HTML page, and sends it to the user's browser.

1. Creating a new document

We defined an action button in the main view of our database. The button uses the following formula to create a new document using the request form:

```notes
@Command([Compose];"ITSO Form").
```

If the user clicks on the HTML representation of the action button, the HTTP task creates a Lotus Notes document.

2. Creating a Web form

The HTTP task creates a Web form that is an HTML representation of the document and the associated form. That Web form is sent to the user.

For additional background on creating documents in Web applications, refer to 4.1.3.3, “Web User” on page 50.

3. User enters request data in the Web form

The user enters data into the HTML page that represents the request form. When the user clicks on the Submit button, the Web form is sent to the HTTP server task and the associated Lotus Notes document is updated with the data entered.

For additional background on Submit buttons in Web forms refer to 4.1.3.3, “Web User” on page 50.

4. Starting the Web agent

We defined a computed text field with the name $$QuerySaveAgent in our form to define the agent that would run. When the Web form is submitted, the HTTP task looks for a field with that name and starts the agent that is named as the value of that field. Because we used different agents for the different MQEI services, but just one form, we had to set the field value (the agent name) dynamically according to the service selection. We implemented the following value formula for the $$QuerySaveAgent field:
ServiceSelection = "Native MQSeries";
    "Web Agent (Native MQSeries)";QSeries";
ServiceSelection = "CICS DPL ECI";
    "Web Agent (CICS DPL ECI)";Series";
ServiceSelection = "CICS 3270 EPI";
    "Web Agent (CICS 3270 EPI)";PL ECI";
ServiceSelection = "CICS DPL via MQSeries";
    "Web Agent (CICS DPL via MQSeries)";
NULL)

For additional background on starting Web agents, refer to 4.1.3.3, "Web User" on page 50.

### Creating the MQEI objects

The first step in our agent is to create the MQEI objects of EIService, EIMessage, EISendOptions, and EIReceiveOptions classes.

```vbscript
Const DBMESSAGE = "VSAMSERVCOMMAREA" ' Name of Message and
Const DBSERVICE = "CICSECI" ' Name of Database Service as
                          ' defined in Definition Database

Dim MySession As EISession
Dim MyService As EIService
Dim MyMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

Set MySession = New EISession
If MySession.ReasonCode <> EIRC_NONE ' Check for errors
    ....
Set MyService = MySession.CreateService(DBSERVICE)
On Event EIError From MyService Call EIServiceErrorEventHandler
    ....
Set MyMessage = MySession.CreateMessage(DBMESSAGE)
```

Chapter 5. ITSO MQEI Sample Application Implementation
On Event EIError From MyMessage Call EIMessageErrorEventHandler

.....
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler

.....
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler

Retrieving Service and Message definitions

When an object of the class EIService is created, the MQEI run-time component reads the service definition document in the Definition database to set the EIService attributes. The message and field definitions are fetched when an EIMessage object is created.

Retrieving user ID and password for the enterprise system logon.

When an object of the class EIService is created, the MQEI run-time component reads the MQEI Security definition of the user who ran the code (the equivalent property is the UserName property of the NotesSession class). The entries in that MQEI Security definition are used as EIService properties. When the EIService object is created in a Web agent or a scheduled agent, the name of the server is set in the UserName property. Consequently MQEI looks for an MQEI Security definition for the server name.

Connecting to the enterprise system

The next step in the agent is the call of the Connect method of the EIService object. The MQEI component establishes a connection with the MQ or CICS server on the enterprise system, using the CICS or MQSeries client (or server) on the Lotus Notes server:

Call MyService.Connect() ' Connect to enterprise services

' Check for warnings
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
        Case EIRC_CONNECTED:

Mapping user input in Lotus Notes document to request message

The agent accesses the request document, reads the user input, and sets the field values in the EIMessage object. To access that Lotus Notes document, the agent accesses the document context property of the session:
Dim Session As New NotesSession
Dim Doc As NotesDocument
Set Doc = Session.DocumentContext

We implemented the reading of item values in that NotesDocument using the following script:

.....
MyMessage.SURNAME = Doc.GetItemValue("I_SURNAME")[0]
MyMessage.FIRST_NAME = Doc.GetItemValue("I_FIRST_NAME")[0]
MyMessage.ACCOUNT = Doc.GetItemValue("I_ACCOUNT")[0]
.....

with MyMessage : an object of class EIMessage
  SURNAME, FIRSTNAME, ACCOUNT : fieldnames as defined in definition database
  I_SURNAME, I_FIRST_NAME, I_ACCOUNT : names of the items

.7 Sending request to enterprise system

Calling the Send method of the EIService object causes the EIMessage object to be sent to the enterprise system. The MQEI component sends a message to the local MQSeries or CICS client. How this is done depends on the MQEI service type. The MQSeries or CICS client sends the data to the MQ or CICS server of the enterprise system:

Call MyService.SendMessage(MyMessage, MySendOptions)

.8 Receiving reply from enterprise system

We used the following script to receive a reply from the enterprise system:

MyReceiveOptions.Identifier = MySendOptions.Identifier
' Only read messages that belong to us
Do
  ' Attempt to receive the message
  Call MyService.ReceiveMessage(MyMessage, MyReceiveOptions)
Loop

Calling the Receive method of the EIService object fetches the reply from the enterprise system and puts it into the EIMessage object. The
MQEI component fetches a message from the local MQSeries or CICS client. How this is done depends on the MQEI Service type. The MQSeries or CICS client fetches the data from the MQSeries or CICS server of the enterprise system.

The EIReceiveOptions WaitInterval and WaitType properties of a EIReceiveOptions object can be used to make the call block or not:

- A blocking call waits for the specified wait interval and return a "no message available" reason code if no message arrives.
- A nonblocking call must be included in a loop until the message arrives. An escape method must also be included in the code to prevent an infinite loop.

Some service-specific considerations are pointed out in the appropriate sections of this chapter. For more detail refer to the MQEI documentation.

Mapping reply from enterprise system to Lotus Notes document

The output values of the received EIMessage object are read, and the values are stored in the document, using the following script:

```plaintext
Doc.I_SURNAME=Trim(MyMessage.SURNAME)
Doc.I_FIRST_NAME=Trim(MyMessage.FIRST_NAME)
Doc.I_ACCOUNT=Trim(MyMessage.ACCOUNT)
```

with MyMessage : the received message, object of class EIMessage
SURNAME, FIRSTNAME, ACCOUNT : fieldnames as defined in definition database
I_SURNAME, I_FIRST_NAME, I_ACCOUNT : names of the document items

It is always necessary to save the NotesDocument object (using the Save method) to commit the changes of item values of a document.

Redisplaying updated document

Until now the agent acted just on the NotesDocument. To redisplay the document on the Web browser, the agent has to send a Print command to the HTTP task. We implemented the following Print command:

```plaintext
Print"[http://"+ServName.Common+Mydb.filename+"/All+Docs/"+Doc.NoteID
+"?OpenDocument]"
```

with ServName.Common : give the server name
Mydb.filename : give the name of the database
All+Docs : the name of the view
Doc.NoteID : give the document unique ID
The Document Unique ID is the unique 32-byte document identifier that can be queried by using the UniversalID property of a NotesDocument object.

Receiving the print command

The HTTP task fetches the document, converts it to HTML, and sends the HTML representation of the document back to the Web browser.

For additional background on output initiated by Web applications refer to 4.1.4, “Design Decision Table” on page 51.

5.1.6 Development Hints and Background on Web Applications

In this section, we provide some hints and background on how we developed the Web part of the ITSO MQEI sample application.

The steps referred to below are defined in 5.1.5, “Implementation of Web User Access (Three Tier Model)” on page 80.

Create documents in Web applications (see step 1)

If you want to create a Lotus Notes document using a form, you have to define an action button (in a view as we did or in a navigator). You cannot use the Lotus Notes menu selection Create, because it is part of the Lotus Notes user interface, and the HTTP task only converts Lotus Notes database elements to HTML. The @Commands that are supported by the HTTP task are limited to Document, View, and Database links (refer to Chapter 11, “Domino Application Developer’s Information” in the Lotus documentation entitled Working with Lotus Notes and the Internet).

Submit buttons in Web forms (see step 2)

By default, the HTTP task adds a submit button at the end of any HTML representation of a form. When this button is clicked, the HTTP task updates the Lotus Notes document in the background; it changes the item values (fields) and saves the document. If the form already contains a button (as our form does), the HTTP task uses this button as the submit button, instead of adding a new one. Therefore you can define the appearance, position, and labeling of a submit button yourself. Any code that is defined for the button click event is ignored by the HTTP task. Even if you define more than one button in a form, the HTTP task uses all buttons as submit buttons in the HTML representation.

Starting Web agents (see step 3)

Coming from the Web, the only way to run LotusScript code that uses the MQEI classes is in an agent started by the HTTP server task. The Domino
server supports three different system fields that can be used to start an agent: $$QuerySaveAgent, $$QueryOpenAgent, and $$Return. The first two fields are easy to handle because you just have to set the agent name as the field value. $$QuerySaveAgent runs an agent when the HTML form representation is submitted (and before the converted Lotus Notes document is saved). $$QueryOpenAgent runs an agent before the Lotus Notes documented is converted to HTML and sent to the Web user. $$Return is flexible, as can you define any URL link by setting the field value. An agent could be started with the following field value:

"[http://Server/Database.nsf/Agent>?OpenAgent]"

with Server : the server name  
    Database : the database name  
    Agent : the agent name

**Output initiated by Web applications** (see step 10)

You can use the Print command in a Web agent (that is, an agent that is called by the HTTP task) to display text on the Web browser or to display any HTML page, in our case the HTML representation of the updated Lotus Notes document.

**No user interface objects in Web agents**

You cannot use any user interface object (for example, of the NotesUIDocument or NotesUIWorkspace classes) in a Web agent. Even the declaration of such object variables causes errors on the Lotus Notes Domino server and affects the HTTP server task.

### 5.2 CICS DPL Direct Service Sample Implementation

Figure 23 on page 89 shows the process flow to link our Lotus Notes application to the CICS program (VSAMSERV) through the CICS DPL direct service.
The process is as follows:

1. MQEI initiates an ECI session with CICS passing a COMMAREA to the VSAMSERVER program.
2. VSAMSERVER executes its business logic based on the application function passed in the COMMAREA and interacts with the application file (TECHBASE).
3. VSAMSERVER terminates and returns an updated COMMAREA to the MQEI, which maps the COMMAREA fields into data fields within the Lotus Notes document.

5.2.1 MQEI

We created the following definitions in MQEI before we ran the CICS DPL direct service sample.

5.2.1.1 MQEI Definition Database

We had to define and build the service and the message definitions.

**Service Definition:** When you create an MQEI Service definition for a CICS DPL direct service, the key properties are:

- ConnectionManager, which gives the name of the CICS server with which a connection will be made. It is the name of the CICS server
defined in the server section of the CICSCLI.INI file of the CICS Client
(TCPIPOS2 in our sample)

- ServiceStep, which gives the name of the DPL program itself
  (VSAMSERV in our sample).

Figure 24 shows our service definition.

Figure 24. CICS DPL Direct Service Definition

**Message Definition:** The MQEI Message definition for the CICS DPL direct
service corresponds to the CICS COMMAREA passed as a parameter to DPL
program VSAMSERV by the EXEC CICS LINK command. Figure 25 on
page 91 shows our message definition.
5.2.1.2 MQEI Security Database

We created an MQEI Security definition for each user with a valid user ID and password (authenticator in MQEI terminology) to use our back-end system called TCPIPOS2 (see Figure 17 on page 67).

5.2.1.3 Security

If security is being used on the CICS server, there are two options for performing authentication of users:

- Specify the CICS user ID and password through the /u and /p options of the CICSCLI command when the CICS client is started
- Let the MQEI pass the CICS user ID and password automatically to the CICS ECI on each SendMessage call

For the first option, CICSCLI /c=servername /u=userid /p=password must be issued when the CICS client is started. The CICS server uses the user ID and password so specified. No other authentication is necessary and the MQEI Security database is not required. This option cannot be used if you are connected directly to a CICS server with a built-in client.

For the second option, the CICS user ID and password are obtained from the EIService UserId and Authenticator properties and passed to the CICS ECI on each SendMessage call. If authentication fails, or if the CICS user ID does not have sufficient authority to access resources used by the DPL
program, an MQEI reason code of EIRC_SECURITY_FAILURE is returned by the ReceiveMessage call.

The MQEI can use the MQEI Service definition SystemName property to read an MQEI Security definition for the Lotus Notes user from the MQEI Security database. This definition should contain the CICS user ID and password for the Lotus Notes user; it is used to set the EIService Userld and Authenticator properties. Alternatively the LotusScript program can prompt the user for this information and set the EIService Userld and Authenticator itself.

For more information about security in CICS DPL direct service, refer to the IBM MQSeries Enterprise Integrator for Lotus Notes User Guide online.

5.2.2 LotusScript

The LotusScript application connects to CICS and carries out, according to the service requested in the Lotus Notes document, the unit of work through calls to the ECI. The benefits and drawbacks in using this connection method to CICS are:

- **Benefits**
  - Coding is not required in LotusScript to handle CICS security; MQEI handles security automatically.
  - The ECI does not require screen scraping.
  - Communication between LotusScript and CICS is simplified with the transfer of only one message (COMMAREA) between environments.

- **Drawbacks**
  - Existing CICS 3270 applications may require modification.

Figure 26 on page 93 shows the steps that the LotusScript application must carry out.
The steps are:

1. Lotus Notes displays the document to collect information to be used to carry out the operation with the enterprise services.
2. Select the service you want to be performed, that is, Add, Read, Update, or Delete.
3. Enter the credit information into the document.
4. Select the CICS DPL ECI access method to the enterprise services.
5. When you click the Submit button, the LotusScript application sends a service request, using a COMMAREA, to the back-end program.
6. CICS returns confirmation data or error messages, using the same COMMAREA, to the LotusScript application.
7. LotusScript updates the Lotus Notes database with the data from the confirmation or error map.

5.2.3 CICS

We created the following CICS definitions on the OS/2 server before we ran the CICS DPL direct service sample. Knowledge of CICS is required to
create the definitions. The CICS CEDA transaction is used to create the definitions.

5.2.3.1 Define Programs
All CICS programs must be defined to CICS. Program definitions are put in the CICS Processing Program Table (PPT).

We defined our CICS program, VSAMSERV, which contains all of the logic for handling reads, adds, updates, and deletes to the application file (see Figure 61 on page 308 for the PPT entry.).

5.2.3.2 Define Files
If the CICS program uses VSAM files, they must be defined to CICS. File definitions are put in the CICS File Control Table (FCT). Tables controlled by DBMSs are not defined to CICS.

We created two FCT entries for our VSAM file:
- TECHBASE, the base VSAM file (see Figure 62 on page 309)
- TECHALT, the alternate index (see Figure 63 on page 309)

5.3 CICS 3270 Direct Service Implementation

Figure 27 on page 95 shows the process flow to link our Lotus Notes application to the CICS 3270 transaction (TECH) using the CICS 3270 direct service.
The process is as follows:

1. MQEI initiates an EPI session with CICS that includes the interaction between the MQEI and CICS to:
   - Sign on to CICS
   - Start the TECH transaction
   - Select an Add, Read, Update, or Delete application function

2. The TECH transaction invokes program TECHPROG.

3. TECHPROG links to program VSAMSERV, passing a CICS COMMAREA.

4. VSAMSERV executes its business logic on the basis of the application function selected and interacts with the application file.

5. VSAMSERV returns to TECHPROG with an updated COMMAREA.

6. TECHPROG sends the map, terminates, and returns control to CICS.

7. The CICS screen information is displayed on the CICS client session and then mapped by the MQEI into data fields within the Lotus Notes document.
5.3.1 MQEI

We performed the following steps before we ran the CICS 3270 direct service sample.

5.3.1.1 MQEI Definition Database

We had to define and build the service and the message definitions.

**Service Definition:** When you create an MQEI Service definition for a CICS 3270 direct service, the key properties are:

- **ConnectionManager**, which gives the name of the CICS server with which the connection will be made. It is the name of the CICS server defined in the server section of the CICSSCL.INI file of the CICS Client (TCP/IP OS/2 in our sample).

- **OutboundConnection**, which gives the name of the CICS model terminal to use as a template for the terminal the MQEI LSX is emulating. In our case, because we did not provide any value and the ModelTerm parameter was not defined in the CICSSCL.INI file of the CICS Client, we used the default model terminal for our CICS OS/2 server.

- **ServiceStep**, which gives the transid of the CICS transaction itself (TECH in our sample).

Figure 28 shows our service definition.

![Figure 28. CICS 3270 Direct Service Definition](image-url)
**Message Definition:** The MQEI Message definition for the CICS 3270 direct service corresponds to either the start data entered along with a CICS transid on the CICS command line or the data in a CICS BMS map.

Figure 29 on page 97 shows our message definitions.

---

<table>
<thead>
<tr>
<th>Message Code</th>
<th>Description</th>
<th>Data Type</th>
<th>Length</th>
<th>Fixed</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>T000</td>
<td>CICS BMS map T001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T001</td>
<td>SessionOverMsg</td>
<td>Session Over Message</td>
<td>STRING (FIXED)</td>
<td>18</td>
<td>N/A</td>
</tr>
<tr>
<td>T002</td>
<td>OPTION</td>
<td>T002 field 1</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>MESS1</td>
<td>T002 field 2</td>
<td>STRING (FIXED)</td>
<td>79</td>
<td>N/A</td>
</tr>
<tr>
<td>T003</td>
<td>SUP2</td>
<td>T002 field 3</td>
<td>STRING (FIXED)</td>
<td>74</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ACC2</td>
<td>T003 field 1</td>
<td>STRING (FIXED)</td>
<td>15</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>MESS2</td>
<td>T003 field 2</td>
<td>STRING (FIXED)</td>
<td>5</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>SUP3</td>
<td>T003 field 3</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ACC3</td>
<td>T003 field 4</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ADD3</td>
<td>T003 field 5</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ADD32</td>
<td>T003 field 6</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>MESS3</td>
<td>T003 field 7</td>
<td>STRING (FIXED)</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>T004</td>
<td>SUR4</td>
<td>T004 field 1</td>
<td>STRING (FIXED)</td>
<td>15</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ADD4</td>
<td>T004 field 2</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ADD34</td>
<td>T004 field 3</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ADD34</td>
<td>T004 field 4</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ADD34</td>
<td>T004 field 5</td>
<td>STRING (FIXED)</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>MESS4</td>
<td>T004 field 6</td>
<td>STRING (FIXED)</td>
<td>75</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 29 (Part 1 of 2). CICS 3270 Direct Service Messages Definition
5.3.1.2 MQEI Security Database
We used the same MQEI Security definitions as in the CICS DPL direct service because the back-end server was the same (TCPIPOS2; see Figure 17 on page 67).

5.3.1.3 Security
If the CICS server runs in a secure environment, authentication must be explicitly performed by the LotusScript program running the CICS CESN transaction (or your local equivalent). It is not possible for the MQEI to do this automatically. Refer to the Client Agent (CICS 3270 EPI) sample code for an example of how to do this, and for examples of an MQEI Service definition for the CESN transaction and an MQEI Message definition for the CESN BMS map.
The MQEI can use the MQEI Service definition SystemName property to read an MQEI Security definition for the Lotus Notes user from the MQEI Security database. This definition should contain the CICS user ID and password for the Lotus Notes user; it is used to set the EIService Userld and Authenticator properties. Alternatively the LotusScript program can prompt the user for this information and set EIService Userld and Authenticator itself.

It is the responsibility of the LotusScript program to set the user ID and authenticator fields of the EIMessage representing the CESN BMS map. The user ID field may be set by the program copying the EIService Userld property. The MQEI automatically sets the authenticator field from the EIService Authenticator property during SendMessage if it is defined with a data type of SYSTEM_AUTHENTICATOR. If the authenticator field is defined with a data type of AUTHENTICATOR, the program explicitly sets it.

For more information about security in CICS 3270 direct service, refer to the IBM MQSeries Enterprise Integrator for Lotus Notes User Guide online.

5.3.2 LotusScript

The LotusScript application mimics what a user would do in front of a 3270 terminal. This programming technique is called screen scraping. The application carries on a conversation with the enterprise system, tricking it into think that it is communicating with a 3270 terminal. The benefits and drawbacks in using this connection method to CICS are:

- **Benefits**
  - Modification to the CICS code is not required.

- **Drawbacks**
  - To develop an application using the CICS 3270 direct service, the developer must have a thorough understanding of the CICS transaction.
  - All messages must be sent and received in the correct order.
  - The CICS 3270 direct service conversation is very complicated.

Figure 30 on page 100 shows the steps the LotusScript application must carry out.
The steps are:

1. Lotus Notes displays the document to collect information to be used to carry out the operation with the enterprise services.

2. Select the service you want to be performed, that is, Add, Read, Update, or Delete.

3. Enter the credit information into the document.

4. Select the CICS 3270 EPI access method to the enterprise services.

5. When you click the Submit button, the LotusScript application send a clear screen to CICS and initiates the CESN transaction.

6. CICS returns a clear screen and an empty signon map to the LotusScript application.
7. The LotusScript application fills in the blank fields in the empty signon map with your user ID and password obtained from the MQEI Security database.

8. The LotusScript application sends back the populate map to CICS.

9. CICS returns a clear screen and a confirmation or error map to the LotusScript application.

10. The LotusScript application sends a clear screen to CICS and initiates the TECH transaction.

11. CICS returns a clear screen and the TECH menu map to the LotusScript application.

12. The LotusScript application enters into the TECH menu map the selection equivalent to your service selection on the Lotus Notes document.

13. The LotusScript application sends back the populate map to CICS.

14. CICS returns a clear screen and an empty data input map to the LotusScript application.

15. The LotusScript application fills in the fields on the data input map with the information you entered in the Lotus Notes document.

16. The LotusScript application sends back the populate map to CICS.

17. CICS returns a clear screen and a confirmation or error map to the LotusScript application.

18. The LotusScript application updates Lotus Notes with the data from the confirmation or error map.

19. The LotusScript application ends the TECH transaction.

20. CICS returns a clear screen and an end of the TECH transaction map.

21. The LotusScript application sends a clear screen to leave CICS in a ready state.

5.3.3 CICS

We created the following CICS definitions on the OS/2 server before we ran the CICS 3270 direct service sample. A knowledge of CICS is required to create the definitions. The CICS CEDA transaction is used to create the definitions.

5.3.3.1 Define Programs

All CICS programs must be defined to CICS. Program definitions are put in the CICS PPT.
We used the same PPT definition for the VSAMSERV program as we used in the CICS DPL direct service implementation (see 5.2.3.1, “Define Programs” on page 94).

We also defined the TECHPROG program, which contains all of the logic for handling CICS map input through mapset TECHMS (see Figure 60 on page 308 for the PPT entry).

5.3.3.2 Define Transactions
You must define all transactions you are using. A transaction definition defines the initial program that is executed for the transaction. If you are using CICS security instead of an external security package, you must also define security for the transaction. Transaction definitions are put in the CICS Program Control Table (PCT).

We defined transaction TECH, which executes TECHPROG (see Figure 56 on page 306 for the PCT entry).

5.3.3.3 Define Files
If the CICS program uses VSAM files, they must be defined to CICS. File definitions are put in the CICS FCT. Tables controlled by DBMSs usually are not defined to CICS.

We used the FCT entries that we created for the CICS DPL direct service implementation (see 5.2.3.2, “Define Files” on page 94).

5.4 Native MQSeries Service Implementation
Figure 31 on page 103 shows the process flow to link our Lotus Notes application to the CICS program (VSAMSERV) using the Native MQSeries service.
Figure 31. Native MQSeries Service Process Flow

The process is as follows:

1. MQEI executes an MQPUT to put a message to MQSeries queue EI.INPUTQUEUE.

2. The server connection channel, SYSTEM.DEF.SVRCONN, is required for communication between the MQSeries client and the MQSeries server. The MQPUT request is processed by the MQSeries server, and the message is placed in queue EI.INPUTQUEUE.

3. Queue EI.INPUTQUEUE has triggering set, so MQSeries looks up the associated Process definition to find out which process to start, in this case, CICS transaction MQ1. MQSeries then writes a trigger message to the initiation queue defined for queue EI.INPUTQUEUE, which is SYSTEM.CICS.INITIATION.QUEUE.
The CICS trigger monitor task, CKT1, which constantly polls the SYSTEM.CICS.INITIATION.QUEUE looking for trigger messages, reads the trigger message.

CKT1 starts transaction MQ1.
MQ1 invokes program AMQSCIC1 which does an MQGET to read a message from the EI.INPUTQUEUE message queue.
AMQSCIC1 links to program VSAMSERV, passing a COMMAREA.
VSAMSERV executes its business logic on the basis of the application function passed in the COMMAREA and interacts with the application file.
VSAMSERV terminates, and an updated COMMAREA is passed back to the AMQSCIC1 program.
AMQSCIC1 does an MQPUT to place the reply message to the ReplyToQueue defined in the original message, in this case, the EI.OUTPUTQUEUE message queue.
The MQEI issues an MQGET to get a message from the EI.OUTPUTQUEUE message queue.
The message is returned to the MQEI, which maps the message fields into data fields within the Lotus Notes document.

5.4.1 MQEI

We performed the following steps before we ran the Native MQSeries service sample.

5.4.1.1 MQEI Definition Database
We had to define and build the service and the message definitions.

Service Definition: When you create an MQEI Service definition for an Native MQSeries service, the key properties are:

- ConnectionManager, which gives the name of the MQSeries queue manager with which connection will be made (EITEST in our sample).
- OutboundConnection, which gives the name of the MQSeries queue used to send messages to the MQSeries-enabled application (EI.INPUTQUEUE in our sample).
- InboundConnection, which gives the name of the MQSeries queue used to receive messages from the MQSeries-enabled application (EI.OUTPUTQUEUE in our sample).

Figure 24 on page 90 shows our service definition.
**Message Definition:** The MQEI Message definition for the Native MQSeries service corresponds to the application data part of an MQSeries message. In our case, we created one message definition to be used by the two EIMessage objects for sending and receiving data. Because we connect to the same CICS program used in the CICS DPL direct service, we used the same message definition (see Figure 25 on page 91).

5.4.1.2 MQEI Security Database

We used the same MQEI Security definitions as in the CICS DPL direct service because the back-end server was the same (TCP/IPOS2; see Figure 17 on page 67).

5.4.1.3 Security

Security here relates to any authentication performed by the MQSeries application to which messages are sent, and to any security checking performed by the queue manager. If the MQSeries application is using its own security, it is assumed that any user ID and password are passed to it either in the message header or as part of the application message data.

The MQEI can use the MQEI Service definition SystemName property to read an MQEI Security definition for the Lotus Notes user from the MQEI Security database. This definition should contain the MQSeries application user ID and password for the Notes user; it is used to set the EIService UserId and Authenticator properties. Alternatively the LotusScript program can prompt...
the user for this information and set the EIService UserId and Authenticator itself.

It is the responsibility of the LotusScript program to set any user ID and authenticator fields of the EIMessage representing the MQSeries message. A User ID field may be set by the program copying the EIService UserId property. An authenticator field is set automatically by the MQEI from the EIService Authenticator property during SendMessage if it is defined with a data type of SYSTEM_AUTHENTICATOR. If the authenticator field is defined with a data type of AUTHENTICATOR, it is explicitly set by the program.

For more information about security in Native MQSeries service, refer to the IBM MQSeries Enterprise Integrator for Lotus Notes User Guide online.

5.4.2 LotusScript

The agents using the Native MQSeries service differ from those using the CICS DPL direct service and CICS 3270 direct service because they handle delayed receipt of replies. In contrast to a direct connection to a CICS system, which is always synchronous, the connection to an enterprise system through MQSeries is asynchronous. Thus you cannot be sure that you will get an immediate reply after sending a request. Therefore we enhanced our sample application with the support of a third request state, in addition to Processed and Unprocessed: No MQ reply found, which indicates that a request has been sent but a reply message has not been received.

We enhanced the form of our sample application, so that if the request is in the No MQSeries reply found state, the Submit button and service selection are hidden, and instead a Retry button and a status description are displayed (see Figure 33 on page 107).
When the **Retry** button is clicked, the same agent is started that is started by a **Submit** button. Therefore we enhanced the agent to handle this request state: Instead of always doing a send and a receive, the agent reads the status field in the document and does a send only when the document is not in the **No MQSeries reply found** state. If the document is in the **No MQSeries reply found** state, the agent skips the code part sending the message and just does a receive.

To receive the right reply message (delayed or not), we set the Identifier property of the **EIReceiveOptions** object with the unique MQ message identifier of the request message to which the reply is related. This identifier can be queried by reading the Identifier property of the **EISendOptions** object after the request is sent. This identifier is stored in a hidden field in the document.
5.4.3 MQSeries

Apart from the MQSeries default and system objects, we created the following MQSeries definitions on the OS/2 server before we ran the Native MQSeries service sample. A knowledge of MQSeries is required to set up the message queuing environment. Refer to the MQSeries product documentation for the platform you are using to determine how to create the following definitions.

5.4.3.1 Queue Manager

A queue manager is that part of an MQSeries product that provides the messaging and queuing services to application programs, through the MQI program calls.

In our sample, the Lotus Notes server connects to a queue manager located on the OS/2 server.

We created queue manager EITEST, using the following OS/2 command:

crtmqm -q EITEST.

5.4.3.2 Queue Definitions

MQSeries uses the following types of queues:

- An initiation queue, which receives trigger messages

  A trigger message indicates that a trigger event has occurred. In our sample, a message has been put onto application queue EI.INPUTQUEUE. Messages are read from the initiation queue by a trigger monitor application (in our sample, the CICS trigger monitor task, CKT1), which then starts the appropriate application to process the message (in our sample, the CICS transaction, MQ1).

  We defined CICS initiation queue SYSTEM.CICS.INITIATION.QUEUE (see Figure 66 on page 313).

- Message queues, which are used to receive messages from applications

  We defined an application input queue, EI.INPUTQUEUE, which allows CICS program AMQSCIC1 invoked by transaction MQ1 to read messages from the Lotus Notes application (see Figure 68 on page 314).

---

5 MQSeries default and system objects are created with the MQSC command file, AMQSCOMA.TST.
We defined an application output queue, EI.OUTPUTQUEUE, which allows the same program to put messages to the Lotus Notes application (see Figure 69 on page 314).

5.4.3.3 Process Definitions
A process definition object defines an application to the MQSeries queue manager. It is used for defining applications to be started by a trigger monitor.

In our sample, we created process definition TECHPROC, which causes the triggering of CICS transaction MQ1 (see Figure 67 on page 313).

5.4.3.4 Channel Definitions
An MQI channel is a logical communication link that connects an MQSeries client to a queue manager on a server machine.

To allow the communication between our Lotus Notes server running the MQSeries client code and the OS/2 server running MQSeries server, we defined a server-connection MQI channel, SYSTEM.DEF.SVRCONN (see Figure 65 on page 313). This channel is referred to in the system environment variable MQSERVER, set in OS/2 in the CONFIG.SYS file and in Windows NT in System Properties under the Environment tab.

5.4.4 CICS
We created the following CICS definitions on the OS/2 server before we ran the Native MQSeries service sample. A knowledge of CICS is required to create the definitions. The CICS CEDA transaction is used to create the definitions.

5.4.4.1 Define Programs
We used the same PPT definition for the VSAMSERV program as we used in the CICS DPL direct service implementation (see 5.2.3.1, “Define Programs” on page 94).

We also defined these other CICS programs:
- AMQLTMC0, which is the CICS trigger monitor program supplied with OS/2 (see Figure 58 on page 307 for the PPT entry)
- AMQSCIC1, which is the CICS program that gets messages off the MQSeries input queue, links to program VSAMSERV to process the messages, and puts the result to the MQSeries output queue (see Figure 59 on page 307 for the PPT entry).
5.4.4.2 Define Transactions
You must define all transactions you are using. A transaction definition defines the initial program that will be executed for the transaction. If you are using CICS security instead of an external security package, you must also define security for the transaction. Transaction definitions are put in the CICS PCT.

We defined the following transactions:
- CKT1, which executes program AMQLTMC0 (see Figure 57 on page 306)
- MQ1, which executes program AMQSCIC1 (see Figure 55 on page 305)

5.4.4.3 Define Files
If the CICS program uses VSAM files, they must be defined to CICS. File definitions are put in the CICS FCT. Tables controlled by DBMSs usually are not defined to CICS.

We used the FCT entries that we created for the CICS DPL direct service implementation (see 5.2.3.2, “Define Files” on page 94).

5.5 CICS DPL via MQSeries Service Implementation
Figure 34 on page 111 shows the process flow to link our Lotus Notes application to the CICS program (VSAMSERV) using the CICS DPL via MQSeries service.
The process is as follows:

1. MQEI executes an MQPUT to put a message to MQSeries queue MQEI_DPL_BRIDGE.

2. The server connection channel, SYSTEM.DEF.SVRCONN, is required for communication between the MQSeries client and the MQSeries server. The MQPUT request is processed by the MQSeries server.

3. Queue MQEI_DPL_BRIDGE is a remote queue, so the message is placed on the transmission queue defined for remote queue MQEI_DPL_BRIDGE.

4. Transmission of the message to remote Queue Manager CQS1 is initiated.
Communication between the queue managers is handled by the sender channel on OS/2, KRYPTON_MVS, and the receiver channel on MVS, KRYPTON_MVS.

The message is placed on local queue MQEI_DPL_BRIDGE on MVS.

The CICS DPL Bridge Monitor constantly polls the MQEI_DPL_BRIDGE looking for messages.

If a message is found that a new task needs to be started, the CICS DPL Bridge Monitor starts a CICS DPL Bridge task, CKBP, which invokes program CSQCBP00.

CSQCBP00 does an MQGET to read a message from the MQEI_DPL_BRIDGE message queue.

CSQCBP00 links to program VSAMSERV, passing a COMMAREA.

VSAMSERV executes its business logic on the basis of the application function passed in the COMMAREA and interacts with the application file.

VSAMSERV terminates, and the update COMMAREA is passed back to the CSQCBP00 program.

CSQCBP00 does an MQPUT to place the reply message to the ReplyToQueue defined in the original message, in this case, remote message queue EI.OUTPUTQUEUE. The bridge task ends.

The message is put on transmission queue EITEST.

Transmission of the message to remote queue manager EITEST is initiated.

Communication between queue managers is handled by the sender channel on MVS, MVS_KRYPTON, and the receiver channel on OS/2, MVS_KRYPTON.

The message is placed on the local queue on OS/2, EI.OUTPUTQUEUE.

The MQEI issues an MQGET to get a message from the EI.OUTPUTQUEUE message queue.

The message is returned to the MQEI, which maps the message fields into data fields within the Lotus Notes document.

5.5.1 MQEI

We performed the following steps before we ran the CICS DPL via MQSeries service sample.

Lotus Notes and the MQSeries Enterprise Integrator
5.5.1.1 MQEI Definition Database

We had to define and build the service and the message definitions.

**Service Definition:** When you create an MQEI Service definition for a CICS DPL via MQSeries service, the key properties are:

- **ConnectionManager**, which gives the name of the MQSeries queue manager with which connection will be made (EITEST in our sample)
- **OutboundConnection**, which gives the name of the MQSeries queue used to send messages to the MQSeries-CICS/ESA DPL bridge (MQEI_DPL_BRIDGE in our sample)
- **InboundConnection**, which gives the name of the MQSeries queue used to receive messages back from the application (EI.OUTPUTQUEUE in our sample). Both InboundConnection and OutboundConnection must be specified, and they cannot be the same queue.
- **ServiceStep**, which gives the name of the DPL program itself (VSAMSERV in our sample)

Figure 35 shows our service definition.

![MQCICSDPL](image)

**Figure 35. CICS DPL via MQSeries Service Definition**
**Message Definition:** For the CICS DPL via MQSeries service, we used the same MQEI Message definition as in the CICS DPL direct service because the same back-end program (VSAMSERV) was used. This definition corresponds to the CICS COMMAREA passed as a parameter to DPL program VSAMSERV by the EXEC CICS LINK command (see Figure 25 on page 91).

5.5.1.2 **MQEI Security Database**
No security is implemented in our sample, so the system name in our service definition is blank.

5.5.1.3 **Security**
The MQEI uses the MQEI Service definition SystemName property to read an MQEI Security definition for the Lotus Notes user. This definition should contain the CICS user ID and password for the Lotus Notes user; it is used to set the EIService UserId and Authenticator properties. Alternatively the LotusScript program can prompt the user for this information and set the EIService UserId and Authenticator itself.

If the CICS server runs in a secure environment, the CICS user ID and password must be passed by the MQEI to the MQSeries-CICS/ESA DPL bridge on each SendMessage call so that CICS can authenticate the Lotus Notes user. The EIService UserId property must contain a user ID when the EIService Connect call is made. The MQMD UserIdentifier is set from the EIService UserId, the MQCIH Authenticator is set from the EIService Authenticator, the MQMD AccountingToken is set to MQACT_NONE, and the MQMD ApplIdentityData is set to blanks.

For more information about security in CICS DPL via MQSeries service, refer to the IBM MQSeries Enterprise Integrator for Lotus Notes User Guide online. Refer also to the MQSeries - CICS/ESA DPL Bridge User's Guide for additional information about MQSeries-CICS/ESA DPL bridge security.

5.5.2 **LotusScript**
The LotusScript code in the agent for the CICS DPL via MQSeries service is identical to that of the Native MQSeries service, except that another Service definition is referred to when the EIService Object is created (see 5.4.2, “LotusScript” on page 106 for a description of the agent.).

5.5.3 **MQSeries on OS/2 and MVS/ESA**
We created the following MQSeries definitions on the OS/2 server and the MVS/ESA server before we ran the CICS DPL via MQSeries service sample. A knowledge of MQSeries is required to set up the message queuing
environment. Refer to the MQSeries product documentation for the platform you are using to determine how to create the following definitions.

5.5.3.1 Queue Manager
In OS/2, we used the same queue manager that we defined for the Native MQSeries service sample (see 5.4.3, “MQSeries” on page 108).

In MVS, we created a queue manager called CSQ1.

5.5.3.2 Queue Definitions
In OS/2, we used the following queues:

- A remote application input queue, MQEI_DPL_BRIDGE, which identifies a queue belonging to the MVS/ESA queue manager (see Figure 70 on page 314)
- A transmission queue, CSQ1, which temporarily stores the messages that are destined for the remote MVS/ESA queue manager, CSQ1 (see Figure 72 on page 315)
- A local application output queue, EI.OUTPUTQUEUE, which we had defined for the Native MQSeries service (see Figure 69 on page 314)

In MVS/ESA, we created the following queues:

- A local application input queue, MQEI_DPL_BRIDGE (see Figure 79 on page 317)\(^6\)
- A transmission queue, EITEST, which temporarily stores the messages that are destined for the remote OS/2 queue manager (see Figure 76 on page 316)
- A remote application output queue, EI.OUTPUTQUEUE, which identifies the queue belonging to the OS/2 queue manager (see Figure 75 on page 316)

5.5.3.3 Channel Definitions
In OS/2, we used the server connection channel, SYSTEM.DEF.SVRCONN, which we defined for the Native MQSeries service sample (see 5.4.3.4, “Channel Definitions” on page 109), to facilitate the communication between our Lotus Notes server running the MQSeries client code and the OS/2 server running the MQSeries server.

---

\(^6\) You can use the JCL in the CSQ4CKBM file of the SupportPac to define the local queue named SYSTEM.CICS.BRIDGE.QUEUE.
To facilitate the communication between the OS/2 server and the MVS/ESA machine, both of which were running the MQSeries server, we defined the following channels:

- **OS/2**
  - Sender channel KRYPTON_MVS (see Figure 73 on page 315)
  - Receiver channel MVS_KRYPTON (see Figure 74 on page 315)

- **MVS/ESA**
  - Sender channel MVS_KRYPTON (see Figure 77 on page 316)
  - Receiver channel KRYPTON_MVS (see Figure 78 on page 316)

### 5.5.4 CICS/ESA

We created the following CICS definitions on the MVS/ESA server before we ran the CICS DPL via MQSeries service sample. A knowledge of CICS is required to create the definitions. The CICS CEDA transaction is used to create the definitions.

#### 5.5.4.1 Define Programs

All CICS programs must be defined to CICS/ESA in the CICS System Definition (CSD) file.

We defined the following CICS program:

- **VSAMSERV**, which contains all of the logic for handling reads, adds, updates, and deletes to the VSAM file (see Figure 61 on page 308 for the PPT entry)

We used resource definition utility DFHCSDUP, using CSQ4CKBC as input, to define these bridge programs and transactions (see Figure 64 on page 310 for the CSQ4CKBC file):

- CSQCBR00, the bridge monitor program
- CSQCBP00, the bridge program
- CSQCBP10, the bridge abend handler program
- CSQCBTXE, the bridge error messages program in lowercase English

#### 5.5.4.2 Define Transactions

You must define all transactions you are using in the CICS CSD.

We defined the following transactions:

- **CKBR**, the MVS/ESA CICS DPL bridge monitor transaction, which executes program CSQCBR00
CKBP, the MVS/ESA CICS DPL bridge transaction, which executes program CSQCGBP00

5.5.4.3 Define Files
If the CICS program uses VSAM files, they must be defined to CICS. File definitions are put in the CICS CSD. Tables controlled by DBMSs usually are not defined to CICS.

5.6 IMS via MQSeries Service Implementation

Figure 36 shows the process flow to link our Lotus Notes application to the IMS transaction (IVTNO) using the IMS via MQSeries service.

Figure 36. IMS via MQSeries Service Process Flow

The process is as follows:
MQEI executes an MQPUT to put a message to MQSeries queue IMSOTMA.QUEUE.

Server connection channel SYSTEM.DEF.SVRCONN is required for communication between the MQSeries client and the MQSeries server. The MQPUT request is processed by the MQSeries server.

Queue IMSOTMA.QUEUE is a remote queue, so the message is placed on the transmission queue defined for remote queue IMSOTMA.

Transmission of the message to remote queue manager CSQ1 is initiated.

Communication between queue managers is handled by the sender channel on OS/2, KRYPTON_MVS, and the receiver channel on MVS, KRYPTON_MVS.

The message is placed on local queue IMSOTMA on MVS.

The IMS bridge code running in address space CSQ1 issues an MQGET of the message.

The IMS bridge uses the OTMA interface to pass the information to IMS.

IMS schedules transaction IVTNO in an MPR.

IMS transaction IVTNO issues a GU call to obtain the information passed to it

The application processes the message and issues an ISRT call to send the reply back to IMS.

IMS sends the reply back to the IMS bridge, using the OTMA interface.

The IMS bridge uses the MQPUT1 call to place the message to the ReplyToQueue defined in the original message, in this case remote message queue EI.OUTPUTQUEUE.

The message is put on transmission queue EITEST.

Transmission of the message to remote queue manager EITEST is initiated.

Communications between queue managers is handled by the sender channel on MVS, MVS_KRYPTON, and the receiver channel on OS/2, MVS_KRYPTON.

The message is placed on the local queue on OS/2 EI.OUTPUTQUEUE.

The MQEI issues an MQGET to get a message from the EI.OUTPUTQUEUE message queue.
The message is returned to the MQEI, which maps the message fields into data fields within the Lotus Notes document.

5.6.1 MQEI

We performed the following steps before we ran the IMS via MQSeries service sample.

5.6.1.1 MQEI Definition Database

We had to define and build the service and the message definitions.

**Service Definition:** When you create an MQEI Service definition for an IMS via MQSeries service, the key properties are:

- ConnectionManager, which gives the name of the MQSeries queue manager with which connection will be made (EITEST in our sample)
- OutboundConnection, which gives the name of the MQSeries queue used to send messages to the MQSeries-IMS bridge (IMSOTMA.QUEUE in our sample)
- InboundConnection, which gives the name of the MQSeries queue used to receive messages back from the application (EI.OUTPUTQUEUE in our sample). Both InboundConnection and OutboundConnection must be specified, and they cannot be the same queue.
- ServiceStep, which gives the name of the IMS transaction (IVTNO in our sample)

Figure 37 on page 120 shows our service definition.
**Figure 37. IMS via MQSeries Service Definition**

**Message Definition:** The MQEI Message definition for the IMS via MQSeries service corresponds to the IMS message data passed to the IMS transaction when it issues a GU to the IOPCB or sent as output by the IMS transaction when it issues an ISRT to the IOPCB. Figure 38 on page 121 shows our message definition.
5.6.1.2 MQEI Security Database

No security is implemented in our sample, therefore the system name in our service definition is blank.

5.6.1.3 Security

The MQEI can use the MQEI Service definition SystemName property to read an MQEI Security definition for the Lotus Notes user. This definition should contain the IMS user ID and password for the Lotus Notes user; it is used to set the EIService UserId and Authenticator properties. Alternatively the LotusScript program can prompt the user for this information and set the EIService UserId and Authenticator itself.

If security is implemented in the IMS server, the MQEI must pass the IMS user ID and password to the MQSeries-IMS bridge on each SendMessage call so that IMS can authenticate the Lotus Notes user. The EIService UserId property must contain a user ID when the EIService Connect call is made.

If authentication fails or if the IMS user ID does not have sufficient authority to access resources used by the IMS transaction, the ReceiveMessage call returns an MQEI reason code of EIRC_SECURITY_FAILURE.
For more information about security in IMS via MQSeries service, refer to the IBM MQSeries Enterprise Integrator for Lotus Notes User Guide online.

5.6.2 LotusScript
The LotusScript code in the agent for the IMS via MQSeries service is identical to that of the Native MQSeries service, except that another Service definition is referred to when the EIService Object is created (see 5.4.2, “LotusScript” on page 106 for a description of the agent).

5.6.3 MQSeries on OS/2 and MVS/ESA
We performed the following MQSeries definitions on the OS/2 server and the MVS/ESA server before we ran the IMS via MQSeries service sample. A knowledge of MQSeries is required to set up the message queuing environment. Refer to the MQSeries product documentation for the platform you are using to determine how to create the following definitions.

5.6.3.1 Queue Manager
In OS/2, we used the same queue manager that we defined for the Native MQSeries service sample (see 5.4.3, “MQSeries” on page 108).

In MVS, we used the same queue manager that we defined for the CICS DPL via MQSeries service sample (see 5.5.3, “MQSeries on OS/2 and MVS/ESA” on page 114).

5.6.3.2 Queue Definitions
In OS/2, we used the following queues:

- A remote application input queue, IMSOTMA.QUEUE, which identifies a queue belonging to the MVS/ESA queue manager (see Figure 71 on page 315)
  - A transmission queue, CSQ1, which temporarily stores the messages that are destined for the remote MVS/ESA queue manager (see Figure 72 on page 315)
- A local application output queue, EI.OUTPUTQUEUE (see Figure 69 on page 314.)

In MVS/ESA, we created the following queues:

- A local application input queue, IMSOTMA.QUEUE (see Figure 80 on page 317)
- A remote application output queue, EI.OUTPUTQUEUE, which identifies a queue belonging to the OS/2 queue manager (see Figure 75 on page 316)
A transmission queue, EITEST, which temporarily stores the messages that are destined for the remote OS/2 queue manager (see Figure 76 on page 316).

5.6.3.3 Channel Definitions

In OS/2, we used server connection channel SYSTEM.DEF.SVRCONN, which we defined for the Native MQSeries service sample (see 5.4.3.4, "Channel Definitions" on page 109), to facilitate the communication between our Lotus Notes server running the MQSeries client code and the OS/2 server running the MQSeries server.

To facilitate the communication between the OS/2 server and the MVS/ESA machine, both of which were running the MQSeries server, we defined the following channels:

- OS/2
  - Sender channel KRYPTON_MVS (see Figure 73 on page 315)
  - Receiver channel MVS_KRYPTON (see Figure 74 on page 315)
- MVS/ESA
  - Sender channel MVS_KRYPTON (see Figure 77 on page 316)
  - Receiver channel KRYPTON_MVS (see Figure 78 on page 316)

5.6.4 IMS

Our sample IMS application is part of the INSTALL/IVP facility of IMS. INSTALL is a facility for installing a new release of IMS. IVP is a facility for verifying the installation of IMS using an extensive sample IMS system.
Appendix A. MQEI Installation Considerations and Requirements

Before you install the MQEI, ensure that you have the correct level of operating system and the correct level of Lotus Notes to run the MQEI.

The MQEI runs only in either the Lotus Notes 4.5 client or the Lotus Domino Server 4.5 environment.

In addition to Lotus Notes, you need the MQSeries client or server installed to enable you to connect to your MQSeries enterprise systems, or you need to have the CICS client installed, to enable you to connect directly to your CICS enterprise systems.

A.1 MQEI MQSeries Requirements

If you use the following MQEI services to connect to an enterprise service:

- Native MQSeries service
- IMS via MQSeries service
- CICS DPL via MQSeries service

the MQEI requires access to either an MQSeries client or an MQSeries server (from the following list) that is installed in the same environment:

- MQSeries client on AIX
- MQSeries client on HP-UX
- MQSeries client on OS/2
- MQSeries client on Sun Solaris
- MQSeries client on Windows 3.1
- MQSeries client on Windows 95
- MQSeries client on Windows NT
- MQSeries for AIX Version 2.2.1 (for the server)
- MQSeries for HP-UX Version 2.2.1 (for the server)
- MQSeries for OS/2 Version 2.0.1 (for the server)
- MQSeries for Sun Solaris Version 2.2 (for the server)
- MQSeries for Windows NT Version 2.0 (for the server)
- MQSeries for Windows Version 2.0 (for the leaf-node server)\(^7\)

---

\(^7\) MQSeries for Windows Version 2.0 is different from the other MQSeries family of products. It is designed to run on a workstation with Microsoft Windows 3.1, Windows for Workgroups, Windows 95, or WIN-OS/2. It provides significantly more function than an MQSeries client, by including a subset of the queue manager functions. To differentiate this type of queue manager from that provided by other MQSeries products, the MQSeries for Windows queue manager is known as a leaf-node queue manager.
If you choose to use one of the MQSeries client environments, connect it to an MQSeries server that supports it. The server can be any MQSeries server that supports the MQSeries client; it does not have to be a server capable of running Notes.

A.2 MQEI CICS Requirements

If you are connecting directly to CICS enterprise systems, using one of the following MQEI services:

- CICS DPL direct service
- CICS 3270 direct service

the MQEI requires access to either a CICS client or a CICS server with a built-in client (from the following list) that is installed in the same environment:

- IBM CICS Client for OS/2
- IBM CICS Client for Windows
- IBM CICS Client for Windows 95
- IBM CICS Client for Windows NT
- IBM CICS for OS/2 (for the server with a built-in client)
- IBM Transaction Server for OS/2 Warp (for the server with a built-in client)
- IBM CICS for Windows NT Version 2 (for the server with a built-in client)

If you choose to use one of the CICS client environments, connect it to a CICS server that supports it. The server does not have to be capable of running Lotus Notes.

A.3 Disk Space Requirements

The additional disk space requirements for the MQEI LSX executable code depend on the platform you are running:

- AIX - 8.5 MB
- HP-UX - 8.5 MB
- OS/2 - 8.5 MB
- Sun Solaris - 9.0 MB
- Windows 3.1 - 7.5 MB
- Windows 95 - 8.5 MB
- Windows NT - 8.5 MB
- Windows for Workgroups - 7.5 MB
- WIN-OS/2 - 7.5 MB
Appendix B. Lotus Notes Agents

In this appendix we list the code used for the Lotus Notes agents.

---

**Downloading the Application**

A Lotus Notes database containing all of the LotusScript agents listed in this appendix is available for download on the Redbooks Web site at [www.redbooks.ibm.com](http://www.redbooks.ibm.com)

Click on the Downloads button, select the SG242217 directory, and select the file to download.

---

**B.1 Lotus Notes Client Agent for CICS DPL Direct Service**

```lsl
'Client Agent (CICS DPL ECI):
******************************************************************************
'* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator*
'* ITSO Sample Application -
'* Lotus Notes Client Agent for the MQEI CICS DPL Direct Service
******************************************************************************
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselsx "elsx"

******************************************************************************
'* Define Enterprise Integrator objects.
******************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MyMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

******************************************************************************
'* Notes variables for form manipulation.
```

*Figure 39 (Part 1 of 9).* Lotus Notes Client Agent for CICS DPL Direct Service
Dim Session As NotesSession
Dim Workspace As NotesUIWorkspace
Dim Doc As NotesDocument
Dim Uidoc As NotesUIDocument

'*****************************************************************************
' Dim variables
'*****************************************************************************

' Dim StrText As String
' Dim ErrorText As String

'*****************************************************************************
'* Wait time in milliseconds for a reply.
'*****************************************************************************
Const FIVE_SECONDS = 5000 ' 5 seconds

'*****************************************************************************
'* Message, Service and file name of sample text.
'*****************************************************************************
Const DBMESSAGE = "VSAMSERVCOMMAREA" ' Name of Message as defined in Definition Database
Const DBSERVICE = "CICSEC" ' Name of Database Service as defined in Definition Database

'*****************************************************************************
'* Translatable text strings used throughout sample.
'*****************************************************************************
Const TITLE_TEXT = "CICS ECI sample"
Const LOTUSSCRIPT_ERROR_TEXT = "LotusScript error!"
Const EILSX_ERROR_TEXT = "EILSX error!"
Const ERROR_NUMBER_TEXT = "Error number"
Const ERROR_DESCRIPTION_TEXT = "Error description"
Const LINE_NUMBER_TEXT = "Line number"
Const EIQC_CONNECTED_TEXT = "An attempt was made to connect to a service when already connected." _
  + "The connect is ignored and the program continues."
Const EIQC_DISCONNECTED_TEXT = "An attempt was made to disconnect to a service when already disconnected. The disconnect is ignored and the program continues."
Const EIQC_NO_MESSAGE_AVAILABLE_TEXT = "No message available - Retry operation?"
Const EIQC_FIELD_TRUNCATED_TEXT = "A length of a string value of an EIMessage field exceeds the length" _
  + "specified on the Definition Database. Correct the LotusScript to make sure that the string" _
  + "value being passed as the EIMessage field value does not exceed the predefined length."
Const EIQC_WRONG_MESSAGE_TEXT = "A message was received that does not match the EIMessage object" _
  + "passed as a parameter on the EIService ReceiveMessage call."
Const UNRECOGNISED_TEXT = "Unrecognised warning message:"
Const EISERVICE_WARNING_TEXT = "Warning message received establishing MQService. ReasonCode: "
Const EISSESSION_ERROR_TEXT = "An error has been reported from the EISession Class. ReasonCode: "
Const EIMESSAGE_ERROR_TEXT = "An error has been reported from the EIMessage Class. ReasonCode: "
Const EISESSION_ERROR_TEXT = "An error has been reported from the EIService Class. ReasonCode: "
Const EISO_ERROR_TEXT = "An error has been reported from the EISendOptions Class. ReasonCode: "
Const EIRO_ERROR_TEXT = "An error has been reported from the EIReceiveOptions Class. ReasonCode: "
Const SAMPLE_END_TEXT = "Sample completed."
Const PRIMARYSYSTEM = "Primary System Error Code"

Figure 39 (Part 2 of 9). Lotus Notes Client Agent for CICS DPL Direct Service
Const SECONDARYSYSTEM = "Secondary System Error Code"
Const SYSTEMERRORTEXT = "Description"
Const INIT_TEXT = "Initialising..."
Const SEND_TEXT = "Sending..."
Const RECEIVE_TEXT = "Receiving..."
Const CONNECT_TEXT = "Connecting..."
Const DISCONNECT_TEXT = "Disconnecting..."
Const GO_TEXT = "Press 'Go' button to start CICS program."

'******************************************************************************
' * Definition of StatusFlag Values
'******************************************************************************
Const SF_PROCESSED="Processed"

Sub Initialize
'*******************************************************************************************
'* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator*
'* ITSO Sample Application -
'* Lotus Notes Client Agent for the MQEI CICS DPL Direct Service
'******************************************************************************
' This sample illustrates how a Lotus Script application can execute
' a CICS DPL ECI program.
'*
'* The CICS program is called VSAMSERV and is a credit information sample
' application.
'*
'* This agent is activated when the "CICS DPL ECI" Service Selection radio
'* button is selected and the "Submit" button is clicked.
'*
'* NOTE: When you see a number enclosed in '<<' and '>>' characters, (eg. <<1>>) 
'* it is referring to a step described in the
' chapter 5.1.1 Implementation of the Lotus Notes Client Access (Two Tier Model)
'* in the ITSO Redbook:
'******************************************************************************
'* Initialize the EILSX objects. <<4>>
'******************************************************************************
    Print INIT_TEXT

    '******************************************************************************
    ' Set up the error handlers (Note: These are not event handlers).
    '******************************************************************************
    On Error Goto GeneralErrorHandler  ' Handle general LotusScript errors
    On Error EILSX_ERROR Goto MQEIErrorHandler  ' EILSX errors

'* Initialize the EISession object <<4>>
'******************************************************************************
    Set MySession = New EISession

' Check for errors

Figure 39 (Part 3 of 9). Lotus Notes Client Agent for CICS DPL Direct Service
If MySession.ReasonCode <> EIRC_NONE Then
    Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
    Call MySession.ClearErrorCodes()
End If

*****************************************************************************
* Create an EI Service
*****************************************************************************
Set MyService = MySession.CreateService(DBSERVICE)
On Event EIError From MyService Call EIIServiceErrorEventHandler

*****************************************************************************
* Create an EI Message
*****************************************************************************
Set MyMessage = MySession.CreateMessage(DBMESSAGE)
On Event EIError From MyMessage Call EIMessageErrorEventHandler

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyMessage.CompletionCode = EICC_WARNING Then
    Select Case MyMessage.ReasonCode
        Case EIRC_FIELD_TRUNCATED:
            MsgBox EIRC_FIELD_TRUNCATED_TEXT, MB_OK, TITLE_TEXT
        Case Else:
            MessageBox UNRECOGNISED_TEXT & " & Cstr(MyMessage.ReasonCode), MB_OK, TITLE_TEXT
    End Select
    Call MyMessage.ClearErrorCodes()
End If

*****************************************************************************
* Create EI Send Options.
*****************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler

MySendOptions.MessageType = EIMT_REQUEST ' Messages are requests

*****************************************************************************
* Create EI Receive Options.
*****************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler

MyReceiveOptions.WaitInterval = FIVE_SECONDS ' Set wait interval of 5 seconds
MyReceiveOptions.WaitType = EIWT_WAIT ' Set wait type to use specified timeout

*****************************************************************************
* Set document variables
*****************************************************************************
Set Session = New NotesSession

Figure 39 (Part 4 of 9). Lotus Notes Client Agent for CICS DPL Direct Service
Set Doc = Session.DocumentContext
Set Workspace = New NotesUINotesWorkspace
Set Uidoc = Workspace.CurrentDocument

'*****************************************************************************
'** Connect to enterprise services <<5>>
'*****************************************************************************
Print CONNECT_TEXT
Call MyService.Connect() ' Connect to enterprise services

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
    Case EIRC_CONNECTED: ' Already connected (probably from previous run)
        messagebox EIRC_CONNECTED_TEXT, MB_OK, TITLE_TEXT
    Case Else :
        messagebox EISERVICE_WARNING_TEXT & " " & Cstr(MyService.ReasonCode) & 
        ",", MB_OK, TITLE_TEXT
    End Select
    Call MyService.ClearErrorCodes()
End If

'*****************************************************************************
'** Setup and send message to CICS <<6>>
'*****************************************************************************
MySendOptions.UnitOfWork = EIUOW_ONLY ' EC01 operates in its own unit of work
MyMessage.SERVER_STATE = Doc.GetItemValue("I_SERVER_STATE")
MyMessage.SERVER_KEY_SIZE = Cint(Doc.GetItemValue("I_SERVER_KEY_SIZE")
MyMessage.SURNAME = ...
MyMessage.BALANCE = Cint(Doc.GetItemValue("I_BALANCE")

'*****************************************************************************
'** Send the message <<7>>
'*****************************************************************************
Print SEND_TEXT
Call MyService.SendMessage (MyMessage, MySendOptions)

'*****************************************************************************
'** Get reply from CICS. <<8>>
'*****************************************************************************
MyReceiveOptions.Identifier = MySendOptions.Identifier 'Only read messages that belong to us
Call MyService.ReceiveMessage(MyMessage,MyReceiveOptions)

'*****************************************************************************
'** Update backend document with receive message <<9>>

Figure 39 (Part 5 of 9). Lotus Notes Client Agent for CICS DPL Direct Service
Doc.I_SERVER_STATE=Trim(MyMessage.SERVER_STATE)
Doc.I_SERVER_KEY_SIZE=Cstr(MyMessage.SERVER_KEY_SIZE)
Doc.I_SURNAME=Trim(MyMessage.SURNAME)
Doc.I_FIRST_NAME=Trim(MyMessage.FIRST_NAME)
Doc.I_ACCOUNT=Trim(MyMessage.ACCOUNT)
Doc.I_ADDRESS=Trim(MyMessage.ADDRESS)
Doc.I_CITY_STATE =Trim(MyMessage.CITY_STATE)
Doc.I_POSTAL_CODE =Trim(MyMessage.POSTAL_CODE)
Doc.I_BALANCE=Cstr(MyMessage.BALANCE)
Doc.O_RESPONSES=Trim(MyMessage.RESPONSES)
Doc.StatusFlag=SF_PROCESSED  ' Set status flag

Call Doc.Save(True, True)  ' Save backend document
Call Uidoc.reload  ' Reload UI document
Call CleanUp  ' Disconnect from enterprise services

 Exit Sub

'****************************************************************************
'* General LotusScript Error Handler
****************************************************************************
GeneralErrorHandler:
Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
Call CleanUp
Exit Sub

'****************************************************************************
'* Handle EILSX errors.
****************************************************************************
MQEIErrorHandler:
Call GlobalErrors(EILSX_ERROR_TEXT)
Call CleanUp
Exit Sub
End Sub

Sub EIServiceErrorEventHandler (thisService As EIService, ReasonCode As Long)
'* Error Handler for the EIService Class. At this point we choose not to take
' any corrective action, the error is simply displayed in a message box.
'* Alternatively you could write a CASE statement to handle your chosen
'* errors. Once handled the Error Codes can be cleared, allowing the
'* main program to continue.
****************************************************************************
Select Case ReasonCode
' Case EIRC_???:
'   ...Error Handle Process...
   thisService.clearErrorCodes

Figure 39 (Part 6 of 9). Lotus Notes Client Agent for CICS DPL Direct Service

132  Lotus Notes and the MQSeries Enterprise Integrator
Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long)
'*******************************************************************************************
' Error Handler for the EIService Class. At this point we choose not to take
' any corrective action, the error is simply displayed in a message box.
' *
' * Alternatively you could write a CASE statement to handle your chosen
' * errors. Once handled the Error Codes can be cleared, allowing the
' * main program to continue.
'*******************************************************************************
Select Case ReasonCode
' Case EIRC_??? : ...Error Handle Process...
' thisSendOptions.clearErrorCodes
' Case EIRC_??? : ...Error Handle Process...
' thisSendOptions.clearErrorCodes
' ...
' Case Else :
' MessageBox EIS0_ERROR_TEXT & " " & Cstr(ReasonCode) & ". Message Name: " & _
' thisSendOptions.Name & ",", MB_OK, TITLE_TEXT
End Select
End Sub

Sub EReceiveOptionsErrorEventHandler (thisReceiveOptions As EIReceiveOptions, ReasonCode As Long)
'*******************************************************************************************
' Error Handler for the EIService Class. At this point we choose not to take
' any corrective action, the error is simply displayed in a message box.
' *
' * Alternatively you could write a CASE statement to handle your chosen
' * errors. Once handled the Error Codes can be cleared, allowing the
' * main program to continue.
'*******************************************************************************
Select Case ReasonCode
' Case EIRC_??? : ...Error Handle Process...
' thisReceiveOptions.clearErrorCodes
' Case EIRC_??? : ...Error Handle Process...
' thisReceiveOptions.clearErrorCodes
' ...
' Case Else :
' MessageBox EIS0_ERROR_TEXT & " " & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
End Select
End Sub

**Figure 39 (Part 7 of 9). Lotus Notes Client Agent for CICS DPL Direct Service**
Error Handler for the EIService Class. At this point we choose not to take any corrective action, the error is simply displayed in a message box. Alternatively you could write a CASE statement to handle your chosen errors. Once handled the Error Codes can be cleared, allowing the main program to continue.

*****************************************************************************
Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
    thisReceiveOptions.clearErrorCodes
  Case EIRC_??? :
    ...Error Handle Process...
    thisReceiveOptions.clearErrorCodes
  ...
Case Else :
  Messagebox EIRO_ERROR_TEXT & " " & Cstr(ReasonCode) & ",", MB_OK, TITLE_TEXT
End Select
End Sub
Sub GlobalErrors(CallerText As String)
*****************************************************************************
** Output error handler messages in a formatted message box.
*****************************************************************************
  Dim ErrorText As String
  Print CallerText
    ' Format the error text for the message box
    ErrorText = CallerText + Chr$(10)
    ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
    ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$
    ErrorText = ErrorText + Chr$(10) + LINE_NUMBER_TEXT + " = " + Str(Erl)
  ' Output the error to message box and print line
  MsgBox ErrorText, MB_OK, TITLE_TEXT
  Print ""
End Sub
Sub Cleanup
*****************************************************************************
** This function is called when the agent is finished.
*****************************************************************************
  On Error Resume Next ' Ignore any error as we need to shutdown regardless.
  "" Disconnect from MQSeries and delete the EI objects.
*****************************************************************************
  Print DISCONNECT_TEXT
  Call MyService.Disconnect()
  "" Delete the EI objects.

*****************************************************************************
Figure 39 (Part 8 of 9). Lotus Notes Client Agent for CICS DPL Direct Service

134 Lotus Notes and the MQSeries Enterprise Integrator
Sub EISessionErrorEventHandler(ThisSession As EISession, ReasonCode As Long)
'******************************************************************************
'  Error event handler for the EISession Class. At this point we choose not to take
'  any corrective action, an error message is simply displayed in a message box.
'  Alternatively you could write a CASE statement to handle your chosen
'  errors. Once handled the Error Codes can be cleared, allowing the
'  main program to continue.
'  Note: this subroutine is registered as an error event handler for EISession
'  events but is also called following a failure in EISession creation in Queryopen
'******************************************************************************
Dim StrText As String

Select Case ReasonCode
  Case EIRC_??? :  ' ...Error Handle Process...
  Case EIRC_??? :  ' ...Error Handle Process...
  ...  Case EIRC_SEC_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " " & Cstr(ReasonCode) & "." + Chr$(10)  
    StrText = StrText + SYSTEMERRORTEXT + " " + Cstr(ThisSession.SystemErrorText) +  
              + Chr$(10)  
    StrText = StrText + PRIMARYSYSTEM + " " + Cstr(ThisSession.PrimarySystemErrorCode)  
              + Chr$(10)  
    StrText = StrText + SECONDARYSYSTEM + " " + Cstr(ThisSession.SecondarySystemErrorCode)  
    Messagebox StrText, MB_OK, TITLE_TEXT  
Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)  
    StrText = StrText + PRIMARYSYSTEM + " " + Cstr(ThisSession.PrimarySystemErrorCode) +  
              + Chr$(10)  
    StrText = StrText + SECONDARYSYSTEM + " " + Cstr(ThisSession.SecondarySystemErrorCode)  
    Messagebox StrText, MB_OK, TITLE_TEXT  
Case Else :  
    Messagebox EISESSION_ERROR_TEXT + " " + Cstr(ThisSession.ReasonCode), MB_OK, TITLE_TEXT
End Select

'******************************************************************************
'  Do not clear the error code, as this will allow error EILSX_ERROR to be
'  caught by the generic error handler.
'******************************************************************************
End Sub

Figure 39 (Part 9 of 9). Lotus Notes Client Agent for CICS DPL Direct Service
**B.2 Lotus Notes Client Agent for CICS 3270 Direct Service**

*CICS Agent (CICS 3270 EPI)*:

******************************************************************************
* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator*
* ITSO Sample Application
* Lotus Notes Client Agent for the MQEI CICS 3270 EPI Service
******************************************************************************

Public Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselsx "eilsx"

******************************************************************************
* Define Enterprise Integrator objects.
******************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

******************************************************************************
* Define Enterprise Messages
******************************************************************************
Dim CESNMessage As EIMessage 'CESN Screen
Dim CL01Message As EIMessage 'Message for ClearScreen
Dim VAR1Message As EIMessage 'Variant Message for TECH
Dim MyMessage As EIMessage 'Generic Message
Dim T001Message As EIMessage 'Screen T001
Dim T002Message As EIMessage
Dim T003Message As EIMessage
Dim T004Message As EIMessage
Dim T005Message As EIMessage
Dim T006Message As EIMessage
Dim T007Message As EIMessage

******************************************************************************
* Notes variables for form manipulation.
******************************************************************************
Dim Session As NotesSession
Dim Workspace As NotesUIWorkspace
Dim Doc As NotesDocument
Dim Uidoc As NotesUIDocument

******************************************************************************

*Figure 40 (Part 1 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service*

136 Lotus Notes and the MQSeries Enterprise Integrator
Const SERVICENAME = "CICSEPICESN"
Dim ReceiveType As Long

Const TITLE_TEXT = "CICS 3270 Direct sample"
Const LOTUSSCRIPT_ERROR_TEXT = "LotusScript error!"
Const EILSX_ERROR_TEXT = "EILSX error!"
Const ERROR_NUMBER_TEXT = "Error number"
Const ERROR_DESCRIPTION_TEXT = "Error description"
Const LINE_NUMBER_TEXT = "Line number"
Const EIRC_CONNECTED_TEXT = "An attempt to connect to a service has been rejected because the service "_ + "is already connected. The connect is ignored and the program continues."
Const RECEIVE_FAILED_TEXT = "Unable to receive message, ending sample"
Const EIRC_NO_MESSAGE_AVAILABLE_TEXT = "No message available - Retry operation?"
Const EIRC_FIELD_TRUNCATED_TEXT = "The program has set a field in an EIMessage where the field is defined" _ + "to be shorter than the length of data supplied. Alternatively, the program has got the data from an" _ + "EIMessage field where the data in the field is too long."
Const EIRC_WRONG_MESSAGE_TEXT = "An EIMessage received does not match the EIMessage object passed" _ + "as the parameter on the EIService ReceiveMessage call."
Const UNRECOGNISED_TEXT = "Unrecognised warning message:"
Const EISERVICE_WARNING_TEXT = "Warning message received establishing EIService, ReasonCode: "
Const EISESSION_ERROR_TEXT = "An error has been reported from the EIsession Class, ReasonCode: "
Const EIMESSAGE_ERROR_TEXT = "An error has been reported from the EIMessage Class, ReasonCode: "
Const EIERROR_TEXT = "An error has been reported from the EIError Class, ReasonCode: "
Const EISO_ERROR_TEXT = "An error has been reported from the EIOptions Class, ReasonCode: "
Const PRIMARISYSTEM = "Primary System Error Code"
Const SECONDARYSYSTEM = "Secondary System Error Code"
Const SYSTEMERRORTEXT = "Description"
Const GET_REPLY_ERROR_TEXT = "An error has been reported from ReceiveMessage, ReasonCode:"
Const SAMPLE_ABEND_TEXT = "Sample terminated abnormally"
Const SYSTEM_INVALID = "A CICS system must be selected"
Const USERID_TOO_LONG = "Userid is a maximum of 8 characters"
Const USERID_REQUIRED = "Userid not stored in security database so must be entered"
Const PASSWORD_TOO_LONG = "Password is a maximum of 8 characters"
Const PASSWORD_REQUIRED = "Password not stored in security database so must be entered"
Const NEWPASSWORD_TOO_LONG = "New password is a maximum of 8 characters"
Const NEWPASSWORD_REQUIRED = "New password must be entered to change the password"
Const VERIFYPASSWORD_MISMATCH = "Verify Password is not the same as New Password. Please re-enter" _ + "New and Verify passwords"
Const INIT1_TEXT = "Initializing session..."
Const INIT2_TEXT = "Initializing general messages..."
Const GO_TEXT = "Ready to go..."
Const SYS1_TEXT = "Creating service..."
Const SYS2_TEXT = "Creating TECH message..."
Const SEND_CLEAR = "Sending CLEAR..."
Const SEND_F3 = "Sending F3..."

Figure 40 (Part 2 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
Const SEND_F12 = "Sending F12..."
Const SEND_TECH = "Sending TECH..."
Const SEND_CESN = "Sending CESN..."
Const SEND_TEXT = "Sending data..."
Const RECEIVE_TEXT = "Receiving..."
Const CONNECT_TEXT = "Connecting..."
Const DISCONNECT_TEXT = "Disconnecting..."
Const DELETE_TEXT = "Deleting..."

**********************************************************************************
* Definition of StatusFlag Values
**********************************************************************************
Const SF_PROCESSED="Processed"
Dim StrText As String
Dim Status As String
Dim Send As String
Sub GlobalErrors(CallerText As String)
' Output error handler messages in a formatted message box.
Dim ErrorText As String

Print CallerText
'
Format the error text for the message box
ErrorText = CallerText + Chr$(10)
ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$
'
Output the error to message box and print line
Messagebox ErrorText, MB_OK, TITLE_TEXT
Print ""
End Sub

Sub EIServiceErrorEventHandler(ThisService As EIService, ReasonCode As Long)
' Error event handler for the EIService Class. At this point we choose not to take
' any corrective action, an error message is simply displayed in a message box.
' Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
' main program to continue.

Select Case ReasonCode
Case EIRC_SERVICE_SYSERROR:
    StrText = EISERVICE_ERROR_TEXT + " + Cstr(ReasonCode) + Chr$(10)
    StrText = StrText + "Service Name : " + ThisService.Name + Chr$(10)
    StrText = StrText + "Primary System Error Text : " + Cstr(ThisService.SystemErrorText) + Chr$(10)
    StrText = StrText + "Primary System Error Code : " + Cstr(ThisService.PrimarySystemErrorCode) + Chr$(10)
Messagebox StrText, MB_OK, TITLE_TEXT
Case Else :
End Sub

Figure 40 (Part 3 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service

138 Lotus Notes and the MQSeries Enterprise Integrator
Sub EISessionErrorEventHandler(ThisSession As EISession, ReasonCode As Long)
  Dim StrText As String
  Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
  Case EIRC_??? :
    ...Error Handle Process...
  ... Case EIRC_SEC_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " " & Cstr(ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " " + Cstr(ThisSession.SystemErrorText) _
    + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " " + Cstr(ThisSession.PrimarySystemErrorCode) _
    + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " " + Cstr(ThisSession.SecondarySystemErrorCode)
    MessageBox StrText, MB_OK, TITLE_TEXT
  Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " " + Cstr(ThisSession.SecondarySystemErrorCode)
    MessageBox StrText, MB_OK, TITLE_TEXT
  Case Else :
    MessageBox EISESSION_ERROR_TEXT + " " + Cstr(ThisSession.ReasonCode), MB_OK, TITLE_TEXT
  End Select

  '**************************************************************
  ' Do not clear the error code, as this will allow error EILSX_ERROR to be
  ' caught by the generic error handler.
  '**************************************************************
End Sub

Figure 40 (Part 4 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
Sub Cleanup
*****************************************************************************
"* This function is called when the agent is finished.
"* Disconnect from MQSeries and delete the EI objects.
*****************************************************************************
On Error Resume Next ' Ignore any error as we need to shutdown regardless.
Print "CLEANUP"
*****************************************************************************
"* Disconnect from MQSeries.
*****************************************************************************
Call MyService.Disconnect()
*****************************************************************************
"* Delete the EI objects.
*****************************************************************************
Delete CESNMessage
Delete MS01Message
Delete T001Message
Delete T002Message
Delete T003Message
Delete T004Message
Delete T005Message
Delete T006Message
Delete T007Message
Delete MyReceiveOptions
Delete MySendOptions
Delete MyService
Delete MySession
Print "End of CleanUp"
End Sub

Sub Initialize
*****************************************************************************
"** SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator"
"** ITSO Sample Application -
"* Lotus Notes Client Agent for the MQEI CICS 3270 EPI Service
*****************************************************************************
"* This sample illustrates how a Lotus Script application can execute
"* a CICS 3270 EPI program.
"* The Lotus Notes application sends a 3270 datastream to CICS transaction TECH.
"* CICS transaction TECH invokes program TECHFROG.
"* Program TECHFROG calls program VSAMSERV with a COMMAREA.
"* CICS program VSAMSERV is a credit information sample application
"* This agent is activated when the "CICS 3270 EPI" Service Selection radio
"* button is selected and the "Submit" button is clicked.
"* NOTE: When you see a number enclosed in '<>' and '>>' characters, (eg. <<C>>) 
"* it is referring to a step described in the
"* chapter 5.1.3 Implementation of the Lotus Notes Client Access (Two Tier Model)
Figure 40 (Part 5 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service

140 Lotus Notes and the MQSeries Enterprise Integrator
On Error Goto GeneralErrorHandler ' Handle general LotusScript errors
On Error EILSX_ERROR Goto MQEIEHandlerer ' EILSX errors

Set Session = New NotesSession
Set Doc = Session.DocumentContext
Set Workspace = New NotesUIWorkspace
Set UIdoc = Workspace.CurrentDocument

Initialize the EISession object <<4>>

Set MySession = New EISession

' Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
    Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
    Call MySession.ClearErrorCodes()
End If

Create an EI Service
' <<a>> defined in the EI Definition database).  
Set MyService = MySession.CreateService(SERVICENAME)
On Event EIError From MyService Call EIServiceErrorEventHandler

Create the general EI Messages 
' <<a>> defined in the EI Definition database).  
Call CreateMyMessage(CESNMessage, "CESN")
Call CreateMyMessage(MLO1Message, "MLO1")
Call CreateMyMessage(VARIMessage, "VariantMsg")
Call CreateMyMessage(T001Message, "T001")
Call CreateMyMessage(T002Message, "T002")
Call CreateMyMessage(T003Message, "T003")
Call CreateMyMessage(T004Message, "T004")
Call CreateMyMessage(T005Message, "T005")
Call CreateMyMessage(T006Message, "T006")
Call CreateMyMessage(T007Message, "T007")

Create EI Send Options.

Figure 40 (Part 6 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler

"** Create EI Receive Options.**
***************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler

"** Connect to the enterprise service **<<5>>
***************************************************************
Call MyService.Connect() ' Connect to CICS

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
   Select Case MyService.ReasonCode
   Case EIRC_CONNECTED: ' Already connected (probably from previous run)
      Messagebox EIRC_CONNECTED_TEXT, MB_OK, TITLE_TEXT
   Case Else :
      Messagebox EISERVICE_WARNING_TEXT & " & Cstr(MySession. ReasonCode) & ", MB_OK, TITLE_TEXT
   End Select
   Call MyService.ClearErrorCodes()
End If

"* Start Transaction CESN **<<6>>
***************************************************************
Call TranCESN

"* If no Error then
"* Start Transaction TECH
***************************************************************
If Status <> "ERROR" Then
   Call TranTECH
End If

"* If no Error then
"* Stop the Conversation
***************************************************************
If Status <> "ERROR" Then
   Call EndConversation
End If

"* Update the UiDocument with the new informations **<<10>>
***************************************************************
Call Doc.Save(True, True) ' Save backend document

Figure 40 (Part 7 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
Call Uidoc.reload ' Reload UI document
Call CleanUp ' Disconnect from enterprise services and delete all MQEI objects
Exit Sub

****************************************************************************
'* General LotusScript Error Handler
*****************************************************************************

GeneralErrorHandler:
Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
Call Cleanup
Exit Sub

****************************************************************************
'* Handle EILSX errors.
*****************************************************************************

MQEILogger:
Call GlobalErrors(EILSX_ERROR_TEXT)
Call Cleanup
Exit Sub

Sub CreateMyMessage (MyMessage As EIMessage, MessageName As String)
'*********************************************************************************'* Create an EI Message (as defined in the MQEI Definition database)
'*********************************************************************************
Set MyMessage = MySession.CreateMessage(MessageName)
On Event EIError From MyMessage Call EIMessageErrorEventHandler

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyMessage.CompletionCode = EICC.WARNING Then
  Select Case MyMessage.ReasonCode
  Case EIRC_FIELD_TRUNCATED:
    MessageBox EIRC_FIELD_TRUNCATED_TEXT, MB_OK, TITLE_TEXT
  Case Else :
    MessageBox UNRECOGNISED_TEXT & " & Cstr(MyMessage.ReasonCode), MB_OK, TITLE_TEXT
  End Select
  Call MyMessage.ClearErrorCodes()
End If
End Sub

Sub EIMessageErrorEventHandler (ThisMessage As EIMessage, ReasonCode As Long)
'***********************************************************************************'* Error event handler for the EIMessage Class. At this point we choose not to take
'***********************************************************************************'* any corrective action, an error message is simply displayed in a message box.
'***********************************************************************************'* Alternatively you could write a CASE statement to handle your chosen
'***********************************************************************************'* errors. Once handled the Error Codes can be cleared, allowing the
'***********************************************************************************'* main program to continue.
'***********************************************************************************
Dim StringText As String
  Select Case ReasonCode
End Sub

Figure 40 (Part 8 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service

Appendix B. Lotus Notes Agents 143
Case EIRC_??? : ...Error Handle Process...
    Call ThisMessage.clearErrorCodes()
Case EIRC_??? : ...Error Handle Process...
    Call ThisMessage.clearErrorCodes()
...
Case Else :
    StringText = EIMESSAGE_ERROR_TEXT + "+" + Cstr(ReasonCode) + Chr$(10)
    StringText = StringText + "Message Name : " + ThisMessage.Name
    MessageBox StringText, MB_OK, TITLE_TEXT
End Select
******************************************************************************
** Do not clear the error code, as this will allow error EILSK_ERROR to be
** caught by the generic error handler.
******************************************************************************
End Sub
Sub EIRReceiveOptionsErrorEventHandler(ThisReceiveOptions As EIRReceiveOptions, ReasonCode As Long)
******************************************************************************
** Error event handler for the EIRReceiveOptions Class. At this point we choose not to take
** any corrective action, an error message is simply displayed in a message box.
**
** Alternatively you could write a CASE statement to handle your chosen
** errors. Once handled the Error Codes can be cleared, allowing the
** main program to continue.
******************************************************************************
Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
        Call ThisReceiveOptions.clearErrorCodes()
    Case EIRC_??? :
        ...Error Handle Process...
        Call ThisReceiveOptions.clearErrorCodes()
    ...
Case Else :
    MessageBox EIRO_ERROR_TEXT & "+" & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
End Select
******************************************************************************
** Do not clear the error code, as this will allow error EILSK_ERROR to be
** caught by the generic error handler.
******************************************************************************
End Sub
Sub EISendOptionsErrorEventHandler(ThisSendOptions As EISendOptions, ReasonCode As Long)
******************************************************************************
** Error event handler for the EISendOptions Class. At this point we choose not to take
** any corrective action, an error message is simply displayed in a message box.
**
** Alternatively you could write a CASE statement to handle your chosen
** errors. Once handled the Error Codes can be cleared, allowing the
** main program to continue.
******************************************************************************
Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...

Figure 40 (Part 9 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
Call ThisSendOptions.clearErrorCodes()

Case EIRC_???:
    ...Error Handle Process...
    Call ThisSendOptions.clearErrorCodes()
    ...

Case Else:
    MessageBox EISO_ERROR_TEXT & " " & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
End Select

******************************************************************************************
* Do not clear the error code, as this will allow error EILSX_ERROR to be          *
* caught by the generic error handler.                                           *
******************************************************************************************

Sub SendMessage(MyMessage As EIMessage, MyAttentionID As Long)
    '***********************************************************************************************
    '* Send A Message                                                            
    '***********************************************************************************************

    MySendOptions.AttentionID = MyAttentionID
    Call MyService.SendMessage (MyMessage, MySendOptions)
End Sub

Sub TranCESN
    '***********************************************************************************************
    '* NOTE: In the following event, numbers enclosed in '{' and '}' characters, (eg. {{1}})  
    '* refer to a step described in the  
    '* chapter 5.3.2 LotusScript  (CICS 3270 Direct Service Implementation)  
    '* in the ITSO Redbook:                                                         
    '* Handling of Transaction CESN is not complete in this sample as we do not cover:  
    '* - password change  
    '* - wrong userid entered  
    '* - ....  
    '***********************************************************************************************

    Sending a Clear (5)
    MySendOptions.AttentionID = EIA1_CLEAR
    Call MyService.SendMessage (Nothing, MySendOptions)

    Sending CESN Transid (5)
    MyService.ServiceStep = "CESN"
    MySendOptions.AttentionID = EIA1_ENTER
    Call MyService.SendMessage (Nothing, MySendOptions)

    Receiving Signon Map (6)
    Status = "FALSE"
    Print "TranCESN RECEIVE CESN"
    ReceiveType = EIRT_RECEIVE
    Call ReceiveMessage (CESNMessage, ReceiveType)

    If Status <> "ERROR" Then

    Figure 40 (Part 10 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
'Sending Userid and Password {8}
CESNMessage.UserID = MyService.UserID
Call SendMessage (CESNMessage, EIAI_ENTER)

Status = "FALSE"
ReceiveType = EIRT_RECEIVE
'Receiving Confirmation {9}
Call ReceiveMessage (ML01Message, ReceiveType)
If Instr(ML01Message.MessageText, "FAA2250I") = 1 Then Print "LOGON OK"
End If

Exit Sub
End Sub

Sub ReceiveMessage(MyMessage As EIMessage, MyReceiveType As Long)

Status = "FALSE"
MyReceiveOptions.ReceiveType = MyReceiveType
*****************************************************************************************************************************************************
'** Timeout is not supported by the EPI.
'** We wait indefinitely on the Receive until
'** we get the answer from the enterprise system
*****************************************************************************************************************************************************
MyReceiveOptions.WaitType = EIWT_WAIT
MyReceiveOptions.WaitInterval = EIWI_UNLIMITED
Do
  ' Receiving the message
  Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
  Select Case MyService.ReasonCode
  Case EIRC_NONE:
    Status = "OK"
  Case EIRC_NO_MESSAGE_AVAILABLE:
    Status = "FALSE"
  Case EIRC_WRONG_MESSAGE:
    '*****************************************************************************************************************************************************
    '** Message received into the buffer but doesn't match the fields
    '** defined in the MyMessage map so the
    '** EIRC_WRONG_MESSAGE warning is raised.
    '** We therefore set ReceiveType to EIRT_RETURN to attempt
    '** to return it against an alternative EIMessage definition, ML01Message
    '*****************************************************************************************************************************************************
    MyReceiveOptions.ReceiveType = EIRT_RETURN
    Call MyService.ReceiveMessage (ML01Message, MyReceiveOptions)
  Select Case MyService.ReasonCode
  Case EIRC_NONE:
    Status = "FALSE"
    MyReceiveOptions.ReceiveType = EIRT_RECEIVE
  Case Else
    Status = "ERROR"
  End Select
  End Select
Loop While Status = "FALSE"

Figure 40 (Part 11 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
Exit Sub

End Sub

Sub EndConversation

' Sending a PF12
MySendOptions.AttentionID = EIAI_F12
Call MyService.SendMessage (Nothing, MySendOptions)

Status = "FALSE"
Print "End of Transaction"
ReceiveType = EIRT_RECEIVE
' Receiving the Clear Screen
Call ReceiveMessage (ML01Message, ReceiveType)
' Receiving the TECH:Session Over message
Call ReceiveMessage (ML01Message, ReceiveType)
End Sub

Sub TranTECH
'*********************************************************************************************'* NOTE: In the following event, numbers enclosed in "[" and "]" characters, (eg. [1])
' refer to a step described in the
' chapter 5.3.2 LotusScript (CICS 3270 Direct Service Implementation)
' in the ITSO Redbook:
'*********************************************************************************************

Print "TranTECH SEND CLEAR"
' Sending a Clear (10)
MySendOptions.AttentionID = EIAI_CLEAR
Call MyService.SendMessage (Nothing, MySendOptions)

Print "TranTECH SEND TECH"
'Sending TECH Transid (10)
MyService.ServiceStep = "TECH"
MySendOptions.AttentionID = EIAI_ENTER
Call MyService.SendMessage (Nothing, MySendOptions)

Do
  ' Receiving maps (14,...)
  Call ReceiveVARI
  If send <> "END" Then  Call SendMessage(MyMessage,EIAI_ENTER)
Loop While Send <> "END"

Exit Sub

End Sub

Sub ReceiveVari

'*********************************************************************************************'*We are receiving message from the enterprise system but the format is not known.
'* We first receive it into a variant message and analyse it.
'* Each screen is identified by a screen identifier in row 1, column 69.
'*********************************************************************************************

Figure 40 (Part 12 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
Dim NumOption As String
Print "TranTECH RECEIVE VARI"
Status = "FALSE"
ReceiveType = EIRT_RECEIVE
Call ReceiveMessage (VARIMessage, ReceiveType)

ReceiveType = EIRT_RETURN
Select Case VARIMessage.MsgIdentifier
  '* Main Menu received
  '' We send back to CICS the selected option
  Case "DDW1MS.T001":
    Call ReceiveMessage(T001Message, ReceiveType)
    Select Case Doc.GetItemValue ("I_SERVER_STATE") (0)
    Case "2": NumOption = "1"
    Case "1": NumOption = "2"
    Case "3": NumOption = "3"
    Case "4": NumOption = "4"
    End Select
    If T001Message.MESS1 = "" Then
      T001Message.OPTION = NumOption
      Set MyMessage = T001Message
      Send = "YES"
    End If

  '* Show Account Details received
  '' We send back to CICS the entered surname or account number
  Case "DDW1MS.T002":
    Call ReceiveMessage(T002Message, ReceiveType)
    If T002Message.MESS2 = "" Then
      T002Message.SUR2 = Doc.GetItemValue("I_Surname") (0)
      T002Message.ACC2 = Doc.GetItemValue("I_Account") (0)
      Set MyMessage = T002Message
      Send = "YES"
    Else
      Doc.O_RESPONSES = T002Message.MESS2
      Send = "END"
    End If

  '* Transaction is ended
  Case "DDW1MS.T003":
    Call ReceiveMessage(T003Message, ReceiveType)
    If T003Message.MESS3 = "" Then
      Doc.I_SURNAME = Trim (T003Message.SUR3)
      Doc.I_FIRST_NAME = Trim (T003Message.FOR3)
      Doc.I_ACCOUNT = Trim (T003Message.ACC3)
      Doc.I_ADDRESS = Trim (T003Message.ADD13)
      Doc.I_CITY_STATE = Trim (T003Message.ADD23)
      Doc.I_POSTAL_CODE = Trim (T003Message.ADD33)
      Doc.I_BALANCE = Format (Ccur(T003Message.BAL3), "Currency")
      Doc.O_RESPONSES = Trim (T003Message.Mess3)
      Set MyMessage = T003Message
End Select
Send = "END"
Else
    Doc.O_RESPONSES = T003Message.Mess3
    Send = "END"
End If
End If
' Add an Account received
' We get the details of the account to be added from the document
' and send back the information to CICS
Case "DDW1MS.T004":
    Call ReceiveMessage(T004Message, ReceiveType)
    If T004Message.MESS4 = "" Then
        T004Message.SUR4 = Doc.GetItemValue("I_SURNAME") (0)
        T004Message.FOR4 = Doc.GetItemValue("I_FIRST_NAME") (0)
        T004Message.ADD4 = Doc.GetItemValue("I_ADDRESS") (0)
        T004Message.ADD24 = Doc.GetItemValue("I_CITY_STATE") (0)
        T004Message.ADD34 = Doc.GetItemValue("I_POSTAL_CODE") (0)
        T004Message.BAL4 = Cstr(Doc.GetItemValue("I_BALANCE") (0))
        Set MyMessage = T004Message
        Send = "YES"
    Else
        Doc.I_ACCOUNT = " "
        Doc.O_RESPONSES = T004Message.MESS4
        Send = "END"
    End If
End If
' Delete an Account received
' We get the account number to be deleted from the document
' and send back the information to CICS
Case "DDW1MS.T005":
    Call ReceiveMessage(T005Message, ReceiveType)
    If T005Message.MESS5 = "" Then
        T005Message.ACC5 = Doc.GetItemValue("I_ACCOUNT") (0)
        Set MyMessage = T005Message
        Send = "YES"
    Else
        Doc.O_RESPONSES = T005Message.MESS5
        Send = "END"
    End If
End If
' Account Update received
' We get the details of the account to be updated from the document
' and send back the information to CICS
Case "DDW1MS.T006":
    Call ReceiveMessage(T006Message, ReceiveType)
    If T006Message.MESS6 = "" Then
        T006Message.SUR6 = Doc.GetItemValue("I_SURNAME") (0)
        T006Message.FOR6 = Doc.GetItemValue("I_FIRST_NAME") (0)
        T006Message.ADD16 = Doc.GetItemValue("I_ADDRESS") (0)
        T006Message.ADD26 = Doc.GetItemValue("I_CITY_STATE") (0)
        T006Message.ADD36 = Doc.GetItemValue("I_POSTAL_CODE") (0)
        T006Message.BAL6 = Cstr(Doc.GetItemValue("I_BALANCE") (0))
        Set MyMessage = T006Message
        Send = "YES"
    Else
        Doc.O_RESPONSES = T006Message.MESS6
        Send = "END"
    End If
    Figure 40 (Part 14 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service

Appendix B. Lotus Notes Agents  149
End If
'* Update an Account received'* We get the account number to be updated from the document'* and send back the information to CICS
Case "DDW1MS.T007":
Call ReceiveMessage(T007Message, ReceiveType)
If T007Message.MESS7 = "" Then
T007Message.ACC7 = Doc.GetItemValue("I_ACCOUNT")
Set MyMessage = T007Message
Send = "YES"
Else
Doc.O_RESPONSES = T007Message.MESS7
Send = "END"
End If
End Select
End Sub

Figure 40 (Part 15 of 15). Lotus Notes Client Agent for CICS 3270 Direct Service
B.3 Lotus Notes Client Agent for CICS DPL via MQSeries Service

'Client Agent (CICS DPL via MQSeries):

******************************************************************************
** SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator
** ITSO Sample Application : Lotus Notes Client Agent for the MQEI CICS DPL via MQSeries Service
******************************************************************************
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselex "eilsx"

******************************************************************************
** Define Enterprise Integrator objects.
******************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendMessage As EIMessage
Dim MyReceiveMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

******************************************************************************
** Notes variables for form manipulation.
******************************************************************************
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim ws As NotesUiWorkspace
Dim uidoc As NotesUiDocument

******************************************************************************
** Wait time in milliseconds for a reply.
******************************************************************************
Const FIVE_SECONDS = 5000  ' 5 seconds

******************************************************************************
** To hold values expected from the EI Definition database
******************************************************************************
Const SERVICE_NAME = "MQCICSDPL"
Const SEND_MESSAGE_NAME = "VSAMSERVCOMAREA"
Const RECEIVE_MESSAGE_NAME = "VSAMSERVCOMAREA"

******************************************************************************

Figure 41 (Part 1 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service
" Print text
**********************************************************************************
Const INIT_TEXT = "Initializing..."
Const SEND_TEXT = "Sending..."
Const RECEIVE_TEXT = "Receiving..."
Const CONSTRUCTING = "Constructing Message..."
Const SENDSUCCESS = "Send Message Completed Successfully"
Const RECEIVESUCCESS = "Receive Message Completed Successfully"
Const CONNECT_TEXT = "Connecting..."
Const DISCONNECT_TEXT = "Disconnecting..."
Const OK_TEXT = "Press 'OK' button to Send Message..."
Const BLANK = " "
**********************************************************************************
Text for use in error message box
**********************************************************************************
Const WARNINGTITLE = "MQEI Warning"
Const ERRORTITLE = "MQEI Error"
Const GENERALTITLE = "General LotusScript Error"
Const NOTEERROR = "LotusScript Error. Exit Sample..."
Const MQIERERROR = "MQEI Error. Exit Sample..."
Const NOTEERROREXIT = "LotusScript Error. Exiting Sample..."
Const MQIERERROREXIT = "MQEI Error. Exiting Sample..."
Const PRIMARYSYSTEM = "Primary System Error Code"
Const SECONDARYSYSTEM = "Secondary System Error Code"
Const SYSTEMERRORTEXT = "Description"
Const TITLE_TEXT = "MQ Native sample"
Const LOTUSSCRIPT_ERROR_TEXT = "LotusScript error!"
Const EILSX_ERROR_TEXT = "EILSX error!"
Const ERROR_NUMBER_TEXT = "Error number"
Const ERROR_DESCRIPTION_TEXT = "Error description"
Const LINE_NUMBER_TEXT = "Line number"
Const EIRC_CONNECTED_TEXT = "An attempt was made to connect to a service when already connected.
  "_ + "The connect is ignored and the program continues.""
Const EIRC_DISCONNECTED_TEXT = "An attempt was made to disconnect to a service when already disconnected.
  "_ + "disconnected. The disconnect is ignored and the program continues.""
Const EIRC_NO_MESSAGE_AVAILABLE_TEXT = "No message available - Retry operation?"
Const EIRC_FIELD_TRUNCATED_TEXT = "A length of a string value of an EIMessage field exceeds the _
  "length specified on the Definition Database. Correct the LotusScript to make sure that the string _
  "value being passed as the EIMessage field value does not exceed the predefined length.""
Const EIRC_WRONG_MESSAGE_TEXT = "$Message was received that does not match the EIMessage object _
  "as passed as a parameter on the ElService ReceiveMessage call."
Const UNRECOGNISED_TEXT = "Unrecognised warning message!"
Const EISSESSION_ERROR_TEXT = "An error has been reported from the EISession Class. ReasonCode: 
  "EIMESSAGE_ERROR_TEXT = "An error has been reported from the EIMessage Class. ReasonCode: 
  "EISERVICE_ERROR_TEXT = "An error has been reported from the EIService Class. ReasonCode: 
  "EI30_ERROR_TEXT = "An error has been reported from the EIOptions Class. ReasonCode: 
  "EIRX_ERROR_TEXT = "An error has been reported from the EIRxOptions Class. ReasonCode: 
  "SAMPLE_END_TEXT = "Sample completed."
**********************************************************************************

Figure 41 (Part 2 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service

152 Lotus Notes and the MQSeries Enterprise Integrator
Const EIRC_CONNECTED_STRING = "An attempt was made to connect " + "to a service when already connected. The connect is ignored and " + "the program continues."
Const EIRC_NO_MSG_AVAILABLE_STRING = "An attempt was made to " + "receive a message but the target message was not found. " + "Please, retry later."
Const EIRC_FIELD_TRUNCATED_STRING = "A length of a string value of " + "an EIMessage field exceeds the length specified on the MQSeries EI Definition " + "database. Correct the LotusScript to make sure that the string value " + "being passed as the EIMessage field value does not exceed the " + "predefined length."
Const EIRC_WRONG_MESSAGE_STRING = "A message was received that " + "does not match the EIMessage object passed as a parameter on the " + "EIService ReceiveMessage call."
Const UNRECOGNISED_STRING = "Unrecognized warning message."
Const EISERVICE_WARNING = "Warning message received establishing " + "MQService. ReasonCode:
Const ESESSION_ERROR = "An error has been reported from the " + "EISession Class. ReasonCode:
Const THISEMESSAGE_ERROR = "An error has been reported from the " + "EIMessage Class. ReasonCode:
Const THISSERVICE_ERROR = "An error has been reported from the " + "EIService Class. ReasonCode:
Const THISSO_ERROR = "An error has been reported from the " + "EISendOptions Class. ReasonCode:
Const THISRO_ERROR = "An error has been reported from the " + "EISendOptions Class. ReasonCode:

Const SF_NOPREPROCESS_YE="No MQ Reply found"
Const SD_NOPREPROCESS_YE="Request has been submitted, but no reply has been received " + "yet. Press button to retry!"
Const SF_NONE=""
Const SD_NONE=""
Const SF_PROCESSED="Processed"

Sub Initialize
"******************************************
' SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator"
' ITSO Sample Application -
' Lotus Notes Client Agent for the MQEI CICS DPL via MQSeries Service
'*****************************************************************************
' This sample illustrates how a Lotus Script application can communicate
' to an enterprise system using MQSeries
'*****************************************************************************
' A LotusScript application writes to a MQSeries queue located on MVS.
' The MQSeries-CICS/ESA DPL bridge monitor starts a CICS DPL bridge

Figure 41 (Part 3 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service
\*\* task which gets the message and call the program VSAMSERV to process
\*\* the message.
\*\* VSAMSERV then returns to the MQSeries-CICS/ESA DPL bridge task which puts
\*\* a reply back to a remote OS/2 queue.
\*\* The LotusScript application gets the message and update the document.
\*\*
\*\* This agent is activated when the "CICS DPL via MQSeries" Service Selection radio
\*\* button is selected and the "Submit" button is clicked.
\*\*
\*\* NOTE: When you see a number enclosed in '<>' and '>>' characters, (eg. '<<1>>)
\*\* it is referring to a step described in the
\*\* chapter 5.1.3 Implementation of the Lotus Notes Client Access (Two Tier Model)
\*\* in the ITSO Redbook:
*******************************************************************************************

*******************************************************************************************

\* Initialize the EILSX objects. <<4>>
*******************************************************************************************
Print INIT_TEXT

*******************************************************************************************
\* Set up the error handlers. (Note: These are not event handlers).
*******************************************************************************************
On Error Goto GeneralErrorHandler
On Error EILSX_ERROR Goto MQEIErrorHandler

*******************************************************************************************

\* Initialize the EIObject object <<4>>
*******************************************************************************************
Set MySession = New EISession

\* Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
    Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
    Call MySession.ClearErrorCodes()
End If

*******************************************************************************************

\* Create an EI Service
\* <<4>> defined in the EI Definition database).
*******************************************************************************************
Set MyService = MySession.CreateService(SERVICE_NAME)
On Event EIError From MyService Call EIServiceErrorEventHandler

*******************************************************************************************

\* Create EI send message
\* <<4>> defined in the EI Definition database).
*******************************************************************************************
Set MySendMessage = MySession.CreateMessage(SEND_MESSAGE_NAME)
On Event EIMessageErrorEventHandler From MySendMessage Call EIMessageErrorEventHandler

If MySendMessage.CompletionCode = EICC_WARNING Then

Figure 41 (Part 4 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service
Select Case MySendMessage.ReasonCode
Case EIRC_FIELD_TRUNCATED :
    MessageBox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
Case Else :
    MessageBox UNRECOGNISED_STRING & " " & Cstr(MySendMessage.ReasonCode), MB_OK, WARNINGTITLE
End Select
Call MySendMessage.ClearErrorCodes
End If

'*****************************************************************************************
' Create EI receive message
' <<4a>> defined in the EI Definition database).
'* Create EI receive message options
'*****************************************************************************************'
Set MyReceiveMessage = MySession.CreateMessage(RECEIVE_MESSAGE_NAME)
On Event EIError From MyReceiveMessage Call EIMessageErrorEventHandler
If MyReceiveMessage.CompletionCode = EICC_WARNING Then
    Select Case MyReceiveMessage.ReasonCode
    Case EIRC_FIELD_TRUNCATED :
        MessageBox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
    Case Else :
        MessageBox UNRECOGNISED_STRING & " " & Cstr(MyReceiveMessage.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MyReceiveMessage.ClearErrorCodes
    End If
End If

'*****************************************************************************************
' Create EI send message options
'*****************************************************************************************'
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler
Option must be set to make MQEI define the MQ message type 'request' in the MQ message descriptor
MySendOptions.MessageType=EIMT_REQUEST

'*****************************************************************************************
' Create EI receive message options
'*****************************************************************************************'
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler
MyReceiveOptions.WaitType=EIWT_WAIT
MyReceiveOptions.WaitInterval=FIVE_SECONDS

Print BLANK

'*****************************************************************************************
' Set document variables
'*****************************************************************************************'
Set Session = New NotesSession
Set Doc=Session.DocumentContext
Set ws = New NotesUiWorkspace

Figure 41 (Part 5 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service
Set Uidoc = ws.CurrentDocument

'************************************************************************************
'* Connect to EI Service <<5>>
'*******************************************************************************************
Print CONNECT_TEXT
Call MyService.Connect
If MySession.CompletionCode = EICC_WARNING Then
    Select Case MySession.ReasonCode
        Case EIRC_CONNECTED:
            MessageBox EIRC_CONNECTED_STRING, MB_OK, WARNINGTITLE
        Case Else:
            MessageBox EISERVICE_WARNING & " " & CStr(MySession.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MySession.ClearErrorCodes
End If

'************************************************************************************************
'* Send the Message to the EI Service selected if StatusFlag is not
'* SF_NO_MQ_REPLY_MSG_YET
'* otherwise message has been sent already and just a receive has to take
'* place
'************************************************************************************************
If Doc.GetFirstItem("StatusFlag").Text <> SF_NO_MQREPLY_YET Then
    Print SEND_TEXT
    '*****************************************************************************
    '* Setup and send message to CICS <<7>>
    '*****************************************************************************
    MySendMessage.SERVER_STATE = Doc.GetItemValue("I_SERVER_STATE") (0)
    MySendMessage.SERVER_KEY_SIZE = CInt(Doc.GetItemValue("I_SERVER_KEY_SIZE") (0))
    MySendMessage.SURNAME = Doc.GetItemValue("I_SURNAME") (0)
    MySendMessage.FIRST_NAME = Doc.GetItemValue("I_FIRST_NAME") (0)
    MySendMessage.ACCOUNT = Doc.GetItemValue("I_ACCOUNT") (0)
    MySendMessage.ADDRESS = Doc.GetItemValue("I_ADDRESS") (0)
    MySendMessage.CITY_STATE = Doc.GetItemValue("I_CITY_STATE") (0)
    MySendMessage.POSTAL_CODE = Doc.GetItemValue("I_POSTAL_CODE") (0)
    MySendMessage.BALANCE = CInt(Doc.GetItemValue("I_BALANCE") (0))
    '*****************************************************************************
    '* Send the message <<7>>
    '*****************************************************************************
    Call MyService.SendMessage ( MySendMessage, MySendOptions )
    Print SENDSUCCESS
    '*****************************************************************************
    '* Generated MQ message Id is stored in document for delayed
    '* receive option
    '*****************************************************************************
    Doc.MQRequestMsgId = MySendOptions.Identifier
End If

Figure 41 (Part 6 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service

156 Lotus Notes and the MQSeries Enterprise Integrator
Receive the message

'************************************************************************************************
'* Receive the message ... MyService.ReceiveMessage ( MyReceiveMessage, MyReceiveOptions )
'************************************************************************************************

'Handle MQEI warning messages'************************************************************************************************

Doc.StatusFlag=SF_NONE
Doc.StatusDescription=SD_NONE
If MyService.CompletionCode = EICC_WARNING Then
  Select Case MyService.ReasonCode
  Case EIRC_NO_MESSAGE_AVAILABLE :
    Messagebox EIRC_NO_MSG_AVAILABLE_STRING, MB_OK, WARNINGTITLE
    'Set Status Flag, so that user can retry later (delayed receive)
    Doc.StatusFlag=SF_NO_MQREPLY_YET
    Doc.StatusDescription=SD_NO_MQREPLY_YET
  Case EIRC_FIELD_TRUNCATED :
    Messagebox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
  Case EIRC_WRONG_MESSAGE :
    Messagebox EIRC_WRONG_MESSAGE_STRING, MB_OK, WARNINGTITLE
  Case Else :
    Messagebox UNRECOGNISED_STRING & " " & MyService.ReasonCode, MB_OK, WARNINGTITLE
  End Select
  Call MyService.ClearErrorCodes
Else
  Print RECEIVESUCCESS
  '************************************************************************************************
  Doc.StatusFlag=SF_PROCESSED 'Set status flag

Figure 41 (Part 7 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service

Appendix B. Lotus Notes Agents 157
End If

'*****************************************************************************
' Redisplay Updated document <<10>>
'*****************************************************************************
Call Doc.Save(True, True) ' Save backend document
Call Uidoc.reload ' Reload UI document
Call Uidoc.refresh ' Reload UI document

Call CleanUp ' Disconnect from enterprise services and delete all MQEI objects
Exit Sub

'*****************************************************************************
' General LotusScript Error Handler
'*****************************************************************************
GeneralErrorHandler:
    Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
    CleanUp
    Exit Sub

'*****************************************************************************
' Handle EILSX errors.
'*****************************************************************************
MQEIErrorHandler:
    Call GlobalErrors(EILSX_ERROR_TEXT)
    CleanUp
    Exit Sub

Sub GlobalErrors(CallerText As String)
'*****************************************************************************
' Output error handler messages in a formatted message box.
'*****************************************************************************
    Dim ErrorText As String
    Print CallerText

    ' Format the error text for the message box
    ErrorText = CallerText + Chr$(10)
    ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
    ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$
    ErrorText = ErrorText + Chr$(10) + LINE_NUMBER_TEXT + " = " + Str(Erl)

    ' Output the error to message box and print line
    MsgBox ErrorText, MB_OK, TITLE_TEXT
    Print "

End Sub
Sub EISessionErrorEventHandler(ThisSession As EISession , ReasonCode As Long)

Figure 41 (Part 8 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service

158 Lotus Notes and the MQSeries Enterprise Integrator
Error event handler for the EISession Class. At this point we choose not to take any corrective action, an error message is simply displayed in a message box.

Alternatively you could write a CASE statement to handle your chosen errors. Once handled the Error Codes can be cleared, allowing the main program to continue.

Note: this subroutine is registered as an error event handler for EISession errors but is also called following a failure in EISession creation in Queryopen.

******************************************************************************
Dim StrText As String

Select Case ReasonCode
  Case EIRC_???:
    ...Error Handle Process...
  Case EIRC_???:
    ...Error Handle Process...
  Case EIRC_SEC_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & & Cstr(ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + ":" + Cstr(ThisSession.SystemErrorText) + 
    + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + ":" + Cstr(ThisSession.PrimarySystemErrorCode) + 
    + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + ":" + Cstr(ThisSession.SecondarySystemErrorCode)
    Messagebox StrText, MB_OK, TITLE_TEXT
  Case Else:
    Messagebox EISESSION_ERROR_TEXT + ":" + Cstr(ThisSession.ReasonCode), MB_OK, TITLE_TEXT
End Select

******************************************************************************

Do not clear the error code, as this will allow error EILSX_ERROR to be caught by the generic error handler.
******************************************************************************

End Sub
Sub CleanUp
******************************************************************************

This function is called when the agent is finished.
******************************************************************************

On Error Resume Next ' Ignore any error as we need to shutdown regardless.
******************************************************************************

Disconnect from MQSeries.
******************************************************************************

Print DISCONNECT_TEXT
Call MyService.Disconnect()

Figure 41 (Part 9 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service
"Delete the EI objects.
*****************************************************************************
Delete MyReceiveOptions
Delete MySendOptions
Delete MySendMessage
Delete MyReceiveMessage
Delete MyService
Delete MySession
End Sub
Sub EIServiceErrorHandler (thisService As EIService, ReasonCode As Long)
*****************************************************************************
" Error Handler for the EIService Class. At this point we choose not to take
" any corrective action, the error is simply displayed in a message box.
" Alternatively you could write a CASE statement to handle your chosen
" errors. Once handled the Error Codes can be cleared, allowing the
" main program to continue.
*****************************************************************************
Select Case ReasonCode
  Case EIRC_???:
    "...Error Handle Process...
    thisService.clearErrorCodes
  Case EIRC_???:
    "...Error Handle Process...
    thisService.clearErrorCodes
  Case Else :
    Messagebox EISERVICE_ERROR_TEXT & " & Cstr(ReasonCode) & ". Service Name: " & thisService.Name & ", MB_OK, TITLE_TEXT
End Select
End Sub
Sub EIMessageErrorHandler (thisMessage As EIMessage, ReasonCode As Long)
*****************************************************************************
" Error Handler for the EIMessage Class. At this point we choose not to take
" any corrective action, the error is simply displayed in a message box.
" Alternatively you could write a CASE statement to handle your chosen
" errors. Once handled the Error Codes can be cleared, allowing the
" main program to continue.
*****************************************************************************
Select Case ReasonCode
  Case EIRC_??? :
    "...Error Handle Process...
    thisMessage.clearErrorCodes
  Case EIRC_??? :
    "...Error Handle Process...
    thisMessage.clearErrorCodes
  Case Else :
    Messagebox EIMESSAGE_ERROR_TEXT & " & Cstr(ReasonCode) & ", Message Name: " & thisMessage.Name & ", MB_OK, TITLE_TEXT
End Select
End Sub

Figure 41 (Part 10 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service

160 Lotus Notes and the MQSeries Enterprise Integrator
End Select
End Sub
Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long)
'***********************************************
' Error Handler for the EIService Class. At this point we choose not to take
' any corrective action, the error is simply displayed in a message box.
' *
' Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
' main program to continue.
'***********************************************
Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
    thisSendOptions.clearErrorCodes
  Case EIRC_??? :
    ...Error Handle Process...
    thisSendOptions.clearErrorCodes
  Case Else :
    messagebox EISO_ERROR_TEXT & " " & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
End Select
End Sub

Sub EIReceiveOptionsErrorEventHandler (thisReceiveOptions As EIReceiveOptions, ReasonCode As Long)
'***********************************************
' Error Handler for the EIService Class. At this point we choose not to take
' any corrective action, the error is simply displayed in a message box.
' *
' Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
' main program to continue.
'***********************************************
Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
    thisReceiveOptions.clearErrorCodes
  Case EIRC_??? :
    ...Error Handle Process...
    thisReceiveOptions.clearErrorCodes
  Case Else :
    messagebox EIRO_ERROR_TEXT & " " & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
End Select
End Sub

Figure 41 (Part 11 of 11). Lotus Notes Client Agent for CICS DPL via MQSeries Service
B.4 Lotus Notes Client Agent for Native MQSeries Service

'Client Agent (Native MQSeries):
*******************************************************************************************
'SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator
'* ITSO Sample Application
'* Lotus Notes Client Agent for the MQEI Native MQSeries Service
******************************************************************************
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselsx "elsx"

******************************************************************************
'* Define Enterprise Integrator objects.
******************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendMessage As EIMessage
Dim MyReceiveMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

******************************************************************************
'* Notes variables for form manipulation.
******************************************************************************
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim ws As NotesUiWorkspace
Dim uidoc As NotesUiDocument
Dim MyDb As NotesDatabase

******************************************************************************
'* Wait time in milliseconds for a reply.
******************************************************************************
Const FIVE_SECONDS = 5000 ' 5 seconds

******************************************************************************
'* To hold values expected from the EI Definition database
******************************************************************************
Const SERVICE_NAME = "MQServ"
Const SEND_MESSAGE_NAME = "VSAMSERVCOMMAREA"
Const RECEIVE_MESSAGE_NAME = "VSAMSERVCOMMAREA"

Figure 42 (Part 1 of 11). Lotus Notes Client Agent for Native MQSeries Service
Appendix B. Lotus Notes Agents 163

Figure 42 (Part 2 of 11). Lotus Notes Client Agent for Native MQSeries Service
Sub Initialize

Sub Initialize

Figure 42 (Part 3 of 11). Lotus Notes Client Agent for Native MQSeries Service
* A CICS transaction, MQ1, is started via MQSeries triggering.
* CICS program AMQCIC1 then executes to get the message from the queue and calls
  program VSAMSERV to process the message.
* VSAMSERV then returns to AMQCIC1 which puts a reply back to a queue.
* The LotusScript application gets the message and update the document.
* This agent is activated when the "Native MQ Series" Service Selection radio
  button is selected and the "Submit" button is clicked.
* "NOTE: When you see a number enclosed in '<<' and '>>' characters, (eg. <<1>>)
  it is referring to a step described in the
chapter 5.1.3 Implementation of the Lotus Notes Client Access (Two Tier Model)
* in the ITSO Redbook:
*******************************************************************************************
*******************************************************************************************
* Initialize the EILSX objects. <<4>>
*******************************************************************************************
Print INIT_TEXT

'***************************************************************************
' Set up the error handlers. (Note: These are not event handlers).
'***************************************************************************
On Error Goto GeneralErrorHandler
On Error EILSX_ERROR Goto MQEIErrorHandler

'***************************************************************************
' Initialize the EISession object <<4>
'***************************************************************************
Set MySession = New EISession

' Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
  Call EIServiceErrorEventHandler(MySession, MySession.ReasonCode)
  Call MySession.ClearErrorCodes()
End If

'***************************************************************************
' Create an EI Service
'***************************************************************************
Set MyService = MySession.CreateService(SERVICE_NAME)
On Event EIServiceError From MyService Call EIServiceErrorEventHandler

'***************************************************************************
' Create EI send message
'***************************************************************************
Set MySendMessage = MySession.CreateMessage(SEND_MESSAGE_NAME)
On Event EIError From MySendMessage Call EIMessageErrorEventHandler

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)

Figure 42 (Part 4 of 11). Lotus Notes Client Agent for Native MQSeries Service
If MySendMessage.CompletionCode = EICC_WARNING Then
    Select Case MySendMessage.ReasonCode
    Case EIRC_FIELD_TRUNCATED :
        MessageBox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
    Case Else :
        MessageBox UNRECOGNISED_STRING & " " & CStr(MySendMessage.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MySendMessage.ClearErrorCodes()
End If

*****************************************************************************************
' Create EI receive message
' <<4a>> defined in the EI Definition database).
*****************************************************************************************
Set MyReceiveMessage = MySession.CreateMessage(RECEIVE_MESSAGE_NAME)
On Event EIError From MyReceiveMessage Call EIMessageErrorEventHandler

If MyReceiveMessage.CompletionCode = EICC_WARNING Then
    Select Case MyReceiveMessage.ReasonCode
    Case EIRC_FIELD_TRUNCATED :
        MessageBox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
    Case Else :
        MessageBox UNRECOGNISED_STRING & " " & CStr(MyReceiveMessage.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MyReceiveMessage.ClearErrorCodes()
End If

*****************************************************************************************
' Create EI send message options
*****************************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler

‘Option must be set to make MQEI define the MQ message type 'request' in the MQ message descriptor
MySendOptions.MessageType=EIMT_REQUEST

*****************************************************************************************
' Create EI receive message options
*****************************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler

MyReceiveOptions.WaitType=EIWT_WAIT
MyReceiveOptions.WaitInterval=FIVE SECONDS

*****************************************************************************************
‘ Set document variables
*****************************************************************************************
Set Session = New NotesSession
Set Doc=Session.DocumentContext
Set ws = New NotesUiWorkspace
Set Uidoc = ws.CurrentDocument

Figure 42 (Part 5 of 11). Lotus Notes Client Agent for Native MQSeries Service
'************************************************************************************
'* Connect to EI Service <<5>>
'*******************************************************************************/
Print CONNECT_TEXT
Call MyService.Connect() ' Connect to enterprise services

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
  Select Case MyService.ReasonCode
  Case EIRC_CONNECTED :
    MessageBox EIRC_CONNECTED_STRING, MB_OK, WARNINGTITLE
  Case Else :
    MessageBox EISERVICE_WARNING & " " & Cstr(MyService.ReasonCode), MB_OK, WARNINGTITLE
  End Select
  Call MyService.ClearErrorCodes
End If

'*******************************************************************************/
'* Send the Message to the EI Service selected if StatusFlag is not'* SF_NO_MQ_REPLY_MSG_YET'* otherwise message has been sent already and just a receive has to take'* place'********************************************************************************/
If Doc.GetFirstItem("StatusFlag").Text<>SF_NO_MQREPLY_YET Then
  Print SEND_TEXT
    ** Setup and send message to CICS <<6>>
    *******************************************************************************/
    MySendMessage.SERVER_STATE = Doc.GetItemValue("I_SERVER_STATE")(0)
    MySendMessage.SERVER_KEY_SIZE = CInt(Doc.GetItemValue("I_SERVER_KEY_SIZE")(0))
    MySendMessage.SURNAME = Doc.GetItemValue("I_SURNAME")(0)
    MySendMessage.FIRST_NAME = Doc.GetItemValue("I_FIRST_NAME")(0)
    MySendMessage.ACCOUNT = Doc.GetItemValue("I_ACCOUNT")(0)
    MySendMessage.ADDRESS = Doc.GetItemValue("I_ADDRESS")(0)
    MySendMessage.CITY_STATE = Doc.GetItemValue("I_CITY_STATE")(0)
    MySendMessage.POSTAL_CODE = Doc.GetItemValue("I_POSTAL_CODE")(0)
    MySendMessage.BALANCE = CInt(Doc.GetItemValue("I_BALANCE")(0))
    *******************************************************************************/
    ** Send the message <<7>>
    *******************************************************************************/
    Call MyService.SendMessage ( MySendMessage, MySendOptions )
    Print SENDSUCCESS
    *******************************************************************************/
    ** Generated MQ message Id is stored in document for delayed'* receive option'********************************************************************************/
    Doc.MQRequestMsgId=MySendOptions.Identifier

Figure 42 (Part 6 of 11). Lotus Notes Client Agent for Native MQSeries Service
End If

]************************************************************************************************
]
]Receive the message <<$>>
]************************************************************************************************

Print RECEIVE_TEXT

' Identifier property must be set to receive the reply message to a given request message

' which is identified by the identifier stored in the MQRequestMsgId of the form
' otherwise MQEI would receive the first message in the queue

MyReceiveOptions.Identifier=Doc.GetItemValue("MQRequestMsgId")(1)
Call MyService.ReceiveMessage ( MyReceiveMessage, MyReceiveOptions )

'******************************************************************************
]
] Handle MQEI warning messages
]******************************************************************************

Doc.StatusFlag=SF_NONE
Doc.StatusDescription=SD_NONE
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
    Case EIRC_NO_MESSAGE_AVAILABLE :
        MessageBox EIRC_NO_MSG_AVAILABLE_STRING, MB_OK, WARNINGTITLE
        " " Set Status Flag, so that user can retry later (delayed receive)
        Doc.StatusFlag=SF_NO_MQREPLY_YET
        Doc.StatusDescription=SD_NO_MQREPLY_YET

    Case EIRC_FIELD_TRUNCATED :
        MessageBox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
    Case EIRC_WRONG_MESSAGE :
        MessageBox EIRC_WRONG_MESSAGE_STRING, MB_OK, WARNINGTITLE
    Case Else :
        MessageBox UNRECOGNISED_STRING & " " & MyService.ReasonCode, MB_OK, WARNINGTITLE
    End Select
    Call MyService.ClearErrorCodes

Else

Print RECEIVESUCCESS

'******************************************************************************
]
] Update backend document with receive message <<$>>
]******************************************************************************

Doc.I_SERVER_STATE=Trim(MyReceiveMessage.SERVER_STATE)
Doc.I_SERVER_KEY_SIZE=Cstr(MyReceiveMessage.SERVER_KEY_SIZE)
Doc.I_SURNIAE=Trim(MyReceiveMessage.SURNAME)
Doc.I_FIRST_NAME=Trim(MyReceiveMessageIRST_NAME)
Doc.I_ACCOUNT=Trim(MyReceiveMessage.ACCOUNT)
Doc.I_ADDRESS=Trim(MyReceiveMessage.ADDRESS)
Doc.I_CITY_STATE =Trim(MyReceiveMessage.CITY_STATE)
Doc.I_POSTAL_CODE =Trim(MyReceiveMessage.POSTAL_CODE)
Doc.I_BALANCE=Cstr(MyReceiveMessage.BALANCE)
Doc.I_RESPONSES=Trim(MyReceiveMessage.RESPONSES)

Figure 42 (Part 7 of 11). Lotus Notes Client Agent for Native MQSeries Service

168 Lotus Notes and the MQSeries Enterprise Integrator
Figure 42 (Part 8 of 11). Lotus Notes Client Agent for Native MQSeries Service
* Error event handler for the EISession Class. At this point we choose not to take
* any corrective action, an error message is simply displayed in a message box.
* Alternatively you could write a CASE statement to handle your chosen
* errors. Once handled the Error Codes can be cleared, allowing the
* main program to continue.
* Note: this subroutine is registered as an error event handler for EISession
* events but is also called following a failure in EISession creation in Queryopen
*******************************************************************************************
Dim StrText As String

Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
  Case EIRC_??? :
    ...Error Handle Process...
  Case EIRC_SEC_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & ": " & Cstr(ReasonCode) & "," + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + ": " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + ": " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + ": " + Cstr(ThisSession.SecondarySystemErrorCode)
    Messagebox StrText, MB_OK, TITLE_TEXT
  Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & ": " & Cstr(ThisSession.ReasonCode) & "," + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + ": " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + ": " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + ": " + Cstr(ThisSession.SecondarySystemErrorCode)
    Messagebox StrText, MB_OK, TITLE_TEXT
  Case Else :
    Messagebox EISESSION_ERROR_TEXT + ": " + Cstr(ThisSession.ReasonCode) , MB_OK, TITLE_TEXT
End Select

On Error Resume Next ' Ignore any error as we need to shutdown regardless.

*******************************************************************************************
* Do not clear the error code, as this will allow error EILSX_ERROR to be
* caught by the generic error handler.
*******************************************************************************************
End Sub
Sub CleanUp
  ' This function is called when the agent is finished.
  ' Disconnect from MQSeries and delete the EI objects.
*******************************************************************************************
  On Error Resume Next ' Ignore any error as we need to shutdown regardless.
  '*******************************************************************************************
  Print DISCONNECT_TEXT

Figure 42 (Part 9 of 11). Lotus Notes Client Agent for Native MQSeries Service

Lotus Notes and the MQSeries Enterprise Integrator
Call MyService.Disconnect()

'*' Delete the EI objects.
'*****************************************************************************
Delete MyReceiveOptions
Delete MySendOptions
Delete MySendMessage
Delete MyReceiveMessage
Delete MyService
Delete MySession
End Sub
Sub EIServiceErrorEventHandler (thisService As EIService, ReasonCode As Long)
'*****************************************************************************
'** Error Handler for the EIService Class. At this point we choose not to take
'** any corrective action, the error is simply displayed in a message box.
'**
'** Alternatively you could write a CASE statement to handle your chosen
'** errors. Once handled the Error Codes can be cleared, allowing the
'** main program to continue.
'*****************************************************************************
    Select Case ReasonCode
    ' Case EIRC_???:
    '      ...Error Handle Process...
    '      thisService.clearErrorCodes
    ' Case EIRC_???:
    '      ...Error Handle Process...
    '      thisService.clearErrorCodes
    '      ...
    Case Else:
        MessageBox EISERVICE_ERROR_TEXT & ", Service Name: " & _
            thisService.Name & ", MB_OK, TITLE_TEXT
    End Select
End Sub
Sub EIMessageErrorEventHandler (thisMessage As EIMessage, ReasonCode As Long)
'*****************************************************************************
'** Error Handler for the EIService Class. At this point we choose not to take
'** any corrective action, the error is simply displayed in a message box.
'**
'** Alternatively you could write a CASE statement to handle your chosen
'** errors. Once handled the Error Codes can be cleared, allowing the
'** main program to continue.
'*****************************************************************************
    Select Case ReasonCode
    ' Case EIRC_???: ...Error Handle Process...
    '      thisMessage.clearErrorCodes
    ' Case EIRC_???: ...Error Handle Process...
    '      thisMessage.clearErrorCodes
    '      ...
    Case Else:
        MessageBox EIMESSAGE_ERROR_TEXT & ", Message Name: " & _
                        thisMessage.Name & ", MB_OK, TITLE_TEXT
    End Select
End Sub

Figure 42 (Part 10 of 11). Lotus Notes Client Agent for Native MQSeries Service
Figure 42 (Part 11 of 11). Lotus Notes Client Agent for Native MQSeries Service
B.5 Lotus Notes Client Agent for IMS via MQSeries Service

*Client Agent (IMS via MQSeries)*:

********************************************************************************************

* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator
* ITSO Sample Application
* Lotus Notes Client Agent for the MQEI IMS via MQSeries Service

*****************************************************************************

Option Public
Option Explicit

* Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

* Load the MQSeries Enterprise Integrator library
Uselex "eilx"

*****************************************************************************

* Define Enterprise Integrator objects.
*****************************************************************************

Dim MySession As EISession
Dim MyService As EIService
Dim MySendMessage As EIMessage
Dim MyReceiveMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

*****************************************************************************

* Wait time in milliseconds for a reply.
*****************************************************************************

Const FIVE_SECONDS = 5000  ' 5 seconds

*****************************************************************************

* Notes variables for form manipulation.
*****************************************************************************

Dim Session As NotesSession
Dim Doc As NotesDocument
Dim ws As NotesUiWorkspace
Dim uidoc As NotesUiDocument

*****************************************************************************

* To hold values expected from the EI Definition database
*****************************************************************************

Const SERVICE_NAME = "IMSMQ"
Const SEND_MESSAGE_NAME = "IMS_REQUEST"
Const RECEIVE_MESSAGE_NAME = "IMS_REPLY"

*****************************************************************************

Figure 43 (Part 1 of 11). Lotus Notes Client Agent for IMS via MQSeries Service
** Print text
**********************************************************************************
Const INIT_TEXT = "Initializing..."
Const SEND_TEXT = "Sending..."
Const RECEIVE_TEXT = "Receiving..."
Const CONSTRUCTING = "Constructing Message..."
Const SENDSUCCESS = "Send Message Completed Successfully"
Const RECEIVESUCCESS = "Receive Message Completed Successfully"
Const CONNECT_TEXT = "Connecting..."
Const DISCONNECT_TEXT = "Disconnecting..."
Const OK_TEXT = "Press 'OK' button to Send Message..."
Const BLANK = " 
**********************************************************************************
** Text for use in error message box
**********************************************************************************
Const WARNINGTITLE = "MQEI Warning"
Const ERRORTITLE = "MQEI Error"
Const GENERALTITLE = "General LotusScript Error"
Const NOTERROR = "LotusScript Error. Exit Sample."  
Const MQIERROREXIT = "MQEI Error. Exit Sample."  
Const NOTERROREXIT = "LotusScript Error. Exiting Sample..."
Const MQIERROREXIT = "MQEI Error. Exiting Sample..."
Const PRIMARYSYSTEM = "Primary System Error Code"
Const SECONDARYSYSTEM = "Secondary System Error Code"
Const SYSTEMERRORTEXT = "Description"
Const TITLE_TEXT = "IMS MQ sample"  
Const LOTUSSCRIPT_ERROR_TEXT = "LotusScript error!"
Const EILSX_ERROR_TEXT = "EILSX error!"
Const ERROR_NUMBER_TEXT = "Error number"
Const ERROR_DESCRIPTION_TEXT = "Error description"
Const LINE_NUMBER_TEXT = "Line number"
Const EIRC_CONNECTED_TEXT = "An attempt was made to connect to a service when already connected. "+ " The connect is ignored and the program continues."
Const EIRC_DISCONNECTED_TEXT = "An attempt was made to disconnect to a service when already "+ " disconnected. The disconnect is ignored and the program continues."
Const EIRC_NO_MESSAGEAVAILABLE_TEXT = "No message available - Retry operation?"
Const EIRC_FIELD_TRUNCATED_TEXT = "A length of a string value of an EIMessage field exceeds "+ " the length specified on the Definition Database. Correct the LotusScript to make sure that "+ " the string value being passed as the EIMessage field value does not exceed the predefined length."
Const EIRC_WRONGMESSAGE_TEXT = "A message was received that does not match the EIMessage object "+ " passed as a parameter on the EIService ReceiveMessage call."
Const UNRECOGNISED_TEXT = "Unrecognised warning message;"
Const EISERVICE_WARNING_TEXT = "Warning message received establishing MQService. ReasonCode: 

**********************************************************************************
Figure 43 (Part 2 of 11). Lotus Notes Client Agent for IMS via MQSeries Service

174 Lotus Notes and the MQSeries Enterprise Integrator
**Definition of warning/error messages**

******************************************************************************************

Const EIRC_CONNECTED_STRING = "An attempt was made to connect " + "+ "the program continues."
Const EIRC_NO_MSGAVAILABLE_STRING = "An attempt was made to " + "+ "receive a message but the target message was not found. " + "+ "Please, retry later."
Const EIRC_FIELD_TRUNCATED_STRING = "A length of a string value of " + "+ "an EIMessage field exceeds the length specified on the MQSeries EI Definition " + "+ "database. Correct the LotusScript to make sure that the string value " + "+ "being passed as the EIMessage field value does not exceed the " + "+ "predefined length."
Const EIRC_WRONG_MESSAGE_STRING = "A message was received that " + "+ "does not match the EIMessage object passed as a parameter on the " + "+ "EIService ReceiveMessage call."
Const UNRECOGNISED_STRING = "Unrecognized warning message."
Const EISERVICE_WARNING = "Warning message received establishing " + "+ "MQSeries. ReasonCode:" Const EISSESSION_ERROR = "An error has been reported from the " + "+ "EISession Class. ReasonCode"
Const THISMESSAGING_ERROR = "An error has been reported from the " + "+ "EIMessage Class. ReasonCode :"
Const THISSERVICE_ERROR = "An error has been reported from the " + "+ "EIService Class. ReasonCode :"
Const THISSEND/error = "An error has been reported from the " + "+ "EIService Options Class. ReasonCode :"
Const THISRE_ERROR = "An error has been reported from the " + "+ "EIService Options Class. ReasonCode :"

******************************************************************************************

**Definition of StatusFlag Values and Status Description**

******************************************************************************************

Const SF_NO_MQREPLY_YET="No MQ Reply found"
Const SD_NO_MQREPLY_YET="Request has been submitted, but no reply has been received yet. " + "+ "Press button to retry!"
Const SF_NONE=""
Const SD_NONE=""
Const SF_PROCESSED="Processed"

Sub Initialize

******************************************************************************************

\* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator*
\* ITSO Sample Application -
\* Lotus Notes Client Agent for the MQEI IMS via MQSeries Service

******************************************************************************************

\* This sample illustrates how a Lotus Script application can access an IMS transaction
\* A LotusScript application writes to a MQSeries queue.
\* The MQSeries-IMS bridge gets the message and uses the OTMA

**Figure 43 (Part 3 of 11). Lotus Notes Client Agent for IMS via MQSeries Service**
In the ITSO Redbook: 
****************************************************************************

Figure 43 (Part 4 of 11). Lotus Notes Client Agent for IMS via MQSeries Service

Lotus Notes and the MQSeries Enterprise Integrator
Case EIRC_FIELD_TRUNCATED :
    MessageBox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
Case Else :
    MessageBox UNRECOGNISED_STRING & " " & Cstr(MySendMessage.ReasonCode), MB_OK, WARNINGTITLE
End Select
Call MySendMessage.ClearErrorCodes
End If

'*****************************************************************************************
'* Create EI receive message'* <<4a>> defined in the EI Definition database).
Set MyReceiveMessage = MySession.CreateMessage(RECEIVE_MESSAGE_NAME)
On Event EIError From MyReceiveMessage Call EIMessageErrorEventHandler
If MyReceiveMessage.CompletionCode = EICC_WARNING Then
    Select Case MyReceiveMessage.ReasonCode
        Case EIRC_FIELD_TRUNCATED :
            MessageBox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
        Case Else :
            MessageBox UNRECOGNISED_STRING & " " & Cstr(MyReceiveMessage.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MyReceiveMessage.ClearErrorCodes
End If

'*****************************************************************************************
'* Create EI send message options'*****************************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler
'* The MySendOptions.MessageType property is ignored by the MQEI LSX and
'* is overridden internally to EIMT_REQUEST

'*****************************************************************************************
'* Create EI receive message options'******************************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler
MyReceiveOptions.WaitType=EIWT_WAIT
MyReceiveOptions.WaitInterval=FIVE_SECONDS

'*****************************************************************************************
'* Set document variables'******************************************************************************************
Set Session = New NotesSession
Set Doc=Session.DocumentContext
Set ws = New NotesUiWorkspace
Set Uidoc = ws.CurrentDocument

Figure 43 (Part 5 of 11). Lotus Notes Client Agent for IMS via MQSeries Service
Connect to EI Service

Check for warnings (We have decided to ignore warnings for the purposes of this sample)

If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
        Case EIRC_CONNECTED:
            MessageBox EIRC_CONNECTED_STRING, MB_OK, WARNINGTITLE
        Case Else:
            MessageBox EISERVICE_WARNING & " & Cstr(MyService.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MyService.ClearErrorCodes
End If

Send the Message to the EI Service selected if StatusFlag is not SF_NO_MQ_REPLY_MSG_YET otherwise message has been sent already and just a receive has to take place

If Doc.GetFirstItem("StatusFlag").Text<>SF_NO_MQREPLY_YET Then
    Print SEND_TEXT
        ' Setup and send message to IMS
        MySendMessage.Format = EFIWT_NONE ' No MFS Map name is set
        MySendMessage.cmd = Doc.GetItemValue("I_CMD")
        MySendMessage.surname = Doc.GetItemValue("I_SURNAME")
        MySendMessage.firstname = Doc.GetItemValue("I_FIRST_NAME")
        MySendMessage.tel = Doc.GetItemValue("I_TEL")
        MySendMessage.zip = Doc.GetItemValue("I_ZIP")

        ' Send the message
        Call MyService.SendMessage ( MySendMessage, MySendOptions )
        Print SENDSUCCESS
        ' Generated MQ message Id is stored in document for delayed receive option
        Doc.MQRequestMsgId=MySendOptions.Identifier
    End If

Receive the message

Figure 43 (Part 6 of 11). Lotus Notes Client Agent for IMS via MQSeries Service

Lotus Notes and the MQSeries Enterprise Integrator
```
Print RECEIVED

' Identifier property must be set to receive the reply message to a given request message
' which is identified by the identifier stored in the MQRequestMsgId of the form
' otherwise MQEI would receive the first message in the queue
MyReceiveOptions.Identifier=Doc.GetItemValue("MQRequestMsgId")
Call MyService.ReceiveMessage ( MyReceiveMessage, MyReceiveOptions )

'******************************************************************************
'* Handle MQEI warning messages
'******************************************************************************

Doc.StatusFlag=SF_NONE
Doc.StatusDescription=SD_NONE
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
    Case EIRC_NO_MESSAGE_AVAILABLE :
        Messagebox EIRC_NO_MSG_AVAILABLE_STRING, MB_OK, WARNINGTITLE
        ' Set Status Flag, so that user can retry later (delayed receive)
        Doc.StatusFlag=SF_NO_MQREPLY_YET
        Doc.StatusDescription=SD_NO_MQREPLY_YET
    Case EIRC_FIELD_TRUNCATED :
        Messagebox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
    Case EIRC_WRONG_MESSAGE :
        Messagebox EIRC_WRONG_MESSAGE_STRING, MB_OK, WARNINGTITLE
    Case Else :
        Messagebox UNRECOGNISED_STRING & " " & MyService.ReasonCode, MB_OK, WARNINGTITLE
    End Select
    Call MyService.ClearErrorCodes
Else
Print RECEIVESUCCESS

'******************************************************************************
' Update backend document with receive message <<9>>
'******************************************************************************
Doc.O_CMD=Trim(MyReceiveMessage.cmd)
Doc.O_SURNAME=Trim(MyReceiveMessage.surname)
Doc.O_FIRST_NAME=Trim(MyReceiveMessage.firstname)
Doc.O_TEL=Trim(MyReceiveMessage.tel)
Doc.O_ZIP=Trim(MyReceiveMessage.zip)
Doc.O_MSG =Trim(MyReceiveMessage.msg)

Doc.StatusFlag=SF_PROCESSED ' Set status flag
End If

'******************************************************************************
' Redisplay Updated document <<10>>
'******************************************************************************
Call Doc.Save(True, True) ' Save backend document
Call Uidoc.reload ' Reloade UI document

Figure 43 (Part 7 of 11). Lotus Notes Client Agent for IMS via MQSeries Service
```
Call Uidoc.refresh
Call CleanUp ' Disconnect from enterprise services and delete all MQEI objects
Exit Sub

*****************************************************************************
** General LotusScript Error Handler
*****************************************************************************
GeneralErrorHandler:
Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
CleanUp
Exit Sub

*****************************************************************************
** Handle EILSX errors.
*****************************************************************************
MQEIServiceHandler:
Call GlobalErrors(EILSX_ERROR_TEXT)
CleanUp
Exit Sub
End Sub
Sub GlobalErrors(CallerText As String)

*****************************************************************************
** Output error handler messages in a formatted message box.
*****************************************************************************

End Sub
Sub EISessionErrorEventHandler(ThisSession As EISession , ReasonCode As Long)

*****************************************************************************
** Error event handler for the EISession Class. At this point we choose not to take
** any corrective action, an error message is simply displayed in a message box.
**
** Alternatively you could write a CASE statement to handle your chosen
** errors. Once handled the Error Codes can be cleared, allowing the
** main program to continue.
** Note: this subroutine is registered as an error event handler for EISession
** events but is also called following a failure in EISession creation in Queryopen
*****************************************************************************

Figure 43 (Part 8 of 11). Lotus Notes Client Agent for IMS via MQSeries Service

180 Lotus Notes and the MQSeries Enterprise Integrator
Dim StrText As String

Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
  Case EIRC_??? :
    ...Error Handle Process...
  Case EIRC_SEC_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " & Cstr(ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
    Messagebox StrText, MB_OK, TITLE_TEXT
  Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
    Messagebox StrText, MB_OK, TITLE_TEXT
  Case Else :
    Messagebox EISESSION_ERROR_TEXT + " : " + Cstr(ThisSession.ReasonCode) , MB_OK, TITLE_TEXT
End Select

'*******************************************************************************************
'  Do not clear the error code, as this will allow error EILSX_ERROR to be caught by the generic error handler.
*******************************************************************************************

End Select

Sub CleanUp

' This function is called when the agent is finished.
' Disconnect from MQSeries and delete the EI objects.
'******************************************************************************
On Error Resume Next ' Ignore any error as we need to shutdown regardless.

'******************************************************************************
' Disconnect from MQSeries.
'******************************************************************************
Print DISCONNECT_TEXT
Call MyService.Disconnect()

'******************************************************************************
' Delete the EI objects.
'******************************************************************************
Delete MyReceiveOptions
Delete MySendOptions
Delete MySendMessage
Delete MyReceiveMessage
Delete MyService

Figure 43 (Part 9 of 11). Lotus Notes Client Agent for IMS via MQSeries Service
End Sub
Sub EIServiceErrorHandler (thisService As EIService, ReasonCode As Long)
'******************************************************************************
'** Error Handler for the EIService Class. At this point we choose not to take
'** any corrective action, the error is simply displayed in a message box.
'**
'** Alternatively you could write a CASE statement to handle your chosen
'** errors. Once handled the Error Codes can be cleared, allowing the
'** main program to continue.
'******************************************************************************
Dim StrText As String
Select Case ReasonCode
  Case EIRC_???:
    "...Error Handle Process..."
    thisService.clearErrorCodes
  Case EIRC_???:
    "...Error Handle Process..."
    thisService.clearErrorCodes
  ...  ' Case EIRC_SERVICE_SYSERROR:
    ' SecondarySystemErrorCode and SystemErrorText are not set for this service
    StrText = THISSERVICE_ERROR + Cstr(ReasonCode) + Chr$(10)
    StrText = StrText + "Service Name : " + ThisService.Name + Chr$(10)
    Messagebox StrText, MB_OK, TITLE_TEXT
  Case Else :
    StrText = THISSERVICE_ERROR + Cstr(ReasonCode) + Chr$(10)
    StrText = StrText + "Service Name : " + ThisService.Name
    Messagebox StrText, MB_OK, TITLE_TEXT
End Select
End Sub
Sub EIMessageErrorHandler (thisMessage As EIMessage, ReasonCode As Long)
'******************************************************************************
'** Error Handler for the EIMessage Class. At this point we choose not to take
'** any corrective action, the error is simply displayed in a message box.
'**
'** Alternatively you could write a CASE statement to handle your chosen
'** errors. Once handled the Error Codes can be cleared, allowing the
'** main program to continue.
'******************************************************************************
Select Case ReasonCode
  Case EIRC_???:
    "...Error Handle Process..."
    thisMessage.clearErrorCodes
  Case EIRC_???:
    "...Error Handle Process..."
    thisMessage.clearErrorCodes
  ...  ' Case EIRC_SERVICE_SYSERROR:
    ' EIMessage_ERROR_TEXT and EIMessage_Name are not set for this service
    StrText = EIMESSAGE_ERROR_TEXT & Cstr(ReasonCode) & ". Message Name: " & & 
    thisMessage.Name & ", MB_OK, TITLE_TEXT
  Case Else:
    "...Error Handle Process..."
    thisMessage.clearErrorCodes
End Select
End Sub

Figure 43 (Part 10 of 11). Lotus Notes Client Agent for IMS via MQSeries Service

182  Lotus Notes and the MQSeries Enterprise Integrator
End Sub
Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long)
'*******************************************************************************************
'' Error Handler for the EIService Class. At this point we choose not to take
'' any corrective action, the error is simply displayed in a message box.
''
'' Alternatively you could write a CASE statement to handle your chosen
'' errors. Once handled the Error Codes can be cleared, allowing the
'' main program to continue.
'*******************************************************************************************
    Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
        thisSendOptions.clearErrorCodes
    Case EIRC_??? :
        ...Error Handle Process...
        thisSendOptions.clearErrorCodes
    Case Else :
        Messagebox EISO_ERROR_TEXT & " " & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
    End Select
End Sub

Sub EIReceiveOptionsErrorEventHandler (thisReceiveOptions As EIReceiveOptions, ReasonCode As Long)
'*******************************************************************************************
'' Error Handler for the EIService Class. At this point we choose not to take
'' any corrective action, the error is simply displayed in a message box.
''
'' Alternatively you could write a CASE statement to handle your chosen
'' errors. Once handled the Error Codes can be cleared, allowing the
'' main program to continue.
'*******************************************************************************************
    Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
        thisReceiveOptions.clearErrorCodes
    Case EIRC_??? :
        ...Error Handle Process...
        thisReceiveOptions.clearErrorCodes
    Case Else :
        Messagebox EIRO_ERROR_TEXT & " " & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
    End Select
End Sub

Figure 43 (Part 11 of 11). Lotus Notes Client Agent for IMS via MQSeries Service
B.6 Scheduled Lotus Notes Agent for Native MQSeries Service

'Scheduled Agent (Native MQSeries):

*******************************************************************************************
* SG24-2217 - Lotus Notes and the MQSeries Enterprise Integrator*
* ITSO Sample Application -
* Scheduled Agent for the MQEI Native MQSeries Service
*******************************************************************************************
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselsx "eilsx"

*******************************************************************************************
** Define Enterprise Integrator objects.
*******************************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendMessage As EIMessage
Dim MyReceiveMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EISendOptions

*******************************************************************************************
** Notes object variables for form use
*******************************************************************************************
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim dc As NotesDocumentCollection

*******************************************************************************************
** Wait time in milliseconds for a reply.
*******************************************************************************************
Const FIVE_SECONDS = 5000 ' 5 seconds

*******************************************************************************************
** To hold values expected from the EI Definition database
*******************************************************************************************
Const SERVICE_NAME = "MQServ"
Const SEND_MESSAGE_NAME = "VSAMSERVCOMMAREA"
Const RECEIVE_MESSAGE_NAME = "VSAMSERVCOMMAREA"

*******************************************************************************************
** Print text

Figure 44 (Part 1 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service
Const INIT_TEXT = "Initializing..."
Const SEND_TEXT = "Sending..."
Const RECEIVE_TEXT = "Receiving..."
Const CONSTRUCTING = "Constructing Message..."
Const SENDSUCCESS = "Send Message Completed Successfully"
Const RECEIVESUCCESS = "Receive Message Completed Successfully"
Const CONNECT_TEXT = "Connecting..."
Const DISCONNECT_TEXT = "Disconnecting..."
Const OK_TEXT = "Press 'OK' button to Send Message..."
Const BLANK = " "

'**********************************************************************************
Const SAMPLE_END_TEXT = "Sample completed."  
'**********************************************************************************

Figure 44 (Part 2 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service
Const EIRC_CONNECTED_STRING = "An attempt was made to connect " _+ "to a service when already connected. The connect is ignored and " _+ "the program continues."
Const EIRC_NO_MSG_AVAILABLE_STRING = "An attempt was made to " _+ "receive a message but the target message was not found. " _+ "Please, retry later."
Const EIRC_FIELD_TRUNCATED_STRING = "A length of a string value of " _+ "an EIMessage field exceeds the length specified on the MQSeries EI Definition " _+ "of " _+ "database. Correct the LotusScript to make sure that the string value " _+ "being passed as the EIMessage field value does not exceed the " _+ "predefined length."
Const EIRC_WRONG_MESSAGE_STRING = "A message was received that " _+ "does not match the EIMessage object passed as a parameter on the " _+ "EIService ReceiveMessage call."
Const UNRECOGNISED_STRING = "Unrecognized warning message :"
Const EISERVICE_WARNING = "Warning message received establishing " _+ "MQSeries. ReasonCode :"
Const EISSESSION_ERROR = "An error has been reported from the " _+ "EISession Class. ReasonCode"
Const THISMESSAGE_ERROR = "An error has been reported from the " _+ "EIMessage Class. ReasonCode :":
Const THISSERVICE_ERROR = "An error has been reported from the " _+ "EIService Class. ReasonCode :":
Const THISSO_ERROR = "An error has been reported from the " _+ "EISendOptions Class. ReasonCode :":
Const THISSO_ERROR = "An error has been reported from the " _+ "EIReceiveOptions Class. ReasonCode :":

Sub Initialize

Figure 44 (Part 3 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service
program VSAMSERV to process the message.
* VSAMSERV then returns to AMQCIC1 which puts a reply back to a queue.
* The LotusScript application gets the message and updates the document.
* 
* NOTE: When you see a number enclosed in '<<' and '>>' characters, (eg. <<1>>)
* it is referring to a step described in the
* chapter 5.1.3 Implementation of the Lotus Notes Client Access (Two Tier Model)
* in the ITSO Redbook:

*******************************************************************************************

On Error Goto GeneralErrorHandler
On Error EILSX_ERROR Goto MQEIErrorHandler

*******************************************************************************************

* Initialize the EISession object <<4>>

*******************************************************************************************

Set MySession = New EISession

Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
    Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
    Call MySession.ClearErrorCodes()
End If

*******************************************************************************************

* Create an EI Service
* <<4a>> defined in the EI Definition database).

Set MyService = MySession.CreateService(SERVICE_NAME)
On Event EIError From MyService Call EIServiceErrorEventHandler

*******************************************************************************************

* Create EI send message
* <<4a>> defined in the EI Definition database).

Set MySendMessage = MySession.CreateMessage(SEND_MESSAGE_NAME)
On Event EIError From MySendMessage Call EIMessageErrorEventHandler

*******************************************************************************************

* Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MySendMessage.CompletionCode = EICC_WARNING Then
    Select Case MySendMessage.ReasonCode
    Case EIRC_FIELD_TRUNCATED :
        Messagebox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
    Case Else :
        Messagebox UNRECOGNISED_STRING & " " & Cstr(MySendMessage.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MySendMessage.ClearErrorCodes

Figure 44 (Part 4 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service
End If

'******************************************************************************************
'* Create EI receive message
'******************************************************************************************
Set MyReceiveMessage = MySession.CreateMessage(RECEIVE_MESSAGE_NAME)
On Event EIError From MyReceiveMessage Call EIMessageErrorEventHandler

If MyReceiveMessage.CompletionCode = EICC_WARNING Then
    Select Case MyReceiveMessage.ReasonCode
        Case EIRC_FIELD_TRUNCATED :
            Messagebox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
        Case Else :
            Messagebox UNRECOGNISED_STRING & " & Cstr(MyReceiveMessage.ReasonCode), MB_OK, WARNINGTITLE
    End Select
    Call MyReceiveMessage.ClearErrorCodes
End If

'******************************************************************************************
'* Create EI send message options
'******************************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler

'Option must be set to make MQEI define the MQ message type 'request' in the MQ message descriptor
MySendOptions.MessageType=EIMT_REQUEST

'******************************************************************************************
'* Create EI receive message options
'******************************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler

MyReceiveOptions.WaitType=EIWT_WAIT
MyReceiveOptions.WaitInterval=FIVE_SECONDS

'******************************************************************************************
'* Set document variables
'******************************************************************************************
Set Session = New NotesSession
Set dc=session.currentdatabase.unprocesseddocuments
Set doc=dc.getfirstdocument

'******************************************************************************************
'* Connect to EI Service
'******************************************************************************************
Print CONNECT_TEXT
Call MyService.Connect ' Connect to enterprise services

If MySession.CompletionCode = EICC_WARNING Then

Figure 44 (Part 5 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service
Select Case MySession.ReasonCode
Case EIRC_CONNECTED:
  Messagebox EIRC_CONNECTED_STRING, MB_OK, WARNINGTITLE
Case Else:
  Messagebox EISERVICE_WARNING & " " & Cstr(MySession.ReasonCode), MB_OK, WARNINGTITLE
End Select
Call MySession.ClearErrorCodes
End If

'************************************************************************************************
'******* Send the Message to the EI Service selected if StatusFlag is not
'******* SF_NO_MQ_REPLY_MSG_YET
'******* otherwise message has been sent already and just a receive has to take
'******* place
'************************************************************************************************
Do Until Doc Is Nothing ' Start the loop for each document
If Doc.GetFirstItem("StatusFlag").Text<>SF_NO_MQ_REPLY_YET Then
  Print SEND_TEXT
  '*****************************************************************************
  '  Setup and send message to CICS
  '*****************************************************************************
  MySendMessage.SERVER_STATE = Doc.GetItemValue("I_SERVER_STATE")
  MySendMessage.SERVER_KEY_SIZE = Cint(Doc.GetItemValue("I_SERVER_KEY_SIZE"))
  MySendMessage.SURNAME = Doc.GetItemValue("I_SURNAME")
  MySendMessage.FIRST_NAME = Doc.GetItemValue("I_FIRST_NAME")
  MySendMessage.ACCOUNT = Doc.GetItemValue("I_ACCOUNT")
  MySendMessage.ADDRESS = Doc.GetItemValue("I_ADDRESS")
  MySendMessage.CITY_STATE = Doc.GetItemValue("I_CITY_STATE")
  MySendMessage.POSTAL_CODE = Doc.GetItemValue("I_POSTAL_CODE")
  MySendMessage.BALANCE = Cint(Doc.GetItemValue("I_BALANCE"))
  '*****************************************************************************
  '  Send the message
  '*****************************************************************************
  Call MyService.SendMessage ( MySendMessage, MySendOptions )
  Print SENDSUCCESS
  '*****************************************************************************
  '  Generated MQ message Id is stored in document for delayed
  '  receive option
  '*****************************************************************************
  Doc.MQRequestMsgId=MySendOptions.Identifier
End If
'************************************************************************************************
'****** Receive the message
'************************************************************************************************
Print RECEIVE_TEXT
'Identifier property must be set to receive the reply message to a given request message
which is identified by the identifier stored in the MQRequestMsgId of the form otherwise MQEI would receive the first message in the queue
MyReceiveOptions.Identifier=Doc.GetItemValue("MQRequestMsgId")(0)
Call MyService.ReceiveMessage (MyReceiveMessage, MyReceiveOptions)

************************************************************************************************
* Handle MQEI warning messages
************************************************************************************************

Doc.StatusFlag=SF_NONE
Doc.StatusDescription=SD_NONE
If MyService.CompletionCode = EICC_WARNING Then
Select Case MyService.ReasonCode
Case EIRC_NO_MESSAGE_AVAILABLE :
Messagebox EIRC_NO_MSG_AVAILABLE_STRING, MB_OK, WARNINGTITLE
' Set Status Flag, so that user can retry later (delayed receive)
Doc.StatusFlag=SF_NO_MQREPLY_YET
Doc.StatusDescription=SD_NO_MQREPLY_YET
CleanUp
Call Doc.Save(True, True)
Exit Sub
Case EIRC_FIELD_TRUNCATED :
Messagebox EIRC_FIELD_TRUNCATED_STRING, MB_OK, WARNINGTITLE
Case EIRC_WRONG_MESSAGE :
Messagebox EIRC_WRONG_MESSAGE_STRING, MB_OK, WARNINGTITLE
Case Else :
Messagebox UNRECOGNISED_STRING & " " & MyService.ReasonCode, MB_OK, WARNINGTITLE
End Select
Call MyService.ClearErrorCodes
Else
Print RECEIVESUCCESS
************************************************************************************************
' Update backend document with receive message <<9>>
************************************************************************************************

Doc.I_SERVER_STATE=Trim(MyReceiveMessage.SERVER_STATE)
Doc.I_SERVER_KEY_SIZE=Cstr(MyReceiveMessage.SERVER_KEY_SIZE)
Doc.I_SUBNAME=Trim(MyReceiveMessage.SUBNAME)
Doc.I_FIRST_NAME=Trim(MyReceiveMessage.FIRST_NAME)
Doc.I_ACCOUNT=Trim(MyReceiveMessage.ACCOUNT)
Doc.I_ADDRESS=Trim(MyReceiveMessage.ADDRESS)
Doc.I_CITY_STATE =Trim(MyReceiveMessage.CITY_STATE)
Doc.I_POSTAL_CODE =Trim(MyReceiveMessage.POSTAL_CODE)
Doc.I_BALANCE=Cstr(MyReceiveMessage.BALANCE)
Doc.O_RESPONSES=Trim(MyReceiveMessage.RESPONSES)

Doc.StatusFlag=SF_PROCESSED

End If
************************************************************************************************
' Save Updated document <<10>>
************************************************************************************************

Figure 44 (Part 7 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service

190 Lotus Notes and the MQSeries Enterprise Integrator
Call Doc.Save(True, True) ' ' Save backend document  
Set doc=dc.getnextdocument(doc) ' Process the next document  
Loop  
CleanUp  ' ' Disconnect from enterprise services and delete all MQEI objects  
Exit Sub  

******************************************************************************  
'* General LotusScript Error Handler  
******************************************************************************  
GeneralErrorHandler:  
Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)  
CleanUp  
Exit Sub  

******************************************************************************  
'* Handle EILSX errors.  
******************************************************************************  
MQEIErrorHandler:  
Call GlobalErrors(EILSX_ERROR_TEXT)  
CleanUp  
Exit Sub  
End Sub  

Sub EIServiceErrorEventHandler (thisService As EIService, ReasonCode As Long)  
******************************************************************************  
'* Error Handler for the EIService Class. At this point we choose not to take  
'* any corrective action, the error is simply displayed in a message box.  
'*  
'* Alternatively you could write a CASE statement to handle your chosen  
'* errors. Once handled the Error Codes can be cleared, allowing the  
'* main program to continue.  
******************************************************************************  
Select Case ReasonCode  
' Case EIRC_???:  
'  ...Error Handle Process...  
'  thisService.clearErrorCodes  
' Case EIRC_???:  
'  ...Error Handle Process...  
'  thisService.clearErrorCodes  
'  ...  
Case Else :  
  MessageBox EISERVICE_ERROR_TEXT & " " & Cstr(ReasonCode) & ". Service Name: " & _  
    thisService.Name & ", MB_OK, TITLE_TEXT  
End Select  
End Sub  

Sub EIMessageErrorEventHandler (thisMessage As EIMessage, ReasonCode As Long)  
******************************************************************************  
'* Error Handler for the EIService Class. At this point we choose not to take  
'* any corrective action, the error is simply displayed in a message box.  
******************************************************************************  
Figure 44 (Part 8 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service
Alternatively you could write a CASE statement to handle your chosen errors. Once handled the Error Codes can be cleared, allowing the main program to continue.

```vbscript
Select Case ReasonCode
    Case EIRC_??? : ...Error Handle Process...
        thisMessage.clearErrorCodes
    Case EIRC_??? : ...Error Handle Process...
        thisMessage.clearErrorCodes
    ...  
    Case Else:
        MessageBox EIMESSAGE_ERROR_TEXT & " " & Cstr(ReasonCode) & ". Message Name: " & _
        thisMessage.Name & ".", MB_OK, TITLE_TEXT
End Select
End Sub
```

Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long)
    ' Error Handler for the EISendOptions Class. At this point we choose not to take any corrective action, the error is simply displayed in a message box.
    ' Alternatively you could write a CASE statement to handle your chosen errors. Once handled the Error Codes can be cleared, allowing the main program to continue.

```vbscript
Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
        thisSendOptions.clearErrorCodes
    Case EIRC_??? :
        ...Error Handle Process...
        thisSendOptions.clearErrorCodes
    ...
    Case Else:  
        MessageBox EISO_ERROR_TEXT & " " & Cstr(ReasonCode) & ".", MB_OK, TITLE_TEXT
End Select
End Sub
```

Sub EIReceiveOptionsErrorEventHandler (thisReceiveOptions As EIReceiveOptions, ReasonCode As Long)
    ' Error Handler for the EIReceiveOptions Class. At this point we choose not to take any corrective action, the error is simply displayed in a message box.
    ' Alternatively you could write a CASE statement to handle your chosen errors. Once handled the Error Codes can be cleared, allowing the main program to continue.

```vbscript
Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
        thisReceiveOptions.clearErrorCodes
    Case EIRC_??? :
        ...Error Handle Process...
```

Figure 44 (Part 9 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service
Sub GlobalErrors(CallerText As String)
'*******************************************************************************************
'* Output error handler messages in a formatted message box.
'********************************************************************************************
    Dim ErrorText As String

    Print CallerText

    ' Format the error text for the message box
    ErrorText = CallerText + Chr$(10)
    ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
    ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$ 
    ErrorText = ErrorText + Chr$(10) + LINE_NUMBER_TEXT + " = " + Str(Erl) 

    ' Output the error to message box and print line
    MsgBox ErrorText, MB_OK, TITLE_TEXT
    Print ""
End Sub

Sub EISessionErrorEventHandler(ThisSession As EISession , ReasonCode As Long)
'*******************************************************************************************
'* Error event handler for the EISession Class. At this point we choose not to take
'* any corrective action, an error message is simply displayed in a message box.
'*
'* Alternatively you could write a CASE statement to handle your chosen
'* errors. Once handled the Error Codes can be cleared, allowing the
'* main program to continue.
'* Note: this subroutine is registered as an error event handler for EISession
'* events but is also called following a failure in EISession creation in Queryopen
'********************************************************************************************
    Dim StrText As String

    Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
    Case EIRC_??? :
        ...Error Handle Process...
    ... 
    Case EIRC_SEC_DB_SYSERROR:
        StrText = EISSESSION_ERROR_TEXT & " & Cstr(ReasonCode) & "." + Chr$(10)
        StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
        StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
        StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
        MsgBox StrText, MB_OK, TITLE_TEXT
    End Select
End Sub

Figure 44 (Part 10 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service

Appendix B. Lotus Notes Agents 193
Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
    Messagebox StrText, MB_OK, TITLE_TEXT
Case Else :
    Messagebox EISESSION_ERROR_TEXT + " : " + Cstr(ThisSession.ReasonCode) , MB_OK, TITLE_TEXT
End Select
******************************************************************************
'* Do not clear the error code, as this will allow error EILSX_ERROR to be
'* caught by the generic error handler.
******************************************************************************
End Sub
Sub CleanUp
******************************************************************************
'* This function is called when the agent is finished.
'* Disconnect from MQSeries and delete the EI objects.
******************************************************************************
    On Error Resume Next ' Ignore any error as we need to shutdown regardless.
    ' Disconnection from MQSeries.
    Print DISCONNECT_TEXT
    Call MyService.Disconnect()
******************************************************************************
'* Delete the EI objects.
******************************************************************************
    Delete MyReceiveOptions
    Delete MySendOptions
    Delete MySendMessage
    Delete MyReceiveMessage
    Delete MyService
    Delete MySession
End Sub

Figure 44 (Part 11 of 11). Scheduled Lotus Notes Agent for Native MQSeries Service

194 Lotus Notes and the MQSeries Enterprise Integrator
Appendix C. Web Agents

In this appendix we list the code used for the Web agents.

C.1 Web Client Agent for CICS DPL Direct Service

```vbnet
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselex "eilsx"

' Define Enterprise Integrator objects.
Dim MySession As EISession
Dim MyService As EIService
Dim MyMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

' Notes variables for form manipulation.
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim MyDb As Notesdatabase
Dim ServName As NotesName

' Wait time in milliseconds for a reply.
Const FIVE_SECONDS = 5000  ' 5 seconds
```

Figure 45 (Part 1 of 10). Web Client Agent for CICS DPL Direct Service
Figure 45 (Part 2 of 10). Web Client Agent for CICS DPL Direct Service
Const EIRC_WRONG_MESSAGE_STRING = "A message was received that does not match the EI ne Message object passed as a parameter on the " + "EIReceiveMessage call."
Const UNRECOGNISED_STRING = "Unrecognized warning message :"
Const EISERVICE_WARNING = "Warning message received establishing " + "MQService. ReasonCode :
Const EISESSION_ERROR = "An error has been reported from the " + "EISession Class. ReasonCode"
Const THISMESSAGE_ERROR = "An error has been reported from the " + "EIMessage Class. ReasonCode :"
Const THISSERVICE_ERROR = "An error has been reported from the " + "EIService Class. ReasonCode :
Const THISSO_ERROR = "An error has been reported from the " + "EISendMessageOptions Class. ReasonCode :
Const THISRO_ERROR = "An error has been reported from the " + "EIReceiveOptions Class. ReasonCode :

'**************************************************************************
'* Definition of StatusFlag Values
**************************************************************************'
Const SF_PROCESSED="Processed"

Sub Initialize
'**************************************************************************
'* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator"
'* ITSO Sample Application -
'* Web Agent for the MQEI CICS DPL Direct Service
**************************************************************************
' This sample illustrates how a Lotus Script application can execute
' a CICS DPL ECI program.
' The CICS program is called VSAMSERV and is a credit information sample
' application.
' This agent is activated when the "CICS DPL ECI" Service Selection radio
' button is selected and the "Submit" button is clicked.
' NOT: When you see a number enclosed in '<<' and '>>' characters, (eg. <<1>>) ' is refering to a step in the Web User Access (Three Tier Model) diagram
' in the ITSO Redbook "Lotus Notes and the MQSeries Enterprise Integrator"
**************************************************************************

'**************************************************************************
'* Initialize the EILSX objects. <<>>
**************************************************************************
'**************************************************************************
'* Set up the error handlers (Note: These are not event handlers).
**************************************************************************
On Error Goto GeneralErrorHandler ' Handle general LotusScript errors
On Error EILSX_ERROR Goto MQEIErrrorHandler ' EILSX errors

Figure 45 (Part 3 of 10). Web Client Agent for CICS DPL Direct Service
**Initialize the EISession object <<4>>**

Set MySession = New EISession

' Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
  Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
  Call MySession.ClearErrorCodes()
End If

'' Create an EI Service
'' <<4a>> defined in the EI Definition database).
Set MyService = MySession.CreateService(DBSERVICE)
On Event EIError From MyService Call EIServiceErrorEventHandler

'' Create an EI Message
'' <<4a>> defined in the EI Definition database).
Set MyMessage = MySession.CreateMessage(DBMESSAGE)
On Event EIError From MyMessage Call EIMessageErrorEventHandler

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyMessage.CompletionCode = EICC_WARNING Then
  Select Case MyMessage.ReasonCode
    Case EIRC_FIELD_TRUNCATED:
      MsgBox EIRC_FIELD_TRUNCATED_TEXT, MB_OK, TITLE_TEXT
    Case Else :
      Messagebox UNRECOGNISED_TEXT & " & Cstr(MyMessage.ReasonCode), MB_OK, TITLE_TEXT
  End Select
  Call MyMessage.ClearErrorCodes()
End If

'' Create EI Send Options.
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler

MySendOptions.MessageType = EIMT_REQUEST  ' Messages are requests

'' Create EI Receive Options.
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EISendOptionsErrorEventHandler

MyReceiveOptions.WaitInterval = FIVE_SECONDS  ' Set wait interval of 5 seconds

Figure 45 (Part 4 of 10). Web Client Agent for CICS DPL Direct Service
MyReceiveOptions.WaitType = EIWT_WAIT ' Set wait type to use specified timeout

'*******************************************************************************
' Set document variables
'*******************************************************************************
Set Session = New NotesSession
Set Doc = Session.DocumentContext
Set MyDb = Session.CurrentDatabase
Set MyName = New NotesName(session.userName)
'*******************************************************************************
' Connect to enterprise services <<5>>
'*******************************************************************************
Call MyService.Connect() ' Connect to enterprise services

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
  Select Case MyService.ReasonCode
  Case EIRC_CONNECTED: ' Already connected (probably from previous run)
    MessageBox EIRC_CONNECTED_TEXT, MB_OK, TITLE_TEXT
  Case Else:
    MessageBox EISERVICE_WARNING_TEXT & " " & Cstr(MyService.ReasonCode) & ",", MB_OK, TITLE_TEXT
  End Select
  Call MyService.ClearErrorCodes()
End If

'*******************************************************************************
' Setup and send message to CICS <<6>>
'*******************************************************************************
MySendOptions.UnitOfWork = EIUOW_ONLY ' EC01 operates in its own unit of work
MyMessage.SERVER_STATE = Doc.GetItemValue("I_SERVER_STATE")(0)
MyMessage.SERVER_KEY_SIZE = CInt(Doc.GetItemValue("I_SERVER_KEY_SIZE")(0))
MyMessage.SURNAME = ...
MyMessage.BALANCE = CInt(Doc.GetItemValue("I_BALANCE")(0))

'*******************************************************************************
'Send the message <<7>>
'*******************************************************************************
Call MyService.SendMessage (MyMessage, MySendOptions)

'*******************************************************************************
' Get reply from CICS. <<8>>
'*******************************************************************************
MyReceiveOptions.Identifier = MySendOptions.Identifier 'Only read messages that belong to us
Call MyService.ReceiveMessage(MyMessage, MyReceiveOptions)

Figure 45 (Part 5 of 10). Web Client Agent for CICS DPL Direct Service

Appendix C. Web Agents 201
Update backend document with receive message

Redisplay Updated document

Call Doc.Save(True, True) ' Save backend document

Call CleanUp ' Disconnect from enterprise servers and delete all MQEI objects

Exit Sub

General LotusScript Error Handler

GeneralErrorHandler:
Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
Call CleanUp
Exit Sub

Handle EILSX errors.

MQEIErrorHandler:
Call GlobalErrors(EILSX_ERROR_TEXT)
Call CleanUp
Exit Sub

End Sub

Sub EIIServiceErrorEventHandler (thisService As EIIService, ReasonCode As Long)

Error Handler for the EIIService Class. At this point we choose not to take
any corrective action, the error is simply displayed in a message box.

Alternatively you could write a CASE statement to handle your chosen
errors. Once handled the Error Codes can be cleared, allowing the
main program to continue.

Figure 45 (Part 6 of 10). Web Client Agent for CICS DPL Direct Service
Dim StrText As String
Select Case ReasonCode
  Case EIRC_???:
    ...Error Handle Process...
    thisService.clearErrorCodes
  Case EIRC_???:
    ...Error Handle Process...
    thisService.clearErrorCodes
  ...
Case EIRC_SERVICE_SYSERROR:
  ' SecondarySystemErrorCode and SystemErrorText are not set for this service
  StrText = THISSERVICE_ERROR + " + Cstr(ReasonCode) + Chr$(10)
  StrText = StrText + "Service Name : " + ThisService.Name + Chr$(10)
  StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisService.PrimarySystemErrorCode)
  Print StrText  TITLE_TEXT
Case Else :
  StrText = THISSERVICE_ERROR + " + Cstr(ReasonCode) + Chr$(10)
  StrText = StrText + "Service Name : " + ThisService.Name
  Print StrText  TITLE_TEXT
End Select
End Sub

Sub EIMessageErrorEventHandler (thisMessage As EIMessage, ReasonCode As Long)
'******************************************************
' Error Handler for the EIService Class. At this point we choose not to take
' any corrective action, the error is simply displayed in a message box.
' *
' Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
' main program to continue.
'******************************************************
Select Case ReasonCode
  Case EIRC_???: ...Error Handle Process...
    thisMessage.clearErrorCodes
  Case EIRC_???: ...Error Handle Process...
    thisMessage.clearErrorCodes
  ...
Case Else :
  Print EIMESSAGE_ERROR_TEXT & " & Cstr(ReasonCode) &
  ". Message Name: " & thisMessage.Name & ".", MB_OK, TITLE_TEXT
End Select
End Sub

Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long)
'******************************************************
' Error Handler for the EIService Class. At this point we choose not to take
' any corrective action, the error is simply displayed in a message box.
' *
' Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
' main program to continue.
'******************************************************
Select Case ReasonCode
End Sub

Figure 45 (Part 7 of 10). Web Client Agent for CICS DPL Direct Service
Sub EIReceiveOptionsErrorEventHandler (thisReceiveOptions As EIReceiveOptions, ReasonCode As Long)

"* Error Handler for the EIService Class. At this point we choose not to take
"* any corrective action, the error is simply displayed in a message box.
**
"* Alternatively you could write a CASE statement to handle your chosen
"* errors. Once handled the Error Codes can be cleared, allowing the
"* main program to continue.
*******************************************************************************************
Select Case ReasonCode
" Case EIRC_??? :
...Error Handle Process...
thisReceiveOptions.clearErrorCodes
" Case EIRC_??? :
...Error Handle Process...
thisReceiveOptions.clearErrorCodes
... 
Case Else :
Print EISO_ERROR_TEXT & "& Cstr(ReasonCode) & "." & TITLE_TEXT
End Select
End Sub
Sub GlobalErrors (CallerText As String)
*******************************************************************************************
"* Output error handler messages in a formatted message box.
*******************************************************************************************
Dim ErrorText As String

' Format the error text for the message box
ErrorText = CallerText + Chr$(10)
ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$
ErrorText = ErrorText + Chr$(10) + LINE_NUMBER_TEXT + " = " + Str(Erl)

' Output the error to message box and print line
Print ErrorText & TITLE_TEXT
End Sub
Sub Cleanup
*******************************************************************************************
"* This function is called when the agent is finished.
"* Disconnect from MQSeries and delete the EI objects.
*******************************************************************************************

Figure 45 (Part 8 of 10). Web Client Agent for CICS DPL Direct Service
On Error Resume Next ' Ignore any error as we need to shutdown regardless.

'*****************************************************************************
'** Disconnect from MQSeries.
'*****************************************************************************
Call MyService.Disconnect()

'*****************************************************************************
'** Delete the EI objects.
'*****************************************************************************
Delete MyReceiveOptions
Delete MySendOptions
Delete MyMessage
Delete MyService
Delete MySession

End Sub

Sub EISessionErrorEventHandler(ThisSession As EISession, ReasonCode As Long)
'*****************************************************************************
'** Error event handler for the EISession Class. At this point we choose not to take
'** any corrective action, an error message is simply displayed in a message box.
'**
'** Alternatively you could write a CASE statement to handle your chosen
'** errors. Once handled the Error Codes can be cleared, allowing the
'** main program to continue.
'** Note: this subroutine is registered as an error event handler for EISession
'** events but is also called following a failure in EISession creation in Queryopen
'*****************************************************************************

Dim StrText As String

Select Case ReasonCode
  ' Case EIRC_??? :
  ' ...Error Handle Process...
  Case EIRC_??? :
    ' ...Error Handle Process...
  Case EIRC_SEC_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " " & Cstr(ReasonCode) & "." & Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
    Print StrText TITLE_TEXT
  Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & " " & Cstr(ThisSession.ReasonCode) & "." & Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
    Print StrText TITLE_TEXT
  Case Else :
    StrText = EISESSION_ERROR_TEXT + " : " + Cstr(ThisSession.ReasonCode)

Figure 45 (Part 9 of 10). Web Client Agent for CICS DPL Direct Service
**Figure 45 (Part 10 of 10). Web Client Agent for CICS DPL Direct Service**
C.2 Web Client Agent for CICS 3270 Direct Service

*Web Agent (CICS 3270 EPI):

*****************************************************************************
* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator*
* ITSO Sample Application  -  Web Agent for the MQEI CICS 3270 EPI Service*
*****************************************************************************
Option Public
Option Explicit

* Include Notes constants file (defines MB_OK etc)*
%INCLUDE "lsconst.lss"

* Load the MQSeries Enterprise Integrator library*
Uselex "eilsx"

*****************************************************************************
* Define Enterprise Integrator objects.\n*****************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

*****************************************************************************
* Define Enterprise Messages\n*****************************************************************************
Dim CESNMessage As EIMessage  'CESN Screen
Dim ML01Message As EIMessage  'Message for ClearScreen
Dim VAR1Message As EIMessage  'Variant Message for TECH
Dim MyMessage As EIMessage  'Generic Message
Dim T001Message As EIMessage  'Screen T001
Dim T002Message As EIMessage
Dim T003Message As EIMessage
Dim T004Message As EIMessage
Dim T005Message As EIMessage
Dim T006Message As EIMessage
Dim T007Message As EIMessage

*****************************************************************************
* Notes variables for form manipulation.\n*****************************************************************************
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim MyDb As NotesDatabase
Dim ServName As NotesName

*****************************************************************************

Figure 46 (Part 1 of 15). Web Client Agent for CICS 3270 Direct Service
"* Message, Service and file name of sample text.
*****************************************************************************
Const SERVICENAME = "CICEPSICESN"
Dim ReceiveType As Long
*****************************************************************************
"* Translatable text strings used throughout sample
*****************************************************************************
Const TITLE_TEXT = "CICS 3270 Direct sample"
Const LOTUSSCRIPT_ERROR_TEXT = "LotusScript error!"
Const EILSX_ERROR_TEXT = "EILSX error!"
Const ERROR_NUMBER_TEXT = "Error number"
Const ERROR_DESCRIPTION_TEXT = "Error description"
Const LINE_NUMBER_TEXT = "Line number"
Const EIRC_CONNECTED_TEXT = "An attempt to connect to a service has been rejected because the service" + " is already connected. The connect is ignored and the program continues."
Const RECEIVE_FAILED_TEXT = "Unable to receive message, ending sample"
Const EIRC_NO_MESSAGE_AVAILABLE_TEXT = "No message available - Retry operation?"
Const EIRC_FIELD_TRUNCATED_TEXT = "The program has set a field in an EIMessage where the field is defined" + " to be shorter than the length of data supplied. Alternatively, the program has got the data from an" + " EIMessage where the data in the field is too long."
Const EIRC_WRONG_MESSAGE_TEXT = "An EIMessage does not match the EIMessage object passed" + " as the parameter on the EIService ReceiveMessage call."
Const UNRECOGNISED_TEXT = "Unrecognised warning message:"
Const EISERVICE_WARNING_TEXT = "Warning message received establishing EIService. ReasonCode: "
Const EISESSION_ERROR_TEXT = "An error has been reported from the EIService Class. ReasonCode: "
Const EIMESSAGE_ERROR_TEXT = "An error has been reported from the EIMessage Class. ReasonCode: "
Const EISERVICE_ERROR_TEXT = "An error has been reported from the EIService Class. ReasonCode: "
Const EISO_ERROR_TEXT = "An error has been reported from the EISendOptions Class. ReasonCode: "
Const EIRO_ERROR_TEXT = "An error has been reported from the EIReceiveOptions Class. ReasonCode: "
Const PRIMARYSYSTEM = "Primary System Error Code"
Const SECONDARYSYSTEM = "Secondary System Error Code"
Const SYSTEMERRORTEXT = "Description"
Const GET_REPLY_ERROR_TEXT = "An error has been reported from ReceiveMessage. ReasonCode:"
Const SAMPLE_ABEND_TEXT = "Sample terminated abnormally"
Const SYSTEM_INVALID = "A CICS system must be selected"
Const USERID_TOO_LONG = "Userid is a maximum of 8 characters"
Const USERID_REQUIRED = "Userid not stored in security database so must be entered"
Const PASSWORD_TOO_LONG = "Password is a maximum of 8 characters"
Const PASSWORD_REQUIRED = "Password not stored in security database so must be entered"
Const NEWPASSWORD_TOO_LONG = "New password is a maximum of 8 characters"
Const NEWPASSWORD_REQUIRED = "New password must be entered to change the password"
Const VERIFYPASSWORD_MISMATCH = "Verify Password is not the same as New Password. Please re-enter" + " New and Verify passwords"
Const INIT1_TEXT = "Initializing session..."
Const INIT2_TEXT = "Initializing general messages..."
Const GO_TEXT = "Ready to go..."
Const SYS1_TEXT = "Creating service..."
Const SYS2_TEXT = "Creating TECH message..."
Const SEND_CLEAR = "Sending CLEAR..."
Const SEND_F3 = "Sending F3..."

Figure 46 (Part 2 of 15). Web Client Agent for CICS 3270 Direct Service

208 Lotus Notes and the MQSeries Enterprise Integrator
Const SEND_F12 = "Sending F12..."
Const SEND_TECH = "Sending TECH..."
Const SEND_CESN = "Sending CESN..."
Const SEND_TEXT = "Sending data..."
Const RECEIVE_TEXT = "Receiving..."
Const CONNECT_TEXT = "Connecting..."
Const DISCONNECT_TEXT = "Disconnecting..."
Const DELETE_TEXT = "Deleting..."

'******************************************************************************************
' * Definition of StatusFlag Values
'******************************************************************************************
Const SF_PROCESSED="Processed"

Sub GlobalErrors(CallerText As String)
    ' Output error handler messages in a formatted message box.
    '*******************************************************************************************
    Dim ErrorText As String
    ErrorText = CallerText + Chr$(10)
    ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
    ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$
    Print ErrorText TITLE_TEXT
End Sub

Sub EIServiceErrorEventHandler (ThisService As EIService, ReasonCode As Long)
    '*******************************************************************************************
    Print "Routine EIService"
    Select Case ReasonCode
        Case EIRC_SERVICE_SYSERROR:
            StrText = EISERVICE_ERROR_TEXT + " + Cstr(ReasonCode) + Chr$(10)
            StrText = StrText + "Service Name : " + ThisService.Name + Chr$(10)
            StrText = StrText + SYSTEMERRORTEXT + " + Cstr(ThisService.SystemErrorText) + Chr$(10)
            StrText = StrText + PRIMARYSYSTEM + " + Cstr(ThisService.PrimarySystemErrorCode) + Chr$(10)
            StrText = StrText + SECONDARYSYSTEM + " + Cstr(ThisService.SecondarySystemErrorCode)
            Print StrText TITLE_TEXT
        Case Else :
            StrText = EISERVICE_ERROR_TEXT + " + Cstr(ReasonCode) + Chr$(10)
            StrText = StrText + "Service Name : " + ThisService.Name
    End Select
End Sub

Figure 46 (Part 3 of 15). Web Client Agent for CICS 3270 Direct Service
Sub EISessionErrorEventHandler(ThisSession As EISession, ReasonCode As Long)
'*******************************************************************************************'* Error event handler for the EISession Class. At this point we choose not to take
'** any corrective action, an error message is simply displayed in a message box.
'**
'** Alternatively you could write a CASE statement to handle your chosen
'** errors. Once handled the Error Codes can be cleared, allowing the
'** main program to continue.
'** Note: this subroutine is registered as an error event handler for EISession
'** events but is also called following a failure in EISession creation in Queryopen
'*******************************************************************************************
Dim StrText As String
Print "Routine EISession"
Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
  Case EIRC_??? :
    ...Error Handle Process...
  Case EIRC_SEC_DB_SYSERROR:
    StrText = EISSESSION_ERROR_TEXT & " " & Cstr(ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + 
              + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + 
              + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
  Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISSESSION_ERROR_TEXT & " " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + 
              + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + 
              + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
  Case Else :
    StrText = EISSESSION_ERROR_TEXT & " " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)
End Select
Print StrText
'*******************************************************************************************'* Do not clear the error code, as this will allow error EILSX_ERROR to be
'** caught by the generic error handler.
'*******************************************************************************************
End Sub

Sub EISessionCleanup()
'*******************************************************************************************'* This function is called when the agent is finished.
*******************************************************************************************
Figure 46 (Part 4 of 15). Web Client Agent for CICS 3270 Direct Service
'** Disconnect from MQSeries and delete the EI objects.
****************************************************************************
On Error Resume Next ' Ignore any error as we need to shutdown regardless.
'******************************************************************************
'** Disconnect from MQSeries.
******************************************************************************
Call MyService.Disconnect()

'******************************************************************************
'** Delete the EI objects.
******************************************************************************
Delete CESPMessage
Delete ML01Message
Delete T001Message
Delete T002Message
Delete T003Message
Delete T004Message
Delete T005Message
Delete T006Message
Delete T007Message
Delete MyReceiveOptions
Delete MySendOptions
Delete MyService
Delete MySession
End Sub

Sub Initialize
'*******************************************************************************************
'** SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator*
'** ITSO Sample Application - Web Agent for the MQEI CICS 3270 EPI Service
'******************************************************************************
'** This sample illustrates how a Lotus Script application can execute
'** a CICS 3270 EPI program.
'** The Lotus Notes application sends a 3270 datastream to CICS transaction TECH.
'** CICS transaction TECH invokes program TECHPROG.
'** Program TECHPROG calls program VSAMSERV with a COMMAREA.
'** CICS program VSAMSERV is a credit information sample application
'**
'** This agent is activated when the "CICS 3270 EPI* Service Selection radio
'** button is selected and the "Submit" button is clicked.
'**
'** NOTE: When you see a number enclosed in '<<' and '>>' characters, {eg. <<1>>}
'** it is referring to a step described in the
'** chapter 5.1.5 Implementation of the Web User Access (Three Tier Model)
'** in the ITSO Redbook:
'******************************************************************************

'******************************************************************************
'** Set up the error handlers (Note: These are not event handlers).
****************************************************************************

Figure 46 (Part 5 of 15). Web Client Agent for CICS 3270 Direct Service
On Error GoTo GeneralErrorHandler ' Handle general LotusScript errors
On Error EILSX_ERROR GoTo MQEIErrorHandler ' EILSX errors

'* Set up LotusScript form variables for form use.
****************************************************************************
Set Session = New NotesSession
Set Doc = Session.DocumentContext
Set Mydb = Session.CurrentDatabase
Set ServName = New NotesName(session.userName)

'* Initialize the EI Session object <<4>>
****************************************************************************
Set MySession = New EISession

' Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
   Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
   Call MySession.ClearErrorCodes()
End If

'* Create an EI Service
'* <<4a>> defined in the EI Definition database.
****************************************************************************
Set MyService = MySession.CreateService(SERVICENAME)
On Event EIError From MyService Call EIServiceErrorEventHandler

'* Create the general EI Messages
'* <<4a>> defined in the EI Definition database.
****************************************************************************
Call CreateMyMessage(CESNMessage, "CESN")
Call CreateMyMessage(MLU1Message, "MLU1")
Call CreateMyMessage(VAR1Message, "VariantMsg")
Call CreateMyMessage(T001Message, "T001")
Call CreateMyMessage(T002Message, "T002")
Call CreateMyMessage(T003Message, "T003")
Call CreateMyMessage(T004Message, "T004")
Call CreateMyMessage(T005Message, "T005")
Call CreateMyMessage(T006Message, "T006")
Call CreateMyMessage(T007Message, "T007")

'* Create EI Send Options.
****************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EIOptionsErrorEventHandler

****************************************************************************
Figure 46 (Part 6 of 15). Web Client Agent for CICS 3270 Direct Service

212   Lotus Notes and the MQSeries Enterprise Integrator
'* Create EI Receive Options.
****************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler
****************************************************************************
'* Connect to the service <<5>>
****************************************************************************
Call MyService.Connect() ' Connect to CICS

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
  Select Case MyService.ReasonCode
    Case EIRC_CONNECTED: ' Already connected (probably from previous run)
      MessageBox EIRC_CONNECTED_TEXT, MB_OK, TITLE_TEXT
    Case Else :
      MessageBox EISERVICE_WARNING_TEXT & " & Cstr(MySession. ReasonCode) & ".", _
        MB_OK, TITLE_TEXT
  End Select
  Call MyService.ClearErrorCodes()
End If

****************************************************************************
'* Start Transaction CESN <<6>>
****************************************************************************
Call TranCESN

****************************************************************************
'* If no Error then'* Start Transaction TECH
****************************************************************************
If Status <> "ERROR" Then
  Call TranTECH
End If

****************************************************************************
'* If no Error then'* Stop the Conversation
****************************************************************************
If Status <> "ERROR" Then
  Call EndConversation
End If

****************************************************************************
'* Update the UiDocument with the new informations
****************************************************************************
Doc.StatusFlag=SF_PROCESSED ' Set status flag

****************************************************************************
'* Redisplay Updated document <<10>>
****************************************************************************
Call Doc.Save(True, True) ' Save backend document

Figure 46 (Part 7 of 15). Web Client Agent for CICS 3270 Direct Service
General ErrorHandler:
Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
Call Cleanup
Exit Sub

MQEIErrorHandler:
Call GlobalErrors(EILSX_ERROR_TEXT)
Call Cleanup
Exit Sub

End Sub

Sub CreateMyMessage(MyMessage As EIMessage, MessageName As String)
'*********************************************************************************
'* Create an EI Message (as defined in the MQEI Definition database)
*********************************************************************************
Set MyMessage = MySession.CreateMessage(MessageName)
On Event EIError From MyMessage Call EIMessageErrorEventHandler

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyMessage.CompletionCode = EICC_WARNING Then
  Select Case MyMessage.ReasonCode
  Case EIRC_FIELD_TRUNCATED:
    Messagebox EIRC_FIELD_TRUNCATED_TEXT, MB_OK, TITLE_TEXT
  Case Else :
    Messagebox UNRECOGNISED_TEXT & " " & Cstr(MyMessage.ReasonCode), MB_OK, TITLE_TEXT
  End Select
  Call MyMessage.ClearErrorCodes()
End If

End Sub

Sub EIMessageErrorEventHandler(ThisMessage As EIMessage, ReasonCode As Long)
'*******************************************************************************
'* Error event handler for the EIMessage Class. At this point we choose not to take
'* any corrective action, an error message is simply displayed in a message box.
'*******************************************************************************
'Sub EIMessageErrorEventHandler(ThisMessage As EIMessage, ReasonCode As Long)
'*******************************************************************************
'* Error event handler for the EIMessage Class. At this point we choose not to take
'* any corrective action, an error message is simply displayed in a message box.
'*******************************************************************************

Figure 46 (Part 8 of 15). Web Client Agent for CICS 3270 Direct Service

214 Lotus Notes and the MQSeries Enterprise Integrator
Dim StringText As String
Select Case ReasonCode
' Case EIRC_??? : ...Error Handle Process...
    Call ThisMessage.clearErrorCodes()
' Case EIRC_??? : ...Error Handle Process...
    Call ThisMessage.clearErrorCodes()
' ...
Case Else :
    StringText = EIMAGE_ERROR_TEXT + " " + Cstr(ReasonCode) + Chr$(10)
    StringText = StringText + "Message Name : " + ThisMessage.Name
    Print StringText TITLE_TEXT
End Select
'*******************************************************************************************'
' Do not clear the error code, as this will allow error EILSX_ERROR to be
' caught by the generic error handler.
'*******************************************************************************************

End Sub
Sub EIRFunctionErrorEventHanler (ThisFunction As EIRFunction, ReasonCode As Long)
'*****************************************************************************
' Error event handler for the EIRFunction Class. At this point we choose not to take
' any corrective action, an error message is simply displayed in a message box.
'***** Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
' main program to continue.
'*****************************************************************************
Dim StringText As String
Select Case ReasonCode
' Case EIRC_??? : ...
    Call ThisFunction.clearErrorCodes()
' Case EIRC_??? : ...
    Call ThisFunction.clearErrorCodes()
' ...
Case Else :
    StringText = EIR0_ERROR_TEXT + " " + Cstr(ReasonCode) + Chr$(10)
    Print StringText TITLE_TEXT
End Select
'*****************************************************************************
' Do not clear the error code, as this will allow error EILSX_ERROR to be
' caught by the generic error handler.
'*****************************************************************************

End Sub
Sub EISendOptionsErrorEventHanler (ThisSendOptions As EISendOptions, ReasonCode As Long)
'*****************************************************************************
' Error event handler for the EISendOptions Class. At this point we choose not to take
' any corrective action, an error message is simply displayed in a message box.
'***** Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
'*****************************************************************************

Figure 46 (Part 9 of 15). Web Client Agent for CICS 3270 Direct Service
** main program to continue.

******************************************************************************
Dim StringText As String
Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
    Call ThisSendOptions.clearErrorCodes()
  Case EIRC_??? :
    ...Error Handle Process...
    Call ThisSendOptions.clearErrorCodes()
  ...
Case Else :
  StringText = EISO_ERROR_TEXT & " " & Cstr(ReasonCode) & "."
  Print StringText TITLE_TEXT
End Select
******************************************************************************
** Do not clear the error code, as this will allow error EILSX_ERROR to be
** caught by the generic error handler.
******************************************************************************

Sub SendMessage(MyMessage As EIMessage, MyAttentionID As Long)
  " Send A Message
  MySendOptions.AttentionID = MyAttentionID
  Call MyService.SendMessage (MyMessage, MySendOptions)
End Sub

Sub TranCESN
  " NOTE: In the following event, numbers enclosed in '{' and '}' characters, (eg. {{1}})
  * refer to a step described in the
  * chapter 5.3.2 LotusScript (CICS 3270 Direct Service Implementation)
  * in the ITSO Redbook:
  *
  * Handling of Transaction CESN is not complete in this sample as we do not cover:
  *  - password change
  *  - wrong userid entered
  *  ....
 ******************************************************************************
  Sending a Clear {{5}}
  MySendOptions.AttentionID = EIAI_CLEAR
  Call MyService.SendMessage (Nothing, MySendOptions)

  Sending CESN Transid {{5}}
  MyService.ServiceStep = "CESN"
  MySendOptions.AttentionID = EIAI_ENTER
  Call MyService.SendMessage (Nothing, MySendOptions)

Figure 46 (Part 10 of 15). Web Client Agent for CICS 3270 Direct Service

216 Lotus Notes and the MQSeries Enterprise Integrator
' Receiving Signon Map (6)  
Status = "FALSE"  
ReceiveType = EIRT_RECEIVE  
Call ReceiveMessage (CESNMessage, ReceiveType)

If Status <> "ERROR" Then  
' Sending Userid and Password (8)  
CESNMessage.Userid = MyService.Userid  
Call SendMessage (CESNMessage, EIAI_ENTER)  

Status = "FALSE"  
ReceiveType = EIRT_RECEIVE  
'Receiving Confirmation (9)  
Call ReceiveMessage (ML01Message, ReceiveType)  
If Instr(ML01Message.MessageText, "FAA2250I") = 1 Then  
Print "LOGON OK"  
Status = "ERROR"  
End If

End If  
Exit Sub
End Sub

Sub ReceiveMessage(MyMessage As EIMessage, MyReceiveType As Long)

Status = "FALSE"  
MyReceiveOptions.ReceiveType = MyReceiveType  
'* Timeout is not supported by the EPI.  
'* We wait indefinitely on the Receive until  
'* we get the answer from the enterprise system  
'******************************************************************************'  
MyReceiveOptions.WaitType = EIWT_WAIT  
MyReceiveOptions.WaitInterval = EIWI_UNLIMITED  
Do  
' Receiving the message  
Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)  
Select Case MyService.ReasonCode  
Case EIRC_NONE:  
  Status = "OK"  
Case EIRC_NO_MESSAGE_AVAILABLE:  
  Status = "FALSE"  
Case EIRC_WRONG_MESSAGE:  
  '******************************************************************************'  
  ' Message received into the buffer but doesn’t match the fields  
  ' defined in the MyMessage map so the  
  ' EIRC_WRONG_MESSAGE warning is raised.  
  ' We therefore set ReceiveType to EIRT_RETURN to attempt  
  ' to return it against an alternative EIMessage definition, ML01Message  
  '******************************************************************************'  
    MyReceiveOptions.ReceiveType = EIRT_RETURN  
    Call MyService.ReceiveMessage (ML01Message, MyReceiveOptions)  
    Select Case MyService.ReasonCode  
    Case EIRC_NONE:

Figure 46 (Part 11 of 15). Web Client Agent for CICS 3270 Direct Service
Sub EndConversation

    ' Sending a FF12
    MySendOptions.AttentionID = EIAI_F12
    Call MyService.SendMessage (Nothing, MySendOptions)

    Status = "FALSE"
    ReceiveType = EIRT_RECEIVE
    ' Receiving the Clear Screen
    Call ReceiveMessage( MLO1Message, ReceiveType)
    ' Receiving the TECH:Session Over message
    Call ReceiveMessage( MLO1Message, ReceiveType)

End Sub

Sub TranTECH

    '*********************************************************************************************'
    ' NOTE: In the following event, numbers enclosed in '([' and '])' characters, (eg. {[1]})
    ' refer to a step described in the
    ' chapter 5.3.2 LotusScript (CICS 3270 Direct Service Implementation)
    ' in the ITSO Redbook:
    '*********************************************************************************************

    ' Sending a Clear (10)
    MySendOptions.AttentionID = EIAI_CLEAR
    Call MyService.SendMessage (Nothing, MySendOptions)

    'Sending TECH Transid (10)
    MyService.ServiceStep = "TECH"
    MySendOptions.AttentionID = EIAI_ENTER
    Call MyService.SendMessage (Nothing, MySendOptions)

    Do
    ' Receiving maps (14,...)
    Call ReceiveVARI
    If send <> "END" Then   Call SendMessage(MyMessage,EIAI_ENTER)
    Loop While Send <> "END"
    Exit Sub

End Sub

Figure 46 (Part 12 of 15). Web Client Agent for CICS 3270 Direct Service
Sub ReceiveVari
'******************************************************************************
' # We are receiving message from the enterprise system but the format is not known.
' # We first receive it into a variant message and analyse it.
' # Each screen is identified by a screen identifier in row 1, column 69.
'******************************************************************************
Dim NumOption As String
Status = "FALSE"
ReceiveType = EIRT_RECEIVE
Call ReceiveMessage (VARIMessage, ReceiveType)

ReceiveType = EIRT_RETURN
Select Case VARIMessage.MsgIdentifier
' Main Menu received
' We send back to CICS the selected option
Case "DDWINS.T001":
Call ReceiveMessage(T001Message, ReceiveType)
Select Case Doc.GetItemValue("I_SERVER_STATE") (0)
Case ":": NumOption = "1"
Case ":": NumOption = "2"
Case ":": NumOption = "3"
Case ":": NumOption = "4"
End Select
If T001Message.MESS1 = "" Then
T001Message.OPTION = NumOption
Send = "YES"
End If
' Show Account Details received
' We send back to CICS the entered surname or account number
Case "DDWINS.T002":
Call ReceiveMessage(T002Message, ReceiveType)
If T002Message.MESS2 = "" Then
T002Message.SUR2 = Doc.GetItemValue("I_Surname") (0)
T002Message.ACC2 = Doc.GetItemValue("I_Account") (0)
Set MyMessage = T002Message
Send = "YES"
Else
Doc.O_RESPONSES = T002Message.MESS2
Send = "END"
End If
' Account Details received
' We display back on the document the details of the account.
' Transaction is ended
Case "DDWINS.T003":
Call ReceiveMessage(T003Message, ReceiveType)
If T003Message.MESS3 = "" Then
Doc.I_SURNAME = Trim (T003Message.SUR3)
Doc.I_FIRST_NAME = Trim (T003Message.FOR3)
Doc.I_ACCOUNT = Trim (T003Message.ACC3)
Doc.I_ADDRESS = Trim (T003Message.ADD13)
Doc.I_CITY_STATE = Trim (T003Message.ADD23)
End If

Figure 46 (Part 13 of 15). Web Client Agent for CICS 3270 Direct Service
**Add an Account received**

We get the details of the account to be added from the document and send back the information to CICS

Case "DDW1MS.T004":

Call ReceiveMessage(T004Message, ReceiveType)

If T004Message.MESS4 = "" Then

T004Message.SUR4 = Doc.GetItemValue("I_SURNAME") (0)  
T004Message.FOR4 = Doc.GetItemValue("I_FIRST_NAME") (0)  
T004Message.ADD4 = Doc.GetItemValue("I_ADDRESS") (0)  
T004Message.ADD24 = Doc.GetItemValue("I_CITY_STATE") (0)  
T004Message.ADD34 = Doc.GetItemValue("I_POSTAL_CODE") (0)  
T004Message.BAL4 = Cstr(Doc.GetItemValue("I_BALANCE") (0) )

Set MyMessage = T004Message
Send = "YES"

Else

Doc.I_ACCOUNT = " "
Doc.O_RESPONSES = T004Message.MESS4
Send = "END"

End If

**Delete an Account received**

We get the account number to be deleted from the document and send back the information to CICS

Case "DDW1MS.T005":

Call ReceiveMessage(T005Message, ReceiveType)

If T005Message.MESS5 = "" Then

T005Message.ACC5 = Doc.GetItemValue("I_ACCOUNT") (0)

Set MyMessage = T005Message
Send = "YES"

Else

Doc.O_RESPONSES = T005Message.MESS5
Send = "END"

End If

**Account Update received**

We get the details of the account to be updated from the document and send back the information to CICS

Case "DDW1MS.T006":

Call ReceiveMessage(T006Message, ReceiveType)

If T006Message.MESS6 = "" Then

T006Message.SUR6 = Doc.GetItemValue("I_SURNAME") (0)  
T006Message.FOR6 = Doc.GetItemValue("I_FIRST_NAME") (0)  
T006Message.ADD26 = Doc.GetItemValue("I_CITY_STATE") (0)  
T006Message.ADD36 = Doc.GetItemValue("I_POSTAL_CODE") (0)  
T006Message.BAL6 = Cstr(Doc.GetItemValue("I_BALANCE") (0) )

Set MyMessage = T006Message
Send = "YES"

Else

Doc.O_RESPONSES = T006Message.MESS6
Send = "END"

End If

Figure 46 (Part 14 of 15). Web Client Agent for CICS 3270 Direct Service

220 Lotus Notes and the MQSeries Enterprise Integrator
T006Message.BAL6 = Cstr(Doc.GetItemValue("I_BALANCE") (0) )
Set MyMessage = T006Message
Send = "YES"
Else
  Doc.O_RESPONSES = T006Message.Mess6
  Send = "END"
End If
' Update an Account received
' We get the account number to be updated from the document
' and send back the information to CICS
Case "DDWIMS.T007":
  Call ReceiveMessage(T007Message, ReceiveType) If T007Message.MESS7 = "" Then
    T007Message.ACC7 = Doc.GetItemValue("I_ACCOUNT") (0) Set MyMessage = T007Message
    Send = "YES"
  Else
    Doc.O_RESPONSES = T007Message.MESS7
    Send = "END"
  End If
End Select
C.3 Web Client Agent for Native MQSeries Service

'Web Agent (Native MQSeries):

*******************************************************************************
* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator*
* ITSO Sample Application*
* Web Agent for the MQEI Native MQSeries Service*
*******************************************************************************
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselsx "eilsx"

*******************************************************************************
* Define Enterprise Integrator objects.*
*******************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendMessage As EIMessage
Dim MyReceiveMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions

*******************************************************************************
* Notes variables for document manipulation.*
*******************************************************************************
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim MyDb As Notesdatabase
Dim ServName As NotesName

*******************************************************************************
* Wait time in milliseconds for a reply.*
*******************************************************************************
Const FIVE_SECONDS = 5000 ' 5 seconds

*******************************************************************************
* To hold values expected from the EI Definition database*
*******************************************************************************
Const SERVICE_NAME = "MQServ"
Const SEND_MESSAGE_NAME = "VSAMSERVCOMMAREA"
Const RECEIVE_MESSAGE_NAME = "VSAMSERVCOMMAREA"

*******************************************************************************

Figure 47 (Part 1 of 11). Web Client Agent for Native MQSeries Service
Figure 47 (Part 2 of 11). Web Client Agent for Native MQSeries Service
"* Definition of warning/error messages
**********************************************************************************
Const EIRC_CONNECTED_STRING = "An attempt was made to connect " _
+ "to a service when already connected. The connect is ignored and " _
+ "the program continues."
Const EIRC_NO_MSG_AVAILABLE_STRING = "An attempt was made to " _
+ "receive a message but the target message was not found. " _
+ "Please, retry later."
Const EIRC_FIELD_TRUNCATED_STRING = "A length of a string value of " _
+ "an EMessage field exceeds the length specified on the MQSeries EI Definition " _
+ "database. Correct the LotusScript to make sure that the string value " _
+ "being passed as the EMessage field value does not exceed the " _
+ "predefined length."
Const EIRC_WRONG_MESSAGE_STRING = "A message was received that " _
+ "does not match the EMessage object passed as a parameter on the " _
+ "EIService ReceiveMessage call."
Const UNRECOGNISED_STRING = "Unrecognized warning message :"
Const EISERVICE_WARNING = "Warning message received establishing " _
+ "MQSeries. ReasonCode :"
Const EISESSION_ERROR = "An error has been reported from the " _
+ "EISession Class. ReasonCode"
Const THISMESSAGE_ERROR = "An error has been reported from the " _
+ "EMessage Class. ReasonCode :":
Const THISSERVICE_ERROR = "An error has been reported from the " _
+ "EIService Class. ReasonCode :":
Const THISSO_ERROR = "An error has been reported from the " _
+ "EISendOptions Class. ReasonCode :":
Const THISRO_ERROR = "An error has been reported from the " _
+ "EIReceiveOptions Class. ReasonCode :":
**********************************************************************************
"* Definition of StatusFlag Values
**********************************************************************************
Const SF_NO_MORE_REPLIES_YET="No MQ Reply found"
Const SD_NO_MORE_REPLIES_YET="Request has been submitted, but no reply has been received yet. " _
+ "Press button to retry!"
Const SF_NONE=""
Const SD_NONE=""
Const SF_PROCESSED="Processed"

Sub Initialize
**********************************************************************************
 )* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator"
)* VIISO Sample Application
)* Web Agent for the MQEI Native MQSeries Service
**********************************************************************************
)* This sample illustrates how a Lotus Script application can communicate
* to an enterprise system using MQSeries
*
*A LotusScript application writes to a MQSeries queue.
*) A CICS transaction, MQ1, is started via MQSeries triggering.
*) CICS program AMQICIC1 then executes to get the message from the queue and calls

Figure 47 (Part 3 of 11). Web Client Agent for Native MQSeries Service

224 Lotus Notes and the MQSeries Enterprise Integrator
program VSAMSERV to process the message.

VSAMSERV then returns to AMQCI1 which puts a reply back to a queue.

The LotusScript application gets the message and updates the document.

This agent is activated when the "Native MQ Series" Service Selection radio button is selected and the "Submit" button is clicked.

NOTE: When you see a number enclosed in '<<' and '>>' characters, (eg. <<1>>) it is referring to a step in the Web User Access (Three Tier Model) diagram in the ITSO Redbook "Lotus Notes and the MQSeries Enterprise Integrator"

Initialize the EILSX objects. <<<>

Set up the error handlers. (Note: These are not event handlers).

On Error GoTo GeneralErrorHandler
On Error EILSX_ERROR GoTo MQEIErrorHandler

Initialize the EISession object <<<>

Set MySession = New EISession

Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
    Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
End If

Create an EI Service

Create EI send message

Check for warnings (We have decided to ignore warnings for the purposes of this sample)

MySendMessage.CompletionCode = EICC_WARNING Then
    Select Case MySendMessage.ReasonCode
        Case EIRC_FIELD_TRUNCATED :
            Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE
        Case Else :

Figure 47 (Part 4 of 11). Web Client Agent for Native MQSeries Service

Appendix C. Web Agents  225
Print UNRECOGNISED_STRING + " " + Cstr(MySendMessage.ReasonCode) WARNINGTITLE
End Select
Call MySendMessage.ClearErrorCodes
End If

*****************************************************************************************
' Create EI receive message
' <<4a>> defined in the EI Definition database).
*****************************************************************************************
Set MyReceiveMessage = MySession.CreateMessage(RECEIVE_MESSAGE_NAME)
On Event EIError From MyReceiveMessage Call EIMessageErrorEventHandler
If MyReceiveMessage.CompletionCode = EICC.WARNING Then
Select Case MyReceiveMessage.ReasonCode
Case EIRC_FIELD_TRUNCATED :
    Print UNRECOGNISED_STRING + " " + Cstr(MySendMessage.ReasonCode) WARNINGTITLE
Case Else :
    Print UNRECOGNISED_STRING + " " + Cstr(MySendMessage.ReasonCode) WARNINGTITLE
End Select
Call MyReceiveMessage.ClearErrorCodes
End If

*****************************************************************************************
' Create EI send message options
*****************************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EIError From MySendOptions Call EISendOptionsErrorEventHandler

'Option must be set to make MQEI define the MQ message type 'request' in the MQ message descriptor
MySendOptions.MessageType=EIMT_REQUEST

*****************************************************************************************
' Create EI receive message options
*****************************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler

MyReceiveOptions.WaitType=EIWT_WAIT
MyReceiveOptions.WaitInterval=FIVE SECONDS

*****************************************************************************************
' Set document variable doc
*****************************************************************************************
Set Session = New NotesSession
Set Doc=Session.DocumentContext
Set Mydb = Session.CurrentDatabase
Set ServName = New NotesName(session.userName)

*****************************************************************************************
' Connect to EI Service <<5>>
*****************************************************************************************
Call MyService.Connect

Figure 47 (Part 5 of 11). Web Client Agent for Native MQSeries Service

226   Lotus Notes and the MQSeries Enterprise Integrator
' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
    Case EIRC_CONNECTED:
        Print EIRC_CONNECTED_STRING WARNINGTITLE
    Case Else:
        Print EISERVICE_WARNING + " " + Cstr(MyService.ReasonCode) WARNINGTITLE
    End Select
    Call MyService.ClearErrorCodes
End If

************************************************************************************************
* Send the Message to the EI Service selected if StatusFlag is not SF_NO_MQ_REPLY_MSG_YET
* otherwise message has been sent already and just a receive has to take place
************************************************************************************************
If Doc.GetFirstItem("StatusFlag").Text<>SF_NO_MQREPLY_YET Then

******************************************************************************
* Setup and send message to CICS <<6>>
******************************************************************************
MySendMessage.SERVER_STATE = Doc.GetItemValue("I_SERVER_STATE")\(0\)
MySendMessage.SERVER_KEY_SIZE = Cint(Doc.GetItemValue("I_SERVER_KEY_SIZE")\(0\))
MySendMessage.SURNAME = Doc.GetItemValue("I_SURNAME")\(0\)
MySendMessage.FIRST_NAME = Doc.GetItemValue("I_FIRST_NAME")\(0\)
MySendMessage.ACCOUNT = Doc.GetItemValue("I_ACCOUNT")\(0\)
MySendMessage.ADDRESS = Doc.GetItemValue("I_ADDRESS")\(0\)
MySendMessage.CITY_STATE = Doc.GetItemValue("I_CITY_STATE")\(0\)
MySendMessage.POSTAL_CODE = Doc.GetItemValue("I_POSTAL_CODE")\(0\)
MySendMessage.BALANCE = Cint(Doc.GetItemValue("I_BALANCE")\(0\))

******************************************************************************
* Send the message <<7>>
******************************************************************************
Call MyService.SendMessage( MySendMessage, MySendOptions ) <<7>>

******************************************************************************
* Generated MQ message Id is stored in document for delayed receive option
******************************************************************************
Doc.MQRequestMsgId=MySendOptions.Identifier
End If

************************************************************************************************
* Receive the message <<8>>
************************************************************************************************
'Identifier property must be set to receive the reply message to a given request message
'which is identified by the identifier stored in the MQRequestMsgId of the form
'otherwise MQEI would receive the first message in the queue
MyReceiveOptions.Identifier=Doc.GetItemValue("MQRequestMsgId")\(0\)

Figure 47 (Part 6 of 11). Web Client Agent for Native MQSeries Service
Call MyService.ReceiveMessage (MyReceiveMessage, MyReceiveOptions) '**************************************************************************
'Handle MQEI warning messages
**************************************************************************
Doc.StatusFlag=SF_NONE
Doc.StatusDescription=SD_NONE
If MyService.CompletionCode = EICC_WARNING Then Select Case MyService.ReasonCode Case EIRC_NO_MESSAGE_AVAILABLE :
   Print EIRC_NO_MSG_AVAILABLE_STRING WARNINGTITLE
   ' Set Status Flag, so that user can retry later (delayed receive)
   Doc.StatusFlag=SF_NO_MQREPLY_YET
   Doc.StatusDescription=SD_NO_MQREPLY_YET
Case EIRC_FIELD_TRUNCATED :
   Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE
Case EIRC_WRONG_MESSAGE :
   Print EIRC_WRONG_MESSAGE_STRING WARNINGTITLE
Case Else :
   Print UNRECOGNISED_STRING & " " & MyService.ReasonCode & WARNINGTITLE
End Select
Call MyService.ClearErrorCodes
Else
   '**************************************************************************
   ' Update backend document with receive message <<9>>
   '**************************************************************************
   Doc.I_SERVER_STATE=Trim(MyReceiveMessage.SERVER_STATE)
   Doc.I_SERVER_KEY_SIZE=Cstr(MyReceiveMessage.SERVER_KEY_SIZE)
   Doc.I_SURNAME=Trim(MyReceiveMessage.SURNAME)
   Doc.I_FIRST_NAME=Trim(MyReceiveMessage.FIRST_NAME)
   Doc.I_ACCOUNT=Trim(MyReceiveMessage.ACCOUNT)
   Doc.I_ADDRESS=Trim(MyReceiveMessage.ADDRESS)
   Doc.I_CITY_STATE =Trim(MyReceiveMessage.CITY_STATE)
   Doc.I_POSTAL_CODE =Trim(MyReceiveMessage.POSTAL_CODE)
   Doc.I_BALANCE=Cstr(MyReceiveMessage.BALANCE)
   Doc.I_RESPONSES=Trim(MyReceiveMessage.RESPONSES)
   Doc.StatusFlag=SF_PROCESSED ' Set status flag
End If

'**************************************************************************
' Redisplay Updated document <<10>>
**************************************************************************
Call Doc.Save(True, True) ' Save backend document
Print "[http://"+ServName.Common+"/"+Mydb.filename+"/All+Docs/"+Doc.NoteID +"?OpenDocument"]"
Call CleanUp ' Disconnect from enterprise serives and delete all MQEI objects

Figure 47 (Part 7 of 11). Web Client Agent for Native MQSeries Service
```vbscript
'*****************************************************************************
'* General LotusScript Error Handler'*****************************************************************************
GeneralErrorHandler:
    Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
    CleanUp
    Exit Sub

'*****************************************************************************
'* Handle EILSX errors.'*****************************************************************************
MQEIErrorHandler:
    Call GlobalErrors(EILSX_ERROR_TEXT)
    CleanUp
    Exit Sub
End Sub

Sub EIServiceErrorEventHandler (thisService As EIService, ReasonCode As Long)
'*******************************************************************************************
'* Error Handler for the EIService Class. At this point we choose not to take
'* any corrective action, the error is simply displayed in a message box.
'*
'* Alternatively you could write a CASE statement to handle your chosen
'* errors. Once handled the Error Codes can be cleared, allowing the
'* main program to continue.
*******************************************************************************************
    Select Case ReasonCode
    ' Case EIRC_????:
    '     ...Error Handle Process...
    '     thisService.clearErrorCodes
    ' Case EIRC_????:
    '     ...Error Handle Process...
    '     thisService.clearErrorCodes
    '     ...
    Case Else:
        Print EISERVICE_ERROR_TEXT + " " + CStr(ReasonCode) + ". Service Name: " + thisService.Name + "." TITLE_TEXT
    End Select
End Sub

Sub EIMessageErrorEventHandler (thisMessage As EIMessage, ReasonCode As Long)
'*******************************************************************************************
'* Error Handler for the EIMessage Class. At this point we choose not to take
'* any corrective action, the error is simply displayed in a message box.
'*
'* Alternatively you could write a CASE statement to handle your chosen
'* errors. Once handled the Error Codes can be cleared, allowing the
'* main program to continue.
*******************************************************************************************
    Select Case ReasonCode
    ' Case EIRC_???? : ...Error Handle Process...

Figure 47 (Part 8 of 11). Web Client Agent for Native MQSeries Service
```
Case Else :
    Print EIMessage_error_text & " " & CStr(ReasonCode) & " . Message Name: " & thisMessage.Name & "." & TITLE_TEXT
End Select
End Sub

Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long) '*******************************************************************************************
" Error Handler for the EIService Class. At this point we choose not to take
" any corrective action, the error is simply displayed in a message box.
" Alternatively you could write a CASE statement to handle your chosen
" errors. Once handled the Error Codes can be cleared, allowing the
" main program to continue.
**************************************************************************
Select Case ReasonCode
  Case EIRC_???:
    "...Error Handle Process...
    thisSendOptions.clearErrorCodes
  Case EIRC_???:
    "...Error Handle Process...
    thisSendOptions.clearErrorCodes
  Case Else :
    Print EISO_ERROR_TEXT & " " & CStr(ReasonCode) & "." & TITLE_TEXT
End Select
End Sub

Sub EIReceiveOptionsErrorEventHandler (thisReceiveOptions As EIReceiveOptions, ReasonCode As Long) '*******************************************************************************************
" Error Handler for the EIService Class. At this point we choose not to take
" any corrective action, the error is simply displayed in a message box.
" Alternatively you could write a CASE statement to handle your chosen
" errors. Once handled the Error Codes can be cleared, allowing the
" main program to continue.
**************************************************************************
Select Case ReasonCode
  Case EIRC_???:
    "...Error Handle Process...
    thisReceiveOptions.clearErrorCodes
  Case EIRC_???:
    "...Error Handle Process...
    thisReceiveOptions.clearErrorCodes
  Case Else :
    Print EIRO_ERROR_TEXT & " " & CStr(ReasonCode) & "." & TITLE_TEXT
End Select
End Sub

Sub GlobalErrors(CallerText As String)

Figure 47 (Part 9 of 11). Web Client Agent for Native MQSeries Service

230  Lotus Notes and the MQSeries Enterprise Integrator
Dim ErrorText As String

Print CallerText

' Format the error text for the message box
ErrorText = CallerText + Chr$(10)
ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$
ErrorText = ErrorText + Chr$(10) + LINE_NUMBER_TEXT + " = " + Str(Erl)

' Output the error to message box and print line
Print ErrorText & TITLE_TEXT

End Sub

Sub EISessionErrorEventHandler(ThisSession As EISession, ReasonCode As Long)

' Error event handler for the EISession Class. At this point we choose not to take
' any corrective action, an error message is simply displayed in a message box.
' *
' Alternatively you could write a CASE statement to handle your chosen
' errors. Once handled the Error Codes can be cleared, allowing the
' main program to continue.
' Note: this subroutine is registered as an error event handler for EISession
' events but is also called following a failure in EISession creation in Queryopen

Dim StrText As String

Select Case ReasonCode
' Case EIRC_??? :
' ...Error Handle Process...
' Case EIRC_??? :
' ...Error Handle Process...
' ...
Case EIRC_SEC_DB_SYSERROR:
StrText = EISSESSION_ERROR_TEXT & " " & Cstr(ReasonCode) & "." + Chr$(10)
StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
Print StrText & TITLE_TEXT
Case EIRC_DEFN_DB_SYSERROR:
StrText = EISSESSION_ERROR_TEXT & " " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)
StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
Print StrText & TITLE_TEXT
Case Else :
StrText = EISSESSION_ERROR_TEXT & " : " & Cstr(ThisSession.ReasonCode)

Figure 47 (Part 10 of 11). Web Client Agent for Native MQSeries Service
End Select

******************************************************************************
** Do not clear the error code, as this will allow error EILSX_ERROR to be
** caught by the generic error handler.
******************************************************************************

End Sub

Sub CleanUp

******************************************************************************
** This function is called when the agent is finished.
** Disconnect from MQSeries and delete the EI objects.
******************************************************************************
On Error Resume Next ' Ignore any error as we need to shutdown regardless.

******************************************************************************
** Disconnect from MQSeries.
******************************************************************************
Call MyService.Disconnect()

******************************************************************************
** Delete the EI objects.
******************************************************************************
Delete MyReceiveOptions
Delete MySendOptions
Delete MySendMessage
Delete MyReceiveMessage
Delete MyService
Delete MySession

End Sub

Figure 47 (Part 11 of 11). Web Client Agent for Native MQSeries Service
C.4 Web Client Agent for CICS DPL via MQSeries Service

"Web Agent (CICS DPL via MQSeries):

******************************************************************************
* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator
* ITSO Sample Application
* Web Agent for the MQEI CICS DPL via MQSeries Service
******************************************************************************
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
Uselex "eilsx"

******************************************************************************
* Define Enterprise Integrator objects.
******************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendMessage As EIMessage
Dim MyReceiveMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EISendOptions

******************************************************************************
* Notes variables for document manipulation.
******************************************************************************
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim MyDb As NotesDatabase
Dim ServName As NotesName

******************************************************************************
* Wait time in milliseconds for a reply.
******************************************************************************
Const FIVE_SECONDS = 5000 ' 5 seconds

******************************************************************************
* To hold values expected from the EI Definition database
******************************************************************************
Const SERVICE_NAME = "MQCICSDPL"
Const SEND_MESSAGE_NAME = "VSAMSERVCOMAREA"
Const RECEIVE_MESSAGE_NAME = "VSAMSERVCOMAREA"

******************************************************************************

Figure 48 (Part 1 of 11). Web Client Agent for CICS DPL via MQSeries Service
Print text

Const INIT_TEXT = "Initializing..."
Const SEND_TEXT = "Sending..."
Const RECEIVE_TEXT = "Receiving..."
Const CONSTRUCTING = "Constructing Message..."
Const SENDSUCCESS = "Send Message Completed Successfully"
Const RECEIVESUCCESS = "Receive Message Completed Successfully"
Const CONNECT_TEXT = "Connecting..."
Const DISCONNECT_TEXT = "Disconnecting..."
Const OK_TEXT = "Press 'OK' button to Send Message..."
Const BLANK = " 

Text for use in error message box

Const WARNINGTITLE = "$MQEI Warning"
Const ERRORTITLE = "$MQEI Error"
Const GENERALTITLE = "General LotusScript Error"
Const NOTESEXERROR = "LotusScript Error. Exit Sample.
Const MQIERERROR = "$MQEI Error. Exit Sample.
Const NOTESEXERROREXIT = "LotusScript Error. Exiting Sample..."
Const MQIERERROREXIT = "$MQEI Error. Exiting Sample..."
Const PRIMARYSYSTEM = "Primary System Error Code"
Const SECONDARYSYSTEM = "Secondary System Error Code"
Const SYSTEMERRORTEXT = "Description"
Const TITLE_TEXT = "MQ Native sample"
Const LOTUSSCRIPTERROR_TEXT = "LotusScript error!"
Const EILSX_ERROR_TEXT = "EILSX error!"
Const ERROR_NUMBER_TEXT = "Error number"
Const ERROR_DESCRIPTION_TEXT = "Error description"
Const LINE_NUMBER_TEXT = "Line number"
Const EIRC_CONNECTED_TEXT = "An attempt was made to connect to a service when already connected." _
  + "The connect is ignored and the program continues."
Const EIRC_DISCONNECTED_TEXT = "An attempt was made to disconnect to a service when already " _
  + "disconnected. The disconnect is ignored and the program continues."
Const EIRC_NO_MESSAGE_AVAILABLE_TEXT = "No message available - Retry operation"
Const EIRC_TRUNCATED_MESSAGE_TEXT = "A length of a string value of an EIMessage field exceeds the " _
  + "length specified on the Definition Database. Correct the LotusScript to make sure that " _
  + "the string value being passed as the EIMessage field value does not exceed the predefined length."
Const EIRC_WRONG_MESSAGE_TEXT = "A message was received that does not match the EIMessage object " _
  + "passed as a parameter on the EIService ReceiveMessage call."
Const UNRECOGNISED_TEXT = "Unrecognised warning message";
Const EISSERVICEWARNING_TEXT = "Warning message received establishing MQService. ReasonCode: 
 Const EISSSESSION_ERROR_TEXT = "An error has been reported from the EISession Class. ReasonCode: 
 Const EIMESSAGE_ERROR_TEXT = "An error has been reported from the EIMessage Class. ReasonCode: 
 Const EISERVICE_ERROR_TEXT = "An error has been reported from the EIService Class. ReasonCode: 
 Const EISO_ERROR_TEXT = "An error has been reported from the EISendOptions Class. ReasonCode: 
 Const EISO_ERROR_TEXT = "An error has been reported from the EISendOptions Class. ReasonCode: 
 Const SAMPLE_END_TEXT = "Sample completed."

Figure 48 (Part 2 of 11). Web Client Agent for CICS DPL via MQSeries Service

Lotus Notes and the MQSeries Enterprise Integrator
Const EIRC_CONNECTED_STRING = "An attempt was made to connect " + "the program continues."
Const EIRC_NO_MSG_AVAILABLE_STRING = "An attempt was made to " + "receive a message but the target message was not found. " + "Please, retry later."
Const EIRC_FIELD_TRUNCATED_STRING = "A length of a string value of " + "an EIMessage field exceeds the length specified on the MQSeries EI Definition " + "database. Correct the LotusScript to make sure that the string value " + "being passed as the EIMessage field value does not exceed the " + "predefined length."
Const EIRC_WRONG_MESSAGE_STRING = "A message was received that " + "does not match the EIMessage object passed as a parameter on the " + "EIReceiveReceiveMessage call."
Const UNRECOGNISED_STRING = "Unrecognized warning message:" Const EISERVICE_WARNING = "Warning message received establishing " + "MQService. ReasonCode:" Const EISSESSION_ERROR = "An error has been reported from the " + "EISession Class. ReasonCode"
Const THISMESSAGE_ERROR = "An error has been reported from the " + "EIMessage Class. ReasonCode :"
Const THISSERVICE_ERROR = "An error has been reported from the " + "EIService Class. ReasonCode :"
Const THISSO_ERROR = "An error has been reported from the " + "EISendOptions Class. ReasonCode :"
Const THISRO_ERROR = "An error has been reported from the " + "EIReceiveOptions Class. ReasonCode :"

Const SF_NO_MORE_REPLY_YET="No MQ Reply found"
Const SD_NO_MORE_REPLY_YET="Request has been submitted, but no reply has been received yet." + "Press button to retry!"
Const SF_NONE=""
Const SD_NONE=""
Const SF_PROCESSED="Processed"

Sub Initialize

Figure 48 (Part 3 of 11). Web Client Agent for CICS DPL via MQSeries Service
** Initialize the EILSX objects. <<4>>

******************************************************************************************

** Set up the error handlers. (Note: These are not event handlers).
******************************************************************************************

On Error Goto GeneralErrorHandler
On Error EILSX_ERROR Goto MQEIErrorHandler

******************************************************************************************

** Initialize the EISession object <<4>>

Set MySession = New EISession

' Check for errors
If MySession.ReasonCode <> EIRC_NONE Then
   Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
   Call MySession.ClearErrorCodes()
End If

******************************************************************************************

** Create an EI Service

<<4a>> defined in the EI Definition database.

Set MyService = MySession.CreateService(SERVICE_NAME)

On Event EIError From MyService Call EIServiceErrorEventHandler

******************************************************************************************

** Create EI send message

<<4a>> defined in the EI Definition database.

Set MySendMessage = MySession.CreateMessage(SEND_MESSAGE_NAME)

On Event EIError From MySendMessage Call EIMessageErrorEventHandler

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MySendMessage.CompletionCode = EICC_WARNING Then
   Select Case MySendMessage.ReasonCode
   Case EIRC_FIELD_TRUNCATED :
      Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE

Figure 48 (Part 4 of 11). Web Client Agent for CICS DPL via MQSeries Service

236  Lotus Notes and the MQSeries Enterprise Integrator
Case Else :
    Print UNRECOGNISED_STRING + " " + Cstr(MySendMessage.ReasonCode) WARNINGTITLE
End Select
Call MySendMessage.ClearErrorCodes
End If

'*****************************************************************************************
'** Create EI receive message
'** <<4a>> defined in the EI Definition database).
'*****************************************************************************************
Set MyReceiveMessage = MySession.CreateMessage(RECEIVE_MESSAGE_NAME)
On Event EIError From MyReceiveMessage Call EIMessageErrorEventHandler
If MyReceiveMessage.CompletionCode = EICC_WARNING Then
    Select Case MyReceiveMessage.ReasonCode
    Case EIRC_FIELD_TRUNCATED :
        Print UNRECOGNISED_STRING + " " + Cstr(MySendMessage.ReasonCode) WARNINGTITLE
    Case Else :
        Print UNRECOGNISED_STRING + " " + Cstr(MySendMessage.ReasonCode) WARNINGTITLE
    End Select
    Call MyReceiveMessage.ClearErrorCodes
End If

'*****************************************************************************************
'** Create EI send message options
'*****************************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EISendOptionsError From MySendOptions Call EISendOptionsErrorEventHandler
MySendOptions.MessageType=EIMT_REQUEST

'*****************************************************************************************
'** Create EI receive message options
'*****************************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIReceiveOptionsError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler
MyReceiveOptions.WaitType=EIWT_WAIT
MyReceiveOptions.WaitInterval=FIVE.SECONDS

'*****************************************************************************************
'** Set document variable Doc
'*****************************************************************************************
Set Session = New NotesSession
Set Doc=Session.DocumentContext
Set Mydb = Session.CurrentDatabase
Set ServName = New NotesName(session.userName)

'*****************************************************************************************
'** Connect to EI Service <<5>>
'*****************************************************************************************

Figure 48 (Part 5 of 11). Web Client Agent for CICS DPL via MQSeries Service
Call MyService.Connect
' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
        Case EIRC_CONNECTED
            Print EIRC_CONNECTED_STRING WARNINGTITLE
        Case Else
            Print EISERVICE_WARNING + " " + Cstr(MyService.ReasonCode) WARNINGTITLE
    End Select
End If
Call MyService.ClearErrorCodes
End If

************************************************************************************************
'**************************************************************
' Send the Message to the EI Service selected if StatusFlag is not
' SF_NO_MQ_REPLY_MSG_YET
' otherwise message has been sent already and just a receive has to take
' place
'******************************************************************************
If Doc.GetFirstItem("StatusFlag").Text<>SF_NO_MQREPLY_YET Then
    '**************************************************************
    ' Setup and send message to CICS  <<6>>
    '******************************************************************************
    MySendMessage.SERVER_STATE = Doc.GetItemValue("I_SERVER_STATE") (0)
    MySendMessage.SERVER_KEY_SIZE = Cint(Doc.GetItemValue("I_SERVER_KEY_SIZE") (0))
    MySendMessage.SURNAME = Doc.GetItemValue("I_SURNAME") (0)
    MySendMessage.FIRST_NAME = Doc.GetItemValue("I_FIRST_NAME") (0)
    MySendMessage.ACCOUNT = Doc.GetItemValue("I_ACCOUNT") (0)
    MySendMessage.ADDRESS = Doc.GetItemValue("I_ADDRESS") (0)
    MySendMessage.CITY_STATE = Doc.GetItemValue("I_CITY_STATE") (0)
    MySendMessage.POSTAL_CODE = Doc.GetItemValue("I_POSTAL_CODE") (0)
    MySendMessage.BALANCE = Cint(Doc.GetItemValue("I_BALANCE") (0))

    '******************************************************************************
    ' Send the message  <<7>>
    '******************************************************************************
    Call MyService.SendMessage ( MySendMessage, MySendOptions ) '<<7>>

    '******************************************************************************
    ' Generated MQ message Id is stored in document for delayed
    ' receive option
    '******************************************************************************
    Doc.MQRequestMsgId=MySendOptions.Identifier
End If

************************************************************************************************
'**************************************************************
' Identifier property must be set to recieve the reply message to a given request message
'which is identified by the identifier stored in the MQRequestMsgId of the form
'otherwise MQEI would recieve the first message in the queue

Figure 48 (Part 6 of 11). Web Client Agent for CICS DPL via MQSeries Service

238

Lotus Notes and the MQSeries Enterprise Integrator
MyReceiveOptions.Identifier=Doc.GetItemValue("MQRequestMsgId")(0)
Call MyService.ReceiveMessage ( MyReceiveMessage, MyReceiveOptions )

'************************************************************************************************
'************************************************************************************************
 Doc.StatusFlag=SF_NONE
Doc.StatusDescription=SD_NONE
If MyService.CompletionCode = EICC_WARNING Then
  Select Case MyService.ReasonCode
  Case EIRC_NO_MESSAGE_AVAILABLE :
    Print EIRC_NO_MSG_AVAILABLE_STRING WARNINGTITLE
    ' Set Status Flag, so that user can retry later (delayed receive)
    Doc.StatusFlag=SF_NO_MQREPLY_YET
    Doc.StatusDescription=SD_NO_MQREPLY_YET
  Case EIRC_FIELD_TRUNCATED :
    Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE
  Case EIRC_WRONG_MESSAGE :
    Print EIRC_WRONG_MESSAGE_STRING WARNINGTITLE
  Case Else :
    Print UNRECOGNISED_STRING & " " & MyService.ReasonCode & " " & WARNINGTITLE
  End Select
  Call MyService.ClearErrorCodes
Else
  '****************************************************************************
  ' Update backend document with receive message ... (MyReceiveMessage.POSTAL_CODE)
  Doc.I_BALANCE=Cstr(MyReceiveMessage.BALANCE)
  Doc.O_RESPONSES=Trim(MyReceiveMessage.RESPONSES)
  Doc.StatusFlag=SF_PROCESSED ' Set status flag
End If

'************************************************************************************************
'************************************************************************************************

Doc.I_SERVER_STATE=Trim(MyReceiveMessage.SERVER_STATE)
Doc.I_SERVER_KEY_SIZE=Cstr(MyReceiveMessage.SERVER_KEY_SIZE)
Doc.I_SUBNAME=Trim(MyReceiveMessage.SUBNAME)
Doc.I_FIRST_NAME=Trim(MyReceiveMessage.FIRST_NAME)
Doc.I_ACCOUNT=Trim(MyReceiveMessage.ACCOUNT)
Doc.I_ADDRESS=Trim(MyReceiveMessage.ADDRESS)
Doc.I_CITY_STATE =Trim(MyReceiveMessage.CITY_STATE)
Doc.I_POSTAL_CODE =Trim(MyReceiveMessage.POSTAL_CODE)
Doc.I_BALANCE=Cstr(MyReceiveMessage.BALANCE)
Doc.O_RESPONSES=Trim(MyReceiveMessage.RESPONSES)

Doc.StatusFlag=SF_PROCESSED ' Set status flag

'*****************************************************************************
'*****************************************************************************

Print "[http://*ServName.Common*/*Mydb.filename*/All+Docs/*Doc.NoteID +""?OpenDocument]
Call CleanUp ' Disconnect from enterprise servies and delete all MQEI objects

Figure 48 (Part 7 of 11). Web Client Agent for CICS DPL via MQSeries Service
Figure 48 (Part 8 of 11). Web Client Agent for CICS DPL via MQSeries Service
Case EIRC_??? : ...Error Handle Process...
    thisMessage.clearErrorCodes
Case EIRC_??? : ...Error Handle Process...
    thisMessage.clearErrorCodes
...
Case Else :
    Print EMESSAGE_ERROR_TEXT & " " & Cstr(ReasonCode) & ". Message Name: " & thisMessage.Name & ". & TITLE_TEXT
End Select
End Sub

Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long)
"** Error Handler for the EIService Class. At this point we choose not to take
" any corrective action, the error is simply displayed in a message box.
**
" Alternatively you could write a CASE statement to handle your chosen
" errors. Once handled the Error Codes can be cleared, allowing the
" main program to continue.
***************************************************************************
Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
        thisSendOptions.clearErrorCodes
    Case EIRC_??? :
        ...Error Handle Process...
        thisSendOptions.clearErrorCodes
    ...
    Case Else :
        Print EISO_ERROR_TEXT & " " & Cstr(ReasonCode) & ". & TITLE_TEXT
End Select
End Sub

Sub EIReceiveOptionsErrorEventHandler (thisReceiveOptions As EIReceiveOptions, ReasonCode As Long)
***************************************************************************
"** Error Handler for the EIService Class. At this point we choose not to take
" any corrective action, the error is simply displayed in a message box.
**
" Alternatively you could write a CASE statement to handle your chosen
" errors. Once handled the Error Codes can be cleared, allowing the
" main program to continue.
***************************************************************************
Select Case ReasonCode
    Case EIRC_??? :
        ...Error Handle Process...
        thisReceiveOptions.clearErrorCodes
    Case EIRC_??? :
        ...Error Handle Process...
        thisReceiveOptions.clearErrorCodes
    ...
    Case Else :
        Print EIRO_ERROR_TEXT & " " & Cstr(ReasonCode) & ". & TITLE_TEXT
End Select
End Sub

Figure 48 (Part 9 of 11). Web Client Agent for CICS DPL via MQSeries Service
Sub GlobalErrors(CallerText As String)
'*******************************************************************************************
'* Output error handler messages in a formatted message box.
'*******************************************************************************************
    Dim ErrorText As String

    Print CallerText

    ' Format the error text for the message box
    ErrorText = CallerText + Chr$(10)
    ErrorText = ErrorText + Chr$(10) + ERROR_NUMBER_TEXT + " = " + Str(Err)
    ErrorText = ErrorText + Chr$(10) + ERROR_DESCRIPTION_TEXT + " = " + Error$ 
    ErrorText = ErrorText + Chr$(10) + LINE_NUMBER_TEXT + " = " + Str(Erl)

    ' Output the error to message box and print line
    Print ErrorText & TITLE_TEXT

End Sub

Sub EISessionErrorEventHandler(ThisSession As EISession , ReasonCode As Long)
'*******************************************************************************************
'* Error event handler for the EISession Class. At this point we choose not to take
'* any corrective action, an error message is simply displayed in a message box.
'* Alternatively you could write a CASE statement to handle your chosen
'* errors. Once handled the Error Codes can be cleared, allowing the
'* main program to continue.
'* Note: this subroutine is registered as an error event handler for EISession
'* events but is also called following a failure in EISession creation in Queryopen
'*******************************************************************************************
    Dim StrText As String

    Select Case ReasonCode
        Case EIRC_??? :
            ...Error Handle Process...
        Case EIRC_??? :
            ...Error Handle Process...
        Case EIRC_SEC_DB_SYSERROR:
            StrText = EISESSION_ERROR_TEXT & " & Cstr(ReasonCode) & "." + Chr$(10)
            StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) 
                + Chr$(10)
            StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) 
                + Chr$(10)
            StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
            Print StrText TITLE_TEXT
        Case EIRC_DEFN_DB_SYSERROR:
            StrText = EISESSION_ERROR_TEXT & " & Cstr(ThisSession.ReasonCode) & "." + Chr$(10)
            StrText = StrText + SYSTEMERRORTEXT + " : " + Cstr(ThisSession.SystemErrorText) + Chr$(10)
            StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
            StrText = StrText + SECONDARYSYSTEM + " : " + Cstr(ThisSession.SecondarySystemErrorCode)
            Print StrText TITLE_TEXT
        Case Else :

Figure 48 (Part 10 of 11). Web Client Agent for CICS DPL via MQSeries Service

242 Lotus Notes and the MQSeries Enterprise Integrator
StrText = ESESSION_ERROR_TEXT & ": " & Cstr(ThisSession.ReasonCode)
Print StrText TITLE_TEXT
End Select

*******************************************************************************************
** Do not clear the error code, as this will allow error EILSX_ERROR to be
** caught by the generic error handler.
*******************************************************************************************
End Sub

Sub CleanUp

*******************************************************************************************
** This function is called when the agent is finished.
** Disconnect from MQSeries and delete the EI objects.
*******************************************************************************************
On Error Resume Next  ' Ignore any error as we need to shutdown regardless.

*******************************************************************************************
** Disconnect from MQSeries.
*******************************************************************************************
Call MyService.Disconnect()

*******************************************************************************************
** Delete the EI objects.
*******************************************************************************************
Delete MyReceiveOptions
Delete MySendOptions
Delete MySendMessage
Delete MyReceiveMessage
Delete MyService
Delete MySession

End Sub

Figure 48 (Part 11 of 11). Web Client Agent for CICS DPL via MQSeries Service
C.5 Web Client Agent for IMS via MQSeries Service

'Web Agent (IMS via MQSeries):
*******************************************************************************************
'* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator
'* ITSO Sample Application
'* Web Agent for the MQEI IMS via MQSeries Service
*******************************************************************************************
Option Public
Option Explicit

' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"

' Load the MQSeries Enterprise Integrator library
UseIsx "eilsx"

*******************************************************************************************
'* Define Enterprise Integrator objects.
*******************************************************************************************
Dim MySession As EISession
Dim MyService As EIService
Dim MySendMessage As EIMessage
Dim MyReceiveMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EISendOptions

*******************************************************************************************
'* Notes variables for form manipulation.
*******************************************************************************************
Dim Session As NotesSession
Dim Doc As NotesDocument
Dim MyDb As NotesDatabase
Dim ServName As NotesName

*******************************************************************************************
'* Wait time in milliseconds for a reply.
*******************************************************************************************
Const FIVE_SECONDS = 5000 ' 5 seconds

*******************************************************************************************
'* To hold values expected from the EI Definition database
*******************************************************************************************
Const SERVICE_NAME = "IMSMQ"
Const SEND_MESSAGE_NAME = "IMS_REQUEST"
Const RECEIVE_MESSAGE_NAME = "IMS_REPLY"

*******************************************************************************************

Figure 49 (Part 1 of 11). Web Client Agent for IMS via MQSeries Service
Figure 49 (Part 2 of 11). Web Client Agent for IMS via MQSeries Service
```
"* Definition of warning/error messages
**********************************************************************************
Const EIRC_CONNECTED_STRING = "An attempt was made to connect " _
+ "to a service when already connected. The connect is ignored and " _
+ "the program continues."
Const EIRC_NO_MSGAVAILABLE_STRING = "An attempt was made to " _
+ "receive a message but the target message was not found. " _
+ "Please, retry later."
Const EIRC_FIELD_TRUNCATED_STRING = "A length of a string value of " _
+ "an EIMessage field exceeds the length specified on the MQSeries EI Definition " _
+ "database. Correct the LotusScript to make sure that the string value " _
+ "being passed as the EIMessage field value does not exceed the " _
+ "predefined length."
Const EIRC_WRONG_MESSAGE_STRING = "A message was received that " _
+ "does not match the EIMessage object passed as a parameter on the " _
+ "EIService ReceiveMessage call."
Const UNRECOGNISED_STRING = "Unrecognized warning message :"
Const EISERVICE_WARNING = "Warning message received establishing " _
+ "MQService. ReasonCode :"
Const EISESSION_ERROR = "An error has been reported from the " _
+ "EISession Class. ReasonCode"
Const THISMESSAGE_ERROR = "An error has been reported from the " _
+ "EIMessage Class. ReasonCode :":
Const THISSSERVICE_ERROR = "An error has been reported from the " _
+ "EIService Class. ReasonCode :":
Const THISSSO_ERROR = "An error has been reported from the " _
+ "EISendOptions Class. ReasonCode :":
Const THISSRO_ERROR = "An error has been reported from the " _
+ "EIReceiveOptions Class. ReasonCode :":
**********************************************************************************
"* Definition of StatusFlag Values and Status Description
**********************************************************************************
Const SF_NO_MORE_REPLY_YET="No MQ Reply found"
Const SD_NO_MORE_REPLY_YET="Request has been submitted, but no reply has been received yet. " _
+ "Press button to retry!"
Const SF_NONE=""
Const SD_NONE=""
Const SF_PROCESSED="Processed"
```

Sub Initialize
"* SG24-2217 : Lotus Notes and the MQSeries Enterprise Integrator"
"* ITSO Sample Application -
"* Web Agent for the MQEI IMS via MQSeries Service
**********************************************************************************
"* This sample illustrates how a Lotus Script application can access an IMS
"* transaction
"* A LotusScript application writes to a MQSeries queue.
"* The MQSeries-IMS bridge gets the message and uses the OTMA

**Figure 49 (Part 3 of 11). Web Client Agent for IMS via MQSeries Service**

246  Lotus Notes and the MQSeries Enterprise Integrator
"* interface to pass the information to the IMS phone book application.
"* This application (IVTNO) processes the message and returns to the
"* MQSeries-IMS bridge task which puts a reply back to the LotusScript application
"* The LotusScript application gets the message and update the document.
"* This agent is activated when the "Submit" button is clicked.
"* NOTE: When you see a number enclosed in '<<' and '>>' characters, (eg. <<1>>)
"* it is referring to a step in the Web User Access (Three Tier Model) diagram
"* in the ITSO Redbook "Lotus Notes and the MQSeries Enterprise Integrator"
*******************************************************************************************
*******************************************************************************************
*******************************************************************************************
*******************************************************************************************

"* Initialize the EILSX objects. <<4>>
*******************************************************************************
*******************************************************************************************

'*******************************************************************************************
 On Error Goto GeneralErrorHandler
 On Error EILSX_ERROR Goto MQEIErroHandler

'*******************************************************************************

'*******************************************************************************************
 Set MySession = New EISession

' Check for errors
 If MySession.ReasonCode <> EIRC_NONE Then
   Call EISessionErrorEventHandler(MySession, MySession.ReasonCode)
   Call MySession.ClearErrorCodes()
 End If

'*******************************************************************************

'*******************************************************************************************
 Set MyService = MySession.CreateService(SERVICE_NAME)
 On Event EIError From MyService Call EIServiceErrorEventHandler

'*******************************************************************************

'*******************************************************************************************
 Set MySendMessage = MySession.CreateMessage(SEND_MESSAGE_NAME)
 On Event EIMessageError From MySendMessage Call EIMessageErrorEventHandler

'*******************************************************************************

' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
 If MySendMessage.CompletionCode = EICC_WARNING Then
   Select Case MySendMessage.ReasonCode
   Case EIRC_FIELD_TRUNCATED :
     Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE
   Case Else :
     Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE
   End Select
 End If

*******************************************************************************

Figure 49 (Part 4 of 11). Web Client Agent for IMS via MQSeries Service
Print UNRECOGNISED_STRING & " " & Cstr(MySendMessage.ReasonCode) WARNINGTITLE
End Select
Call MySendMessage.ClearErrorCodes
End If

'*****************************************************************************************
'* Create EI receive message'* <<4a>> defined in the EI Definition database).
'*****************************************************************************************
Set MyReceiveMessage = MySession.CreateMessage(RECEIVE_MESSAGE_NAME)
On Event EIIError From MyReceiveMessage Call EIMessageErrorEventHandler
If MyReceiveMessage.CompletionCode = EICC_WARNING Then
 Select Case MyReceiveMessage.ReasonCode
 Case EIRC_FIELD_TRUNCATED :
  Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE
 Case Else :
  Print UNRECOGNISED_STRING & " " & Cstr(MySendMessage.ReasonCode) WARNINGTITLE
 End Select
 Call MyReceiveMessage.ClearErrorCodes
 End If

'*****************************************************************************************
'* Create EI send message options'* <<4a>> defined in the EI Definition database).
'*****************************************************************************************
Set MySendOptions = MySession.CreateSendOptions
On Event EIIError From MySendOptions Call EISendOptionsErrorEventHandler

'* The MySendOptions.MessageType property is ignored by the MQEI LSX and*
'* is overridden internally to EIMT_REQUEST

'*****************************************************************************************
'* Create EI receive message options'* <<4a>> defined in the EI Definition database).
'*****************************************************************************************
Set MyReceiveOptions = MySession.CreateReceiveOptions
On Event EIIError From MyReceiveOptions Call EIReceiveOptionsErrorEventHandler
MyReceiveOptions.WaitType=EIWT_WAIT
MyReceiveOptions.WaitInterval=FIVE_SECONDS

'*****************************************************************************************
'* Set document variables'* <<4a>> defined in the EI Definition database).
'*****************************************************************************************
Set Session = New NotesSession
Set Doc=Session.DocumentContext
Set Mydb = Session.CurrentDatabase
Set ServName = New NotesName(session.userName)

'*****************************************************************************************
'* Connect to EI Service <<5>>'*****************************************************************************************
'*****************************************************************************************
Call MyService.Connect ' Connect to enterprise services

Figure 49 (Part 5 of 11). Web Client Agent for IMS via MQSeries Service

Lotus Notes and the MQSeries Enterprise Integrator
' Check for warnings (We have decided to ignore warnings for the purposes of this sample)
If MyService.CompletionCode = EICC_WARNING Then
    Select Case MyService.ReasonCode
    Case EIRC_CONNECTED :     
        Print EIRC_CONNECTED_STRING WARNINGTITLE
    Case Else :
        Print EISERVICE_WARNING & " " & Cstr(MyService.ReasonCode) WARNINGTITLE
    End Select
    Call MyService.ClearErrorCodes
End If

'*'......................................................................................................................
'*' Send the Message to the EI Service selected if StatusFlag is not
'*' SF_NO_MQ_REPLY_MSG_YET
'*' otherwise message has been sent already and just a receive has to take
'*' place
'*'......................................................................................................................
If Doc.GetFirstItem("StatusFlag").Text<>SF_NO_MQ_REPLY_MSG_YET Then
    '*' Setup and send message to IMS <<6>>
    '......................................................................................................................
    MySendMessage.Format = EIMTF_NONE
    MySendMessage.cmd = Doc.GetItemValue("I_CMD")
    MySendMessage.surname = Doc.GetItemValue("I_SNAME")
    MySendMessage.firstname = Doc.GetItemValue("I_FNAM")
    MySendMessage.tel = Doc.GetItemValue("I_TEL")
    MySendMessage.zip = Doc.GetItemValue("I_ZIP")

    '*' Send the message <<7>>
    '......................................................................................................................
    Call MyService.SendMessage ( MySendMessage, MySendOptions )
    '......................................................................................................................
    '*' Generated MQ message Id is stored in document for delayed
    '*' receive option
    '......................................................................................................................
    Doc.MQRequestMsgId=MySendOptions.Identifier
End If

'*'......................................................................................................................
'*' Receive the message <<8>>
'*'......................................................................................................................
'Identifier property must be set to receive the reply message to a given request message
'which is identified by the identifier stored in the MQRequestMsgId of the form
'otherwise MQEI would receive the first message in the queue
Call MyService.ReceiveMessage ( MyReceiveMessage, MyReceiveOptions )

Figure 49 (Part 6 of 11). Web Client Agent for IMS via MQSeries Service
Doc.StatusFlag=SF_NONE
Doc.StatusDescription=SD_NONE
If MyService.CompletionCode = EICC_WARNING Then
Select Case MyService.ReasonCode
Case EIRC_NO_MESSAGEAVAILABLE:
   Print EIRC_NO_MSG_AVAILABLE_STRING WARNINGTITLE
   ' Set Status Flag, so that user can retry later (delayed receive)
   Doc.StatusFlag=SF_NO_MQREPLY_YET
   Doc.StatusDescription=SD_NO_MQREPLY_YET
Case EIRC_FIELD_TRUNCATED:
   Print EIRC_FIELD_TRUNCATED_STRING WARNINGTITLE
Case EIRC_WRONGMESSAGE:
   Print EIRC_WRONG_MESSAGE_STRING WARNINGTITLE
Case Else:
   Print UNRECOGNISED_STRING & " " & MyService.ReasonCode WARNINGTITLE
End Select
Call MyService.ClearErrorCodes
Else
'****************************************************************************
' Update backend document with receive message <<9>>
'****************************************************************************
Doc.O_CMD=Trim(MyReceiveMessage.cmd)
Doc.O_SURNAMET=Trim(MyReceiveMessage.surname)
Doc.O_FIRST_NAME=Trim(MyReceiveMessage.firstname)
Doc.O_TEL=Trim(MyReceiveMessage.tel)
Doc.O_ZIP=Trim(MyReceiveMessage.zip)
Doc.O_MSG =Trim(MyReceiveMessage.msg)
Doc.StatusFlag=SF_PROCESSED ' Set status flag
End If
'****************************************************************************
' General LotusScript Error Handler
'****************************************************************************
Figure 49 (Part 7 of 11). Web Client Agent for IMS via MQSeries Service
GeneralErrorHandler:
Call GlobalErrors(LOTUSSCRIPT_ERROR_TEXT)
CleanUp
Exit Sub

'*****************************************************************************
'** Handle EILSX errors.
'*****************************************************************************
MQEISXErrorHandler:
Call GlobalErrors(EILSX_ERROR_TEXT)
CleanUp
Exit Sub
End Sub

Sub GlobalErrors(CallerText As String)
'*****************************************************************************
'** Output error handler messages in a formatted message box.
'*****************************************************************************
    Dim ErrorText As String

    Print CallerText

    ' Format the error text for the message box
    ErrorText = CallerText + Chr$(10)
    ErrorText = ErrorText + ERROR_NUMBER_TEXT + " = " + Str(Err)
    ErrorText = ErrorText + ERROR_DESCRIPTION_TEXT + " = " + Error$
    ErrorText = ErrorText + LINE_NUMBER_TEXT + " = " + Str(Erl)

    ' Output the error to message box and print line
    Print ErrorText & TITLE_TEXT
End Sub

Sub EISessionErrorEventHandler(ThisSession As EISession, ReasonCode As Long)
'*****************************************************************************
'** Error event handler for the EISession Class. At this point we choose not to take
'** any corrective action, an error message is simply displayed in a message box.
'**
'** Alternatively you could write a CASE statement to handle your chosen
'** errors. Once handled the Error Codes can be cleared, allowing the
'** main program to continue.
'** Note: this subroutine is registered as an error event handler for EISession
'** events but is also called following a failure in EISession creation in Queryopen
'*****************************************************************************
    Dim StrText As String

    Select Case ReasonCode
        Case EIRC_??? :
            ...Error Handle Process...
        Case EIRC_??? :
            ...Error Handle Process...
        ...Case EIRC_SEC_DB_SYSERROR:

Figure 49 (Part 8 of 11). Web Client Agent for IMS via MQSeries Service

Appendix C. Web Agents  251
StrText = EISESSION_ERROR_TEXT & "]" & Cstr(ThisSession.ReasonCode) & "]" + Chr$(10)
StrText = StrText + SYSTEMERRORTEXT + "]" + Cstr(ThisSession.SystemErrorText) + Chr$(10)
StrText = StrText + PRIMARYSYSTEM + "]" + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
StrText = StrText + SECONDARYSYSTEM + "]" + Cstr(ThisSession.SecondarySystemErrorCode)
Print StrText TITLE_TEXT

Case EIRC_DEFN_DB_SYSERROR:
    StrText = EISESSION_ERROR_TEXT & "]" & Cstr(ThisSession.ReasonCode) & "]" + Chr$(10)
    StrText = StrText + SYSTEMERRORTEXT + "]" + Cstr(ThisSession.SystemErrorText) + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + "]" + Cstr(ThisSession.PrimarySystemErrorCode) + Chr$(10)
    StrText = StrText + SECONDARYSYSTEM + "]" + Cstr(ThisSession.SecondarySystemErrorCode)
    Print StrText TITLE_TEXT

Case Else:
    StrText = EISESSION_ERROR_TEXT + "]" + Cstr(ThisSession.ReasonCode)
    Print StrText TITLE_TEXT
End Select

*******************************************************************************************

** Do not clear the error code, as this will allow error EILSX_ERROR to be caught by the generic error handler.**

*******************************************************************************************

End Sub

Sub CleanUp

** This function is called when the agent is finished.**

** Disconnect from MQSeries and delete the EI objects.**

On Error Resume Next ' Ignore any error as we need to shutdown regardless.

'******************************************************************************

" Disconnect from MQSeries.
******************************************************************************

Call MyService.Disconnect()

******************************************************************************

" Delete the EI objects.
******************************************************************************

Delete MyReceiveOptions
Delete MySendOptions
Delete MySendMessage
Delete MyReceiveMessage
Delete MyService
Delete MySession

End Sub

Sub EIIServiceErrorEventHandler (thisService As EIIService, ReasonCode As Long)

******************************************************************************

" Error Handler for the EIIService Class. At this point we choose not to take any corrective action, the error is simply displayed in a message box.

******************************************************************************

Figure 49 (Part 9 of 11). Web Client Agent for IMS via MQSeries Service
**errors. Once handled the Error Codes can be cleared, allowing the**
**main program to continue.**

*****************************************************************************
Dim StrText As String
Select Case ReasonCode
  Case EIRC_????:
    ...Error Handle Process...
    thisService.clearErrorCodes
  Case EIRC_????:
    ...Error Handle Process...
    thisService.clearErrorCodes
  Case EIRC_SERVICE_SYSERROR:
    ' SecondarySystemErrorCode and SystemErrorText are not set for this service
    StrText = THISSERVICE_ERROR + " " + Cstr(ReasonCode) + Chr$(10)
    StrText = StrText + "Service Name : " + ThisService.Name + Chr$(10)
    StrText = StrText + PRIMARYSYSTEM + " : " + Cstr(ThisService.PrimarySystemErrorCode)
    Print StrText TITLE_TEXT
  Case Else :
    StrText = THISSERVICE_ERROR + " " + Cstr(ReasonCode) + Chr$(10)
    StrText = StrText + "Service Name : " + ThisService.Name
    Print StrText TITLE_TEXT
End Select
End Sub
Sub EIMessageErrorEventHandler (thisMessage As EIMessage, ReasonCode As Long)
*****************************************************************************
  **Error Handler for the EIService Class. At this point we choose not to take**
  **any corrective action, the error is simply displayed in a message box.**
  **Alternatively you could write a CASE statement to handle your chosen**
  **errors. Once handled the Error Codes can be cleared, allowing the**
  **main program to continue.**
*****************************************************************************
Select Case ReasonCode
  Case EIRC_????:
    ...Error Handle Process...
    thisMessage.clearErrorCodes
  Case EIRC_????:
    ...Error Handle Process...
    thisMessage.clearErrorCodes
  Case Else :
    Print EIMESSAGE_ERROR_TEXT & " " & Cstr(ReasonCode) & ". Message Name: " & thisMessage.Name & ".", MB_OK, TITLE_TEXT
End Select
End Sub
Sub EISendOptionsErrorEventHandler (thisSendOptions As EISendOptions, ReasonCode As Long)
*****************************************************************************
  **Error Handler for the EIService Class. At this point we choose not to take**
  **any corrective action, the error is simply displayed in a message box.**
  **Alternatively you could write a CASE statement to handle your chosen**
  **errors. Once handled the Error Codes can be cleared, allowing the**
  **main program to continue.**
*****************************************************************************
Figure 49 (Part 10 of 11). Web Client Agent for IMS via MQSeries Service
Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
    thisSendOptions.clearErrorCode
  Case EIRC_??? :
    ...Error Handle Process...
    thisSendOptions.clearErrorCode
  ...
  Case Else :
    Print EISO_ERROR_TEXT & " " & CStr(ReasonCode) & "." & TITLE_TEXT
End Select
End Sub

Sub EIReceiveOptionsErrorHandler (thisReceiveOptions As EReceiveOptions, ReasonCode As Long)

  ** Error Handler for the EIService Class. At this point we choose not to take
  ** any corrective action, the error is simply displayed in a message box.
  **
  ** Alternatively you could write a CASE statement to handle your chosen
  ** errors. Once handled the Error Codes can be cleared, allowing the
  ** main program to continue.

  Select Case ReasonCode
  Case EIRC_??? :
    ...Error Handle Process...
    thisReceiveOptions.clearErrorCode
  Case EIRC_??? :
    ...Error Handle Process...
    thisReceiveOptions.clearErrorCode
  ...
  Case Else :
    Print EIRO_ERROR_TEXT & " " & CStr(ReasonCode) & "." & TITLE_TEXT
End Select
End Sub

Figure 49 (Part 11 of 11). Web Client Agent for IMS via MQSeries Service

254 Lotus Notes and the MQSeries Enterprise Integrator
Appendix D. CICS Program Listings

This appendix contains the CICS program listings used in our sample application.

D.1 VSAMSERV

Figure 50 shows the VSAMSERV program that is shipped with the IBM CICS gateway for Lotus Notes.

```
*******************************************************************************
* * *
* MODULE NAME   VSAMSERV.CCP *
* DESCRIPTIVE NAME CICS GATEWAY DEMO (ECI BACK-END) *
* Statement:   Licensed Materials - Property of IBM *
* 63H9790 *
* (C) Copyright IBM Corp. 1995,1996 *
* See Copyright Instructions. *
* All rights reserved. *
* U.S. Government Users Restricted Rights - *
* use, duplication or disclosure restricted *
* by GSA ADP Schedule Contract with IBM Corp. *
* *
* Status:       Version 2 Release 0 *
* Components:   VSAMSERV.CCP *
* NOTES: :- *
* VSAMSERV is the Back-End CICS application that gets called *
* from the ECI Front-End of the CICS Gateway to Lotus Notes *
* Sample program. It performs one of the File operations: *
* READ/WRITE/UPDATE/DELETE depending on the value specified in *
* the SERVER-STATE field of the COMMAREA. *
* *
*******************************************************************************
```

Figure 50 (Part 1 of 9). VSAMSERV Source
IDENTIFICATION DIVISION.
PROGRAM-ID. VSAMSERV.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-PC.
OBJECT-COMPUTER. IBM-PC.

*****************************************************************
DATA DIVISION.
WORKING-STORAGE SECTION.
* PUT IN FOR WRITEQ VVVV
01 WRITEQ-WORDS.
  03 OPERATING-SYSTEM PIC X(9) VALUE 'OS:'.
  03 PROGRAM-HEADER PIC X(9) VALUE 'PROGRAM: '.
  03 PROGRAM-NAME PIC X(9) VALUE 'CICSOS2 '.
  03 COMMENT-FIELD PIC X(40).

01 SERVER-IO-BUFFER.
  03 SERVER-STATE PIC 9.
  03 RETURN-VALUE PIC 9.
* Change the next line to COMP when you compile for CICS/ESA
  03 SERVER-KEY-SIZE PIC S9(4) COMP-5.
  03 SERVER-CUSTOMER-RECORD.
    05 ACCOUNTO PIC X(5).
    05 SURNAMEO PIC X(15).
    05 FIRST-NAMEO PIC X(10).
    05 ADDRESSO PIC X(45).
* Change the next line to COMP when you compile for CICS/ESA
    05 BALANCEO PIC S9(4) COMP-5.
  03 RESPONSES PIC X(50).

*************************
01 MISCEL-VARS.
* LENGTH-VAR made a COMP so its binary, needed for READ
  03 CUST-REC-LEN PIC S9(4) VALUE 77.
  03 LENGTH-VAR PIC S9(4) COMP-5 VALUE 0.
* Change the next line to COMP when you compile for CICS/ESA
  03 KEY-SIZE PIC S9(4) COMP-5.
  03 ACCOUNT-NUM PIC 9(5) VALUE 0.

Figure 50 (Part 2 of 9). VSAMSERV Source

256   Lotus Notes and the MQSeries Enterprise Integrator
03 CHAR-BUFFER PIC X(15).
03 SPLIT-BUFFER REDEFINES CHAR-BUFFER.
  05 BUFFER-ELEMENT OCCURS 15 TIMES PIC X.
03 ELEMENT-PTR PIC 99.
03 BUFFER-LEN PIC 99.
03 CHAR-FOUND PIC 9 VALUE 0.
03 BACKUP-CUSTOMER-RECORD.
  05 ACCOUNT-BAK PIC X(5).
  05 SURNAM-BAK PIC X(15).
  05 FIRST-NAME-BAK PIC X(10).
  05 ADDRESS-BAK PIC X(45).
* Change the next line to COMP when you compile for CICS/ESA
  05 BALANCE-BAK PIC S9(4) COMP-5.
03 RESPONSES-BAK PIC X(50).
* Change the next line to COMP when you compile for CICS/ESA
  03 LEN-PARM PIC S9(4) COMP-5.
  03 VALID-STATES.
    05 SERVER-STATE-CREATE PIC 9 VALUE 1.
    05 SERVER-STATE-READ PIC 9 VALUE 2.
    05 SERVER-STATE-UPDATE PIC 9 VALUE 3.
    05 SERVER-STATE-DELETE PIC 9 VALUE 4.
COPY DFHAID.

LINKAGE SECTION.
01 DFHCOMMAREA PIC X(131).
************************************************************
PROCEDURE DIVISION.

MAINLINE SECTION.

MOVE DFHCOMMAREA TO SERVER-IO-BUFFER
* Convert trailing LOW-VALUES to SPACES in the customer record
  INSPECT SURNAMEO REPLACING ALL LOW-VALUES BY SPACES
  INSPECT FIRST-NAMEO REPLACING ALL LOW-VALUES BY SPACES
  INSPECT ADDRESSO REPLACING ALL LOW-VALUES BY SPACES
* DEBUGGING CODE
  MOVE 'STARTING' TO COMMENT-FIELD
  PERFORM WRITEQ-TS
  MOVE SERVER-KEY-SIZE TO KEY-SIZE

Figure 50 (Part 3 of 9). VSAMSERV Source
EVALUATE SERVER-STATE
  WHEN SERVER-STATE-CREATE
    PERFORM CREATE-RECORD
  WHEN SERVER-STATE-READ
    PERFORM READ-RECORD
  WHEN SERVER-STATE-UPDATE
    PERFORM UPDATE-RECORD
  WHEN SERVER-STATE-DELETE
    PERFORM DELETE-RECORD
END-EVALUATE

MOVE SERVER-IO-BUFFER TO DFHCOMMAREA

* DEBUGGING CODE
  MOVE 'EXITING' TO COMMENT-FIELD
  PERFORM WRITEQ-TSEXEC
  EXEC CICS RETURN
END-EXEC.

MAINLINE-EXIT.
EXIT.

*****************************************************************
CREATE-RECORD SECTION.
* check for duplicate record in file
* calculate key size (temp support for detecting dup records
  MOVE LOW-VALUES TO CHAR-BUFFER
  MOVE SURNAMEO TO CHAR-BUFFER
  MOVE 15 TO ELEMENT-PTR
  PERFORM STRING-LENGTH
  MOVE BUFFER-LEN TO KEY-SIZE
  MOVE CUST-REC-LEN TO LENGTH-VAR
  MOVE LOW-VALUES TO BACKUP-CUSTOMER-RECORD
  EXEC CICS READ
    FILE('TECHALT')
    INTO(BACKUP-CUSTOMER-RECORD)
    LENGTH LENGTH-VAR
    RIDFLD SURNAMEO
    KEYLENGTH KEY-SIZE
    GENERIC
    NOHANDLE

Figure 50 (Part 4 of 9). VSAMSERV Source
IF SURNAMEO = SURNAME-BAK
    MOVE 14 TO EIBRESP
    PERFORM EVALUATE-RESP
    GO TO CREATE-RECORD-EXIT
END-IF

* get max record number
MOVE SERVER-CUSTOMER-RECORD TO BACKUP-CUSTOMER-RECORD
MOVE HIGH-VALUES TO ACCOUNTO
EXEC CICS STARTBR
    FILE('TECHBASE')
    RIDFLD(ACCOUNTO)
    GTEQ
    NOHANDLE
END-EXEC
EXEC CICS READPREV
    FILE('TECHBASE')
    INTO(SERVER-CUSTOMER-RECORD)
    LENGTH(LENGTH-VAR)
    RIDFLD(ACCOUNTO)
    NOHANDLE
END-EXEC
EXEC CICS ENDBR
EXEC CICS ENDR
    FILE('TECHBASE')
    NOHANDLE
END-EXEC

* if it is the first record to be added, set the a/c # to 05000
IF ACCOUNTO = HIGH-VALUES
    MOVE '05000' TO ACCOUNTO
END-IF

* add 1 and enter new record
MOVE ACCOUNTO TO ACCOUNT-NUM
ADD 1 TO ACCOUNT-NUM
MOVE BACKUP-CUSTOMER-RECORD TO SERVER-CUSTOMER-RECORD
MOVE ACCOUNT-NUM TO ACCOUNTO
MOVE 5 TO KEY-SIZE
MOVE CUST-REC-LEN TO LENGTH-VAR

Figure 50 (Part 5 of 9). VSAMSERV Source
EXEC CICS WRITE
   FILE('TECHBASE')
   FROM(SERVER-CUSTOMER-RECORD)
   LENGTH(LENGTH-VAR)
   RIDFLD(ACCOUNTO)
   KEYLENGTH(KEY-SIZE)
   NOHANDLE
END-EXEC
PERFORM EVALUATE-RESP.
CREATE-RECORD-EXIT.
EXIT.

*****************************************************************
READ-RECORD SECTION.
   MOVE CUST-REC-LEN TO LENGTH-VAR
   IF ACCOUNTO = LOW-VALUES
   EXEC CICS READ
      FILE('TECHALT')
      INTO(SERVER-CUSTOMER-RECORD)
      LENGTH(LENGTH-VAR)
      RIDFLD(SURNAMEO)
      KEYLENGTH(KEY-SIZE)
      GENERIC
      NOHANDLE
   END-EXEC
   ELSE
   EXEC CICS READ
      FILE('TECHBASE')
      INTO(SERVER-CUSTOMER-RECORD)
      LENGTH(LENGTH-VAR)
      RIDFLD(ACCOUNTO)
      NOHANDLE
   END-EXEC
   END-IF
PERFORM EVALUATE-RESP.
READ-RECORD-EXIT.
EXIT.

*****************************************************************
UPDATE-RECORD SECTION.
   MOVE SERVER-CUSTOMER-RECORD TO BACKUP-CUSTOMER-RECORD

Figure 50 (Part 6 of 9). VSAMSERV Source
MOVE CUST-REC-LEN TO LENGTH-VAR
EXEC CICS READ
  FILE('TECHBASE')
  INTO(SERVER-CUSTOMER-RECORD)
  LENGTH(LENGTH-VAR)
  RIDFLD(ACCOUNTO)
  UPDATE
  NOHANDLE
END-EXEC
IF EIBRESP = 0
  MOVE BACKUP-CUSTOMER-RECORD TO SERVER-CUSTOMER-RECORD
  MOVE CUST-REC-LEN TO LENGTH-VAR
  EXEC CICS RENWRITE
    FILE('TECHBASE')
    FROM(SERVER-CUSTOMER-RECORD)
    LENGTH(LENGTH-VAR)
    NOHANDLE
END-EXEC
ELSE
  PERFORM EVALUATE-RESP
END-IF
PERFORM EVALUATE-RESP.
UPDATE-RECORD-EXIT.
EXIT.
*******************************************************************
DELETE-RECORD SECTION.
MOVE CUST-REC-LEN TO LENGTH-VAR
EXEC CICS READ
  FILE('TECHBASE')
  INTO(SERVER-CUSTOMER-RECORD)
  LENGTH(LENGTH-VAR)
  RIDFLD(ACCOUNTO)
  UPDATE
  NOHANDLE
END-EXEC
IF EIBRESP = 0
  EXEC CICS DELETE
    FILE('TECHBASE')
    NOHANDLE
END-EXEC

Figure 50 (Part 7 of 9). VSAMSERV Source
ELSE
  PERFORM EVALUATE-RESP
END-IF
PERFORM EVALUATE-RESP.
DELETE-RECORD-EXIT.
EXIT.
*****************************************************************
EVALUATE-RESP SECTION.
EVALUATE EIBRESP
  WHEN DFHRESP (NORMAL)
    MOVE 0 TO RETURN-VALUE
  WHEN DFHRESP (DISABLED)
    MOVE 'File is out of action' TO RESPONSES
  WHEN 12
    MOVE 'File cannot be found' TO RESPONSES
  WHEN DFHRESP (DUPREC)
    MOVE 'Duplicate found, record NOT created' TO RESPONSES
  WHEN DFHRESP (NOTFND)
    MOVE 'Record cannot be found' TO RESPONSES
  WHEN DFHRESP (ILLOGIC)
    MOVE 'Illogic operation Hmm!' TO RESPONSES
  WHEN DFHRESP (INVREQ)
    MOVE 'Invalid request' TO RESPONSES
  WHEN DFHRESP (IOERR)
    MOVE 'IO error disk full or setup wrong !' TO RESPONSES
  WHEN DFHRESP (LENGERR)
    MOVE 'Invalid length Hmm!' TO RESPONSES
  WHEN DFHRESP (NOSPACE)
    MOVE 'Not enough disk space ! Hmm!' TO RESPONSES
END-EVALUATE.
EVALUATE-RESP-EXIT.

Figure 50 (Part 8 of 9). VSAMSERV Source
EXIT.
*****************************************************************
STRING-LENGTH SECTION.
* used to calculate keysize
MOVE ELEMENT-PTR TO BUFFER-LEN
PERFORM COUNT-LOW-VALS UNTIL ELEMENT-PTR = 1
GO TO STRING-LENGTH-EXIT.
COUNT-LOW-VALS.
  IF BUFFER-ELEMENT(ELEMENT-PTR) = LOW-VALUE OR SPACE
      SUBTRACT 1 FROM ELEMENT-PTR
  ELSE
      MOVE ELEMENT-PTR TO BUFFER-LEN
      MOVE 1 TO ELEMENT-PTR
  END-IF.
STRING-LENGTH-EXIT.
EXIT.
*****************************************************************
WRITEQ-TS SECTION.
* This section added to write to the CEBR0000 TS Queue at
* various points in the program so that you can prove the
* program is being used.
EXEC CICS WRITEQ TS
  QUEUE('CEBR0000')
  FROM(WRITEQ-WORDS)
  LENGTH(LENGTH OF WRITEQ-WORDS)
END-EXEC.
WRITEQ-TS-EXIT.
EXIT.

Figure 50 (Part 9 of 9). VSAMSERV Source

D.2 AMQSCIC1

Figure 51 on page 264 shows the header files and Figure 52 on page 265 shows the program listing for AMQSCIC1. The program is invoked by the MQ1 CICS transaction, which reads MQSeries messages from a queue and links to program VSAMSERV to process the messages.
#ifndef AMQSCIH0_DEFINED /* File not yet included? */
define AMQSCIH0_DEFINED /* Show file now included */
define TRUE 1
define FALSE 0
define WAIT_INTERVAL 30000
define APPL_NAME "AMQSCIC0"

/* Error messages */
#define ERROR_MSG_1 "An error has occurred in " 
"transaction %4.4s Task no. %07d " 
"on %8.8s %8.8s"
#define ERROR_MSG_2 "Error - Operation %-12.12s " 
"CompCode %1.1d Reason %4.4d " 
"Object %-48.48s"

/* Queue Names */
define DEAD_QNAME "SYSTEM.DEAD.LETTER.QUEUE"
define INPUT_QNAME "EI.INPUTQUEUE"
define OUTPUT_QNAME "EI.OUTPUTQUEUE"

Figure 51 (Part 1 of 2). AMQSCIH1 Header File
typedef struct tagAMQSCIC0_INPUT {
    MQXQH XmitHeader;
    char MessageData[2000];
} AMQSCIC0_INPUT;

typedef struct tagAMQSCIC0_DLQ {
    MQDLH DeadLetterHeader;
    char MessageData[2000];
} AMQSCIC0_DLQ;

#ifndef min
    #define min(a,b) (((a) < (b)) ? (a) : (b))
#endif

/* End of header file */

Figure 51 (Part 2 of 2). AMQSCIH1 Header File

/* Program name: AMQSCICS1.CCS */
/* */
/* Environment : CICS/2 Version 2.0; */
/* MQ Version 2.0.1; */
/* */
/* Description : Sample program to Get a message from a MQ Queue, */
/* link to a CICS program, VSAMSERV, to update */
/* a VSAM file and then PUT a response message on */

Figure 52 (Part 1 of 18). AMQSCIC1 CICS MQSeries Program
/* a MQ queue. */
/* */
/* Statement: Licensed Materials - Property of IBM */
/* */
/* 33H2205, 5622-908 */
/* 33H2267, 5765-623 */
/* 29H0990, 5697-176 */
/* (C) Copyright IBM Corp. 1994, 1995 */
/* */
*********************************************************************/
/* */
/*************************************************************/
/* */
/* Program logic */
/* */
/* */
/* main */
/* */
/* **** */
/* */
/* Establish CICS addressability */
/* */
/* Connect to default Queue Manager */
/* */
/* If connect unsuccessful */
/* */
/* Call RecordCallError */
/* */
/* Else */
/* */
/* Open the input queue. */
/* */
/* If open unsuccessful */
/* */
/* Call RecordCallError */
/* */
/* Else */
/* */
/* Set the MQGET call parameters */
/* */
/* Get a message from the input queue with wait */
/* */
/* Do while there are messages to be retrieved */
/* */
/* If expected message type */
/* */
/* Move message to CICS commarea */
/* */
/* Call ProcessMessage */
/* */
/* Else */
/* */
/* Call CheckQUnknownMsg */
/* */
/* End-if */
/* */
/* Reset the call parameters */
/* */
/* Get next message from the input queue with wait */
/* */
/* End-do */
/* */
/* If loop has ended for reason other than 'no */
/* message available' */
/* */
/* Call RecordCallError */
/* */

Figure 52 (Part 2 of 18). AMQSCIC1 CICS MQSeries Program

266 Lotus Notes and the MQSeries Enterprise Integrator
/* End-if */
/* Close the input queue */
/* If close unsuccessful */
/* Call RecordCallError */
/* End-if */
/* End-if */
/* Disconnect from the Queue Manager */
/* End-if */
/* */
/* Return to CICS */
/* */
/* */
/* ProcessMessage */
/* */
/* */
/* Link to CICS program VSAMSERV passing the commarea. */
/* Move returned commarea to the message. */
/* Set the MQPUT1 call parameters */
/* Put the message on the reply queue */
/* If put unsuccessful */
/* Call RecordCallError */
/* Call ForwardMsgToDLQ, to forward the */
/* message to the sample's dead-letter queue */
/* End-if */
/* */
/* Return to main function */
/* */
/* */
/* CheckQUnknownMsg */
/* */
/* */
/* CheckQUnknownMsg */
/* */
/* Call RecordCallError */
/* Call ForwardMsgToDLQ, to forward the */
/* message to the sample's dead-letter queue */
/* */
/* Return to main function */
/* */
/* */
/* RecordCallError */
/* */
/* Get the time from CICS */
/* Get CICS to format the date and time */
/* Call PackToLong to convert the packed decimal */

Figure 52 (Part 3 of 18). AMQSCIC1 CICS MQSeries Program
/* number to an unpacked number */
/* Build the output message data */
/* Write the call error message to CSMT TDQ */
/* Write the log error message to CSMT TDQ */
/* Return to main function */
/* */
/* ForwardMsgToDLQ */
/* ---------------- */
/* Set the MQPUT1 call parameters to enable the message */
/* to be put on the samples dead-letter queue */
/* If the message to be sent is longer than the buffer */
/* Set message length to the full buffer length */
/* End-if */
/* Put the message on the sample's dead-letter queue */
/* Call RecordCallError */
/* */
/* Return to main function */
/* */
/* */
/* PackToLong */
/* ------------ */
/* Convert the CICS packed decimal task number to a */
/* long integer */
/* */
/* Return to main function */
/* */
/* */

#include <stdio.h> /* used for standard I/O */

Figure 52 (Part 4 of 18). AMQSCIC1 CICS MQSeries Program
```c
#include <string.h> /* string functions */
#include <stdlib.h>
#include <ctype.h>
#include <cmqc.h> /* MQI header file */
#include <amqsci.h> /* Local definitions */
#include <eicomma.h> /* Local definitions */

MQLONG Hconn; /* Connection handle */
MQOBJ Hobj_InputQ; /* Object handle */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Qualifying reason */
MQOD ObjDesc = {MQOD_DEFAULT}; /* Object descriptor */
MQMD MsgDesc = {MQMD_DEFAULT}; /* Message descriptor */
MQLONG OpenOptions; /* Control the MQOPEN call */

MQGMO GetMsgOpts = {MQGMO_DEFAULT}; /* Get-Message Options */
MQLONG MsgBuffLen; /* Length of message buffer */

AMQSCIC0_INPUT MsgBuffer; /* Message structure */
MQLONG DataLen; /* Length of message */

AMQSCIC0_DLQ DLQBuffer; /* Message structure */
MQLONG DLQLen; /* Length of message */

MQPMO PutMsgOpts = {MQPMO_DEFAULT}; /* Put-Message Options */
MQLONG PutBuffLen; /* Length of message buffer */

MQCHAR48 QMName = "", /* Queue Manager Name */
MQCHAR48 InputQName = INPUT_QNAME; /* Input Queue Name */
MQLONG InputLength = MQ_Q_NAME_LENGTH;

ei_commarea_def ei_commarea; /* CICS commarea */

char compcode_string[11];
char* compcode_string_ptr;
char* dummyptr;
long dummylong;
char ErrorMsg[100]; /* Error Message Buffer */
char Operation[21]; /* Operation in which error occurred */
char ObjName[MQ_Q_NAME_LENGTH]; /* Name of Object */
```

Figure 52 (Part 5 of 18). AMQSCIC1 CICS MQSeries Program
int bConnected = FALSE;
int bBadMsg = FALSE;

cchar calledOK[28] = "MQ1 call successful ";
cchar connOK[28] = "MQ1 MQCONN successful ";
cchar openOK[28] = "MQ1 MQOPEN successful ";
cchar getOK[28] = "MQ1 MQGET successful ";
cchar put1OK[28] = "MQ1 MQPUT1 successful ";
cchar discOK[28] = "MQ1 MQDISC successful ";

/**************************************************************************/  
/* Function prototypes   */ 
/**************************************************************************/  
void ProcessMessage(void);
void CheckQUnknownMsg(void);
void RecordCallError(void);
void ForwardMsgToDLQ(void);
void PackToLong(unsigned char *source,
                   unsigned int source_size,
                   long int *target);

/**************************************************************************/  
/* The main function initializes and controls the program flow. */ 
/**************************************************************************/  
void main(void)
{
  /**************************************************************************/  
  /* Establish CICS addressability   */ 
  /**************************************************************************/  
  EXEC CICS ADDRESS EIB(dfheiptr);
  EXEC CICS WRITE OPERATOR TEXT(calledOK) TEXTLENGTH(27);

  /**************************************************************************/  
  /* Connect to the Queue Manager   */ 
  /**************************************************************************/  
  Figure 52 (Part 6 of 18). AMQSCIC1 CICS MQSeries Program
/******************************************************************/
MQCONN(QMName,
   &Hconn,
   &CompCode,
   &Reason);

/**************************************************************/
/* If we failed, then report error and exit */
/**************************************************************/
if (CompCode != MQCC_OK)
{
   strncpy(Operation, "MQCONN", sizeof(Operation));
   strncpy(ObjName, QMName, MQ_Q_NAME_LENGTH);
   RecordCallError();
   memset(compcode_string,0,10);
   sprintf(compcode_string,"%d",CompCode);
   strcpy(ei_commaarea.respons,strcat("Error on MQCONN ",compcode_string));
}
else
{
   EXEC CICS WRITE OPERATOR TEXT(connOK) TEXTLENGTH(27);
   bConnected = TRUE;
}

/**************************************************************/
/* Set up the name of the work queue */
/**************************************************************/
strncpy(ObjDesc.ObjectName, InputQName, MQ_Q_NAME_LENGTH);

/**************************************************************/
/* Initialize options and open the queue for input */
/**************************************************************/
if (CompCode == MQCC_OK)
{
   OpenOptions = MQOO_INPUT_SHARED +
   MQOO_SAVE_ALL_CONTEXT;

   MQOPEN(Hconn,
      &ObjDesc,
      OpenOptions,

Figure 52 (Part 7 of 18). AMQSCIC1 CICS MQSeries Program
&Hobj_InputQ, 
&CompCode, 
&Reason);

/******************************************************************/
/* Test the output from the open, if not OK then build an error */
/* message */
******************************************************************/
if (CompCode != MQCC_OK)
{
   strncpy(Operation, "MQOPEN", sizeof(Operation));
   strncpy(ObjName, ObjDescObjectName, MQQ_NAME_LENGTH);
   RecordCallError();
   memset(compcode_string,0,10);
   fprintf(compcode_string, "%d", CompCode);
   strcpy(ei_commarea, respons, strcat("Error on MQOPEN ", compcode_string));
} else
{
   EXEC CICS WRITE OPERATOR TEXT(openOK) TEXTLENGTH(27);
   /******************************************************************/
   /* Get and process messages */
   /******************************************************************/
   GetMsgOpts.Options = MQGMO_ACCEPT_TRUNCATED_MSG;
   MsgBuffLen = sizeof(MsgBuffer);
   memcpy(MsgDesc.MsgId, MQMI_NONE, sizeof(MsgDesc.MsgId));
   memcpy(MsgDesc.CorrelId, MQCI_NONE, sizeof(MsgDesc.CorrelId));
   /******************************************************************/
   /* Make the first get call outside the loop */
   /******************************************************************/
   MQGET(Hconn, 
      &Hobj_InputQ, 
      &MsgDesc, 
      &GetMsgOpts, 
      MsgBuffLen, 
      &MsgBuffer,

Figure 52 (Part 8 of 18). AMQSCIC1 CICS MQSeries Program
while (CompCode != MQCC_FAILED)
{
EXEC CICS WRITE OPERATOR TEXT(getOK) TEXTLENGTH(27);
memcpy(&ei_commarea, &MsgBuffer, sizeof(ei_commarea));

EXEC CICS SYNCPOINT;

MQGET(Hconn,
     Hobj_InputQ,&MsgDesc,&GetMsgOpts,MsgBuffLen,&MsgBuffer,&DataLen,&CompCode,&Reason);
} /* end loop */

Figure 52 (Part 9 of 18). AMQSCIC1 CICS MQSeries Program
if (Reason != MQRC_NO_MSG_AVAILABLE)
{
        strncpy(Operation, "MQGET", sizeof(Operation));
        strncpy(ObjName, ObjDescObjectName, MQ_Q_NAME_LENGTH);
        RecordCallError();
}
MQCLOSE(&Hconn, MQQCO_NONE, &CompCode, &Reason);
if (CompCode != MQCC_OK)
{
        strncpy(Operation, "MQCLOSE", sizeof(Operation));
        strncpy(ObjName, InputQName, MQ_Q_NAME_LENGTH);
        RecordCallError();
        memset(compcode_string, 0, 10);
        sprintf(compcode_string, "%d", CompCode);
        strcpy(ei_commarea.responses, strcat("Error on MQCLOSE ", compcode_string));
}
/* end else */

if (bConnected == TRUE)
{
        MQDISC(&Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK)
        {
                strncpy(Operation, "MQDISC", sizeof(Operation));
                strncpy(ObjName, QName, MQ_Q_MGR_NAME_LENGTH);
                RecordCallError();
                memset(compcode_string, 0, 10);
                sprintf(compcode_string, "%d", CompCode);
                strcpy(ei_commarea.responses, strcat("Error on MQDISC ", compcode_string));
        }
        else

Figure 52 (Part 10 of 18). AMQSCIC1 CICS MQSeries Program
EXEC CICS WRITE OPERATOR TEXT(discOK) TEXTLENGTH(27);

EXEC CICS RETURN;

void ProcessMessage(void)
{

EXEC CICS LINK PROGRAM ("VSAMSERV")
COMMAREA (&ei_commarea)
LENGTH(sizeof(ei_commarea));

memcpy(&MsgBuffer.MessageData, &ei_commarea, sizeof(ei_commarea));

strncpy(ObjDesc.ObjectName, MsgDesc.ReplyToQ, sizeof(ei_commarea));

Figure 52 (Part 11 of 18). AMQSCIC1 CICS MQSeries Program
MQ_Q_NAME_LENGTH);
/**
 * Use Value of ReplyToQMgr field in header of input message
 * as destination Qmgr for MQPUT1 command
 */
strncpy(ObjDesc.ObjectQMgrName,
        MsgDesc.ReplyToQMgr,
        MQ_Q_MGR_NAME_LENGTH);

/**************************
 * Set Reply MsgType
 **************************/
MsgDesc.MsgType=MQMT_REPLY;

/**************************
 * Use Value of MsgId field in header of input message
 * as CorrelId
 **************************/
memcpy(MsgDesc.CorrelId,
       MsgDesc.MsgId,
       sizeof(MsgDesc.MsgId));

/**************************
 * Null MsgId field
 **************************/
memcpy(MsgDesc.MsgId,
       MQMI_NONE,
       sizeof(MsgDesc.MsgId));

/**************************
 * Blank out ReplyToQ and ReplyToQMgr
 **************************/
strncpy(MsgDesc.ReplyToQ,
        "",
        sizeof(MsgDesc.ReplyToQ));
strncpy(MsgDesc.ReplyToQMgr,
        "",
        sizeof(MsgDesc.ReplyToQMgr));

Figure 52 (Part 12 of 18). AMQSCIC1 CICS MQSeries Program
/*************************************************************/
/* Set CICS PutApplType */
/* *************************************************************/
MsgDesc.PutApplType = MQAT_CICS;
MsgBuffLen = sizeof(ei_commarea);
PutBuffLen = MsgBuffLen;

MQPUT1(Hconn,
    &ObjDesc,
    &MsgDesc,
    &PutMsgOpts,
    PutBuffLen,
    &MsgBuffer.MessageData,
    &CompCode,
    &Reason);

if (CompCode != MQCC_OK)
{
    strncpy(Operation, "MQPUT1", sizeof(Operation));
    strncpy(ObjName, ObjDesc.ObjectName, MQ_Q_NAME_LENGTH);
    RecordCallError();
    bBadMsg = FALSE;
    ForwardMsgToDLQ();
    memset(compcode_string, 0, 10);
    strftime(compcode_string, "%d", CompCode);
    strcat(ei_commarea.respmsg, strcat("Error on MQPUT1 ", compcode_string));
}
else
{
    EXEC CICS WRITE OPERATOR TEXT (put1OK) TEXTLENGTH(27);
}

strncpy(ObjDesc.ObjectName, InputQName, MQ_Q_NAME_LENGTH);
return;

/*************************************************************/
/* This function handles unexpected messages by recording the */
Figure 52 (Part 13 of 18). AMQSCIC1 CICS MQSeries Program

Appendix D. CICS Program Listings 277
void CheckQUnknownMsg(void)
{
    strncpy(Operation, "BAD MSG", sizeof(Operation));
    strncpy(ObjName, ObjDesc.ObjectName, MQ_Q_NAME_LENGTH);
    RecordCallError();
    bBadMsg = TRUE;
    ForwardMsgToDLQ();
    return;
}

void RecordCallError(void)
{
    char Abstime[8];
    char Date[8];
    char Time[8];
    char Transaction[4];
    long TaskNum;
    short MsgLen;

    EXEC CICS ASKTIME
    ABSTIME(Abstime);
    EXEC CICS FORMATTIME
    ABSTIME(Abstime)
    DATE(Date) DATESEP
    TIME(Time) TIMESEP;
    strncpy(Transaction, dfheiptr->eibtrnid, sizeof(Transaction));
}

Figure 52 (Part 14 of 18). AMQSCIC1 CICS MQSeries Program
/*******************************************************************************
/* Call function to convert the packed decimal number into unpacked number */
/*******************************************************************************/
PackToLong((unsigned char *)dfheiptr->eibtaskn,
    sizeof(dfheiptr->eibtaskn), &TaskNum);

/*******************************************************************************/
/* Format first error message */
/*******************************************************************************/
MsgLen = (short) sprintf(ErrorMsg, ERROR_MSG_1, Transaction,
    TaskNum, Date, Time);
EXEC CICS WRITEQ TD QUEUE("CSMT") FROM(ErrorMsg) LENGTH(MsgLen);

/*******************************************************************************/
/* Format second error message */
/*******************************************************************************/
MsgLen = (short) sprintf(ErrorMsg, ERROR_MSG_2, Operation,
    CompCode, Reason, ObjName);
EXEC CICS WRITEQ TD QUEUE("CSMT") FROM(ErrorMsg) LENGTH(MsgLen);
return;
}

/*******************************************************************************
/* This function forwards a message to the sample's dead letter queue. */
/*******************************************************************************
void ForwardMsgToDLQ(void)
{
    PMQDLH pDLQHdr;

    /*******************************************************************************/
    /* Set up Put message options for dead letter queue */
    /*******************************************************************************/
    PutMsgOpts.Options = MQPMO_PASS_IDENTITY_CONTEXT;
    PutMsgOpts.Context = Hobj_InputQ;

    /*******************************************************************************/

Figure 52 (Part 15 of 18). AMQSCIC1 CICS MQSeries Program
/* Set up dead letter message data */
/*******************************/
PutBuffLen = 2000;
memcpy( DLQBuffer.MessageData,
   &MsgBuffer,
   (size_t) PutBuffLen);

/*******************************/
/* Set up dead letter message header */
/*******************************/
pDLQHdr = &DLQBuffer.DeadLetterHeader;
memcpy( pDLQHdr->StrucId , MQDLH_STRUC_ID , sizeof(MQCHAR4) );
pDLQHdr->Version = MQDLH_VERSION_1;
if (bBadMsg == FALSE)
{
    pDLQHdr->Reason = Reason;
    memcpy( pDLQHdr->DestQName,
        ObjDesc.ObjectName,
        MQ_Q_NAME_LENGTH );
}
else
{
    pDLQHdr->Reason = MQFB_XMIT_Q_MSG_ERROR;
    memset( pDLQHdr->DestQName, ' ', MQ_Q_NAME_LENGTH );
}
memset( pDLQHdr->DestQMgrName, ' ', MQ_Q_NAME_LENGTH );
pDLQHdr->Encoding = MsgDesc.Encoding;
pDLQHdr->CodedCharSetId = MsgDesc.CodedCharSetId;
memcpy( pDLQHdr->Format , MsgDesc.Format , MQ_FORMAT_LENGTH );
pDLQHdr->PutApplType = MQAT_CICS;
memcpy( pDLQHdr->PutApplName , APPL_NAME,
    MQ_PUT_APPL_NAME_LENGTH );

/*******************************/
/* Set up Object descriptor for dead letter queue */
/*******************************/
ObjDesc.ObjectType = MQOT_Q;
strncpy(ObjDescObjectName, DEAD_QNAME, MQ_Q_NAME_LENGTH);

Figure 52 (Part 16 of 18). AMQSCIC1 CICS MQSeries Program
/** */
/* Set up Message descriptor for dead letter queue */
/*****************************************************************************/
MsgDesc.Persistence = MQPER_PERSISTENCE_AS_Q_DEF;
memcpy(MsgDesc.Format,
MQFMT_DEAD_LETTER_HEADER,
MQ_FORMAT_LENGTH);
/*****************************************************************************/
/* Put the message to the dead letter queue */
/*****************************************************************************/
MQPUT1(Hconn,&ObjDesc,&MsgDesc,&PutMsgOpts,&PutBuffLen,&DLQBuffer,&CompCode,&Reason);
strncpy(Operation, "MQPUT TO DLQ", sizeof(Operation));
strncpy(ObjName, DEAD_QNAME, MQ_Q_NAME_LENGTH);
RecordCallError();
return;
}

/*****************************************************************************/
/* This function converts the CICS packed decimal task number to */
/* a long integer. */
/*****************************************************************************/
void PackToLong(unsigned char *source,
unsigned int source_size,
long int *target)
{
static const unsigned char lmask = 0x0f;
int i;
long int result = 0;
unsigned int high_nibble, low_nibble;

Figure 52 (Part 17 of 18). AMQSCIC1 CICS MQSeries Program
for (i = 0; i < source_size-1; i++)
{
    high_nibble = ((unsigned int)(source[i]>>4))&lmask;
    result *= 10;
    result += (long) high_nibble;

    low_nibble = (unsigned int) source[i]&lmask;
    result *= 10;
    result += (long) low_nibble;
}

high_nibble = ((unsigned int)(source[i]>>4))&lmask;
result *= 10;
result += (long) high_nibble;
low_nibble = (unsigned int) (source[i]&lmask);
if (low_nibble == 0xd)
{
    result *= -1;
}
*target = result;
return;
}

Figure 52 (Part 18 of 18). AMQSCIC1 CICS MQSeries Program

D.3 TECHPROG

Figure 53 shows the TECHPROG program that is shipped with the IBM CICS gateway for Lotus Notes.

*************************************************************************
* MODULE NAME TECHPROG.CCP *
* DESCRIPTIVE NAME CICS GATEWAY DEMO (3270 FRONT-END) *
* Statement: Licensed Materials - Property of IBM *
* 63H9790 *
*************************************************************************

Figure 53 (Part 1 of 14). TECHPROG Source
IDENTIFICATION DIVISION.
PROGRAM-ID. TECHPROG.

ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
* PUT IN FOR WRITEQ VVVV
01 WRITEQ-WORDS.
  03 OPERATING-SYSTEM PIC X(9) VALUE 'OS:'.
  03 PROGRAM-HEADER PIC X(9) VALUE 'PROGRAM: '.
  03 PROGRAM-NAME PIC X(9) VALUE 'TECHPROG '.
  03 COMMENT-FIELD PIC X(40).

01 WS-VALUES.
  05 MESSAGE-TO-SEND PIC X(20).
  05 RESP-FIELD PIC S9(8) COMP-5.
  05 PROGRAM-STATUS PIC S9(8) COMP-5.
  05 CREATE-REQUEST PIC 9 VALUE 1.
  05 READ-REQUEST PIC 9 VALUE 2.
  05 UPDATE-REQUEST PIC 9 VALUE 3.
  05 DELETE-REQUEST PIC 9 VALUE 4.
  05 RM-OS2-PROG PIC X(8) VALUE 'VSAMSERV'.

01 ACCOUNT-ADDED-MESSAGE.
  05 FILLER PIC X(36)
    VALUE 'Account created with account number '.

Figure 53 (Part 2 of 14). TECHPROG Source
05 ACCOUNT-NUMBER-ADDED PIC 9(5).

01 NO-RESOURCE-MANAGER-MESSAGE PIC X(36)
   VALUE 'TECH: No resource Manager present!!'.

01 WS-COMMAREA.
   05 COMMAREA-REQUEST PIC 9.
   05 COMMAREA-RETURN-VALUE PIC 9.
   05 COMMAREA-KEY-SIZE PIC S9(4) COMP-5.
   05 COMMAREA-ACCOUNT-RECORD.
      10 COMMAREA-ACCOUNT PIC X(5).
      10 COMMAREA-SURNAME PIC X(15).
      10 COMMAREA-FORENAME PIC X(10).
      10 COMMAREA-ADDR1 PIC X(15).
      10 COMMAREA-ADDR2 PIC X(15).
      10 COMMAREA-ADDR3 PIC X(15).
      10 COMMAREA-BALANCE PIC S9(4) COMP-5.
   05 COMMAREA-RESPONSES PIC X(50).

COPY TECHMS.
COPY DFHAID.

LINKAGE SECTION.
01 DFHCOMMAREA PIC X.

PROCEDURE DIVISION.
* DEBUGGING CODE
   MOVE 'STARTING' TO COMMENT-FIELD
   PERFORM WRITEQ-TS
   MOVE LOW-VALUES TO T000I T001I T002I T003I T004I T005I
       T006I T007I.
   EXEC CICS SEND CONTROL
       ERASE
   END-EXEC.

***********************************************************
* Main Program loop starts here....  *
***********************************************************
MAIN-PROCESS-LOOP.
   MOVE ' ' TO OPTIONO.

Figure 53 (Part 3 of 14). TECHPROG Source
EXEC CICS SEND MAP('T001')
  MAPSET('TECHMS')
  FREEKB ERASEAUP
END-EXEC.
EXEC CICS RECEIVE MAP('T001')
  MAPSET('TECHMS')
  NOHANDLE
END-EXEC.
MOVE ' ' TO MESS10.
EVALUATE EIBAID
  WHEN DFHPF3 PERFORM EXIT-TRANSACTION
  WHEN DFHPF12 PERFORM EXIT-TRANSACTION
  WHEN DFHCLEAR CONTINUE
  WHEN DFHENTER
    IF OPTIONL = 0
      MOVE 'You must enter an option' TO MESS10
    ELSE
      EVALUATE OPTIONI
        WHEN '1'
          PERFORM PROCESS-SHOW THRU
          PROCESS-SHOW-EXIT
          EXEC CICS SEND CONTROL
          ERASE
          FREEKB
        END-EXEC
        WHEN '2'
          PERFORM PROCESS-ADD THRU PROCESS-ADD-EXIT
          EXEC CICS SEND CONTROL
          ERASE
          FREEKB
        END-EXEC
        WHEN '3'
          PERFORM PROCESS-UPDATE THRU
          PROCESS-UPDATE-EXIT
          EXEC CICS SEND CONTROL
          ERASE
          FREEKB
        END-EXEC
        WHEN '4'
          PERFORM PROCESS-DELETE THRU
          EXEC CICS SEND CONTROL
          ERASE
          FREEKB
        END-EXEC
Figure 53 (Part 4 of 14). TECHPROG Source
PROCESS-DELETE-EXIT
EXEC CICS SEND CONTROL
   ERASE
   FREEKB
END-EXEC
WHEN OTHER
   MOVE 'Invalid option chosen' to MESS1O
END-EVALUATE
END-IF
WHEN OTHER
   MOVE 'Invalid function key' TO MESS1O
END-EVALUATE.
GO TO MAIN-PROCESS-LOOP.
MAIN-PROCESS-LOOP-EXIT.
EXEC CICS RETURN
END-EXEC.
* Main Program loop ENDS here....

***********************************************************
* Process-Update Starts here... ****************************
***********************************************************
PROCESS-UPDATE.
EXEC CICS SEND CONTROL
   ERASE
END-EXEC.
PROCESS-UPDATE-LOOP.
MOVE LOW-VALUES TO ACC7O.
EXEC CICS SEND MAP('T007')
   MAPSET('TECHMS')
   ERASEAUP
   FREEKB
END-EXEC.
EXEC CICS RECEIVE MAP('T007')
   MAPSET('TECHMS')
   NOHANDLE
END-EXEC.
MOVE ' ' TO MESS7O.
EVALUATE EIBAID
   WHEN DFHCLEAR CONTINUE
   WHEN DFHPF3 GO TO PROCESS-UPDATE-EXIT

Figure 53 (Part 5 of 14). TECHPROG Source
WHEN DFHPF12 PERFORM EXIT-TRANSACTION
WHEN DFHENTER
  IF ACC7L = 0
    MOVE 'You must enter an account number for update' TO MESS70
  ELSE
    MOVE ACC7I TO COMMAREA-ACCOUNT
    MOVE 1 TO COMMAREA-RETURN-VALUE
    MOVE READ-REQUEST TO COMMAREA-REQUEST
    PERFORM LINK-BACKEND
    IF COMMAREA-RETURN-VALUE NOT = 0
      MOVE COMMAREA-RESPONSES TO MESS70
    ELSE
      PERFORM UPDATE-ACCOUNT THRU
      UPDATE-ACCOUNT-EXIT
      EXEC CICS SEND CONTROL
      ERASE
    END-EXEC
  END-IF
END-EVALUATE.
GO TO PROCESS-UPDATE-LOOP.
PROCESS-UPDATE-EXIT.
EXIT.
* Process-Update ends here...

*****************************************************************************
* Start of Link to Backend.... ***************************************
*****************************************************************************

LINK-BACKEND.
  MOVE 'ABOUT TO LINK TO BACKEND.' TO COMMENT-FIELD
  PERFORM WRIEQ-7S
  EXEC CICS LINK PROGRAM(RM-OS2-PROG)
    COMMAREA(WS-COMMAREA)
    LENGTH(LENGTH OF WS-COMMAREA)
    NOHANDLE
  END-EXEC
  IF EIBRESP NOT = 0
    MOVE 'BACKEND NOT AVAILABLE' TO COMMAREA-RESPONSES
  END-IF
* DEBUGGING CODE
  MOVE 'RETURNED FROM BACKEND' TO COMMENT-FIELD
  PERFORM WRITEQ-TS

  EXEC CICS SYNCPOINT
  END-EXEC.
* End of Link to Backend....  *

******************************************************************************
* End of this transaction, put out message and die...  *
******************************************************************************
EXIT-TRANSACTION.
  MOVE 'EXITING.....' TO COMMENT-FIELD
  PERFORM WRITEQ-TS

  MOVE 'TECH: Session Over' to MESSAGE-TO-SEND.
  EXEC CICS SEND CONTROL
      ERASE
      FREEKB
  END-EXEC.
  EXEC CICS SEND FROM(MESSAGE-TO-SEND)
      LENGTH(LENGTH OF MESSAGE-TO-SEND)
  END-EXEC.
  EXEC CICS RETURN
  END-EXEC.
* End of End of this transaction, put out message and DIE *

******************************************************************************
* SHOW-ACCOUNT start....  *
******************************************************************************
SHOW-ACCOUNT.
  MOVE COMMAREA-SURNAME TO SUR3O.
  MOVE COMMAREA-FORENAME TO FOR3O.
  MOVE COMMAREA-ADDR1 TO ADD13O.
  MOVE COMMAREA-ADDR2 TO ADD23O.
  MOVE COMMAREA-ADDR3 TO ADD33O.
  MOVE COMMAREA-ACCOUNT TO ACC3O.
  MOVE COMMAREA-BALANCE TO BAL3O.
SHOW-ACCOUNT-LOOP.
  EXEC CICS SEND CONTROL

Figure 53 (Part 7 of 14). TECHPROG Source
ERASE

END-EXEC.

EXEC CICS SEND MAP('T003')
   MAPSET('TECHMS')
FREEKB

END-EXEC.

EXEC CICS RECEIVE MAP('T003')
   MAPSET('TECHMS')
   NOHANDLE

END-EXEC.

EVALUATE EIBAID
   WHEN DFHPF12 PERFORM EXIT-TRANSACTION
   WHEN DFHENTER GO TO SHOW-ACCOUNT-EXIT
   WHEN DFHPF3 GO TO SHOW-ACCOUNT-EXIT
   WHEN DFHCLEAR
      MOVE COMMAREA-SURNAME TO SUR3O
      MOVE COMMAREA-FORENAME TO FOR3O
      MOVE COMMAREA-ADDR1 TO ADD13O
      MOVE COMMAREA-ADDR2 TO ADD23O
      MOVE COMMAREA-ADDR3 TO ADD33O
      MOVE COMMAREA-ACCOUNT TO ACC3O
      MOVE COMMAREA-BALANCE TO BAL3O
   END-EVALUATE.
   GO TO SHOW-ACCOUNT-LOOP.
SHOW-ACCOUNT-EXIT.
EXIT.
* SHOW-ACCOUNT end.... *

******************************************************************************
* PROCESS-SHOW start.... *
******************************************************************************

PROCESS-SHOW.
   EXEC CICS SEND CONTROL
      ERASE

END-EXEC.

PROCESS-SHOW-LOOP.
   MOVE LOW-VALUES to SUR2O.
   MOVE LOW-VALUES to ACC2O.
   EXEC CICS SEND MAP('T002')
      MAPSET('TECHMS')

Figure 53 (Part 8 of 14). TECHPROG Source
FREEKB
ERASE AUP
END-EXEC.
EXEC CICS RECEIVE MAP('T002')
   MAPSET('TECHMS')
   NOHANDLE
END-EXEC.
MOVE ' ' TO MESS2O.
EVALUATE EIBAID
   WHEN DFHENTER
      IF ((SUR2L NOT = 0) and (ACC2L NOT = 0))
         MOVE 'Not valid to enter data in BOTH surname and account'
         TO MESS2O
      ELSE
         IF ((SUR2L = 0) AND (ACC2L = 0))
            MOVE 'You must enter either a Surname OR an Account number'
            TO MESS2O
         ELSE
            MOVE SUR2I TO COMMAREA-SURNAME
            MOVE ACC2I TO COMMAREA-ACCOUNT
            MOVE SUR2L TO COMMAREA-KEY-SIZE
            MOVE 1 TO COMMAREA-RETURN-VALUE
            MOVE READ-REQUEST TO COMMAREA-REQUEST
            PERFORM LINK-BACKEND
            IF COMMAREA-RETURN-VALUE NOT = 0
               MOVE COMMAREA-RESPONSES TO MESS2O
            ELSE
               PERFORM SHOW-ACCOUNT THRU
               SHOW-ACCOUNT-EXIT
               EXEC CICS SEND CONTROL
               ERASE
            END-EXEC
         END-IF
      END-IF
   WHEN DFHPF3
      GO TO PROCESS-SHOW-EXIT
   WHEN DFHPF12
      PERFORM EXIT-TRANSACTION
   WHEN DFHCLEAR
      CONTINUE
   WHEN OTHER
      MOVE 'Invalid function key' TO MESS2O

Figure 53 (Part 9 of 14). TECHPROG Source
END-EVALUATE.
GO TO PROCESS-SHOW-LOOP.
PROCESS-SHOW-EXIT.
EXIT.

* PROCESS-SHOW end.... *

***********************************************************************
* PROCESS-ADD start... ************************************************

PROCESS-ADD.
EXEC CICS SEND CONTROL
ERASE
END-EXEC.
MOVE '' TO SUR4O.
MOVE '' TO FOR4O.
MOVE '' TO ADD14O.
MOVE '' TO ADD24O.
MOVE '' TO ADD34O.
MOVE '' TO BAL4O.
PROCESS-ADD-LOOP.
EXEC CICS SEND MAP('T004')
   MAPSET('TECHMS')FREEKB
END-EXEC.
EXEC CICS RECEIVE MAP('T004')
   MAPSET('TECHMS')NOHANDLE
END-EXEC.
MOVE '' TO MESS4O.
EVALUATE EIBAID
WHEN DFHENTER
   IF (((SUR4L = 0) or (SUR4I = LOW-VALUE) OR (SUR4I = ' '))
      MOVE 'You must enter a Surname' TO MESS4O
ELSE
   MOVE SUR4I TO COMMAREA-SURNAME
   MOVE FOR4I TO COMMAREA-FORENAME
   MOVE ADD14I TO COMMAREA-ADDR1
   MOVE ADD24I TO COMMAREA-ADDR2
   MOVE ADD34I TO COMMAREA-ADDR3
   MOVE BAL4I TO COMMAREA-BALANCE

Figure 53 (Part 10 of 14). TECHPROG Source
MOVE CREATE-REQUEST TO COMMAREA-REQUEST
MOVE 1 TO COMMAREA-RETURN-VALUE
PERFORM LINK-BACKEND
IF COMMAREA-RETURN-VALUE NOT = 0
   MOVE COMMAREA-RESPONSES TO MESS4O
ELSE
   MOVE COMMAREA-ACCOUNT TO
        ACCOUNT-NUMBER-ADDED
   MOVE ACCOUNT-ADDED-MESSAGE
        TO MESS4O
END-IF
END-IF
WHEN DFHPF3        GO TO PROCESS-ADD-EXIT
WHEN DFHPF12 PERFORM EXIT-TRANSACTION
WHEN DFHCLEAR
   MOVE ' ' to SUR4O
   MOVE ' ' to FOR4O
   MOVE ' ' to ADD14O
   MOVE ' ' TO ADD24O
   MOVE ' ' TO ADD34O
   MOVE ' ' TO BAL4O
WHEN OTHER
   MOVE 'Invalid function key' TO MESS4O
END-EVALUATE.
GO TO PROCESS-ADD-LOOP.
PROCESS-ADD-EXIT.
EXIT.
* PROCESS-ADD end... *

***********************************************************************
* PROCESS-DELETE start... ***********************************
***********************************************************************
PROCESS-DELETE.
   EXEC CICS SEND CONTROL
        ERASE
   END-EXEC.
PROCESS-DELETE-LOOP.
   MOVE LOW-VALUES to ACC5O.
   EXEC CICS SEND MAP('T005')
         MAPSET('TECHMS')

Figure 53 (Part 11 of 14). TECHPROG Source
ERASEAUP
FREEKB
END-EXEC.
EXEC CICS RECEIVE MAP('T005')
  MAPSET('TECHMS')
  NOHANDLE
END-EXEC.
MOVE '' TO MESS50.
EVALUATE EIBAID
  WHEN DFHCLEAR CONTINUE
  WHEN DFHPF3 GO TO PROCESS-DELETE-EXIT
  WHEN DFHPF12 PERFORM EXIT-TRANSACTION
  WHEN DFHENTER
    IF ACC5L = 0
      MOVE 'You must enter an account number for deletion'
      TO MESS50
    ELSE
      MOVE ACC5I TO COMMAREA-ACCOUNT
      MOVE DELETE-REQUEST TO COMMAREA-REQUEST
      MOVE 1 TO COMMAREA-RETURN-VALUE
      PERFORM LINK-BACKEND
      IF COMMAREA-RETURN-VALUE NOT = '0'
        MOVE COMMAREA-RESPONSES TO MESS50
      ELSE
        MOVE 'Account deleted' TO MESS50
      END-IF
    END-IF
END-EVALUATE.
GO TO PROCESS-DELETE-LOOP.
PROCESS-DELETE-EXIT.
EXIT.
* PROCESS-DELETE end...  *

********************************************************************
* UPDATE ACCOUNT start...  *
********************************************************************
UPDATE-ACCOUNT.
  EXEC CICS SEND CONTROL
  ERASE

Figure 53 (Part 12 of 14). TECHPROG Source
UPDATE-ACCOUNT-LOOP.
EXEC CICS SEND MAP('T006')
   MAPSET('TECHMS')
FREEKB
END-EXEC.
EXEC CICS RECEIVE MAP('T006')
   MAPSET('TECHMS')
NOHANDLE
END-EXEC.
MOVE ' ' TO MESS6O.
EVALUATE EIBAID
WHEN DFHCLEAR
   MOVE 'Modifications undone' TO MESS6O
   MOVE COMMAREA-SURNAME TO SUR6O
   MOVE COMMAREA-FORENAME TO FOR6O
   MOVE COMMAREA-ADDR1 TO ADD16O
   MOVE COMMAREA-ADDR2 TO ADD26O
   MOVE COMMAREA-ADDR3 TO ADD36O
   MOVE COMMAREA-ACCOUNT TO ACC6O
   MOVE COMMAREA-BALANCE TO BAL6O
WHEN DFHPF3
   MOVE 'Updated Canceled' TO MESS7O
   GO TO UPDATE-ACCOUNT-EXIT
WHEN DFHPF12 PERFORM EXIT-TRANSACTION
WHEN DFHENTER
   MOVE SUR6I TO COMMAREA-SURNAME
   MOVE FOR6I TO COMMAREA-FORENAME
   MOVE ADD16I TO COMMAREA-ADDR1
   MOVE ADD26I TO COMMAREA-ADDR2
   MOVE ADD36I TO COMMAREA-ADDR3
   MOVE BAL6I TO COMMAREA-BALANCE
   MOVE UPDATE-REQUEST to COMMAREA-REQUEST

Figure 53 (Part 13 of 14). TECHPROG Source
MOVE 1 TO COMMAREA-RETURN-VALUE
PERFORM LINK-BACKEND
IF COMMAREA-RETURN-VALUE NOT = 0
   MOVE COMMAREA-RESPONSES TO MESS6O
ELSE
   MOVE 'Account Updated' TO MESS7O
   GO TO UPDATE-ACCOUNT-EXIT
END-IF
END-EVALUATE.
GO TO UPDATE-ACCOUNT-LOOP.
UPDATE-ACCOUNT-EXIT.
EXIT.

* UPDATE ACCOUNT start... *

***************
* This section added to write to the CEBR0000 TS Queue *
* at various points in the program so that you can *
* prove the program is being used. *
***************
WRITEQ-TS SECTION.
   EXEC CICS WRITEQ TS
      QUEUE('CEBR0000')
      FROM(WRITEQ-WORDS)
      LENGTH(LENGTH OF WRITEQ-WORDS)
   END-EXEC.

WRITEQ-TS-EXIT.
EXIT.

Figure 53 (Part 14 of 14). TECHPROG Source
Figure 53 on page 282 shows the TECHMS BMS map that is shipped with the IBM CICS gateway for Lotus Notes.

**********************************************************************
* *                       TECHMS.BMS                             *
* * DESCRIPTIVE NAME      MAP FOR CICS GATEWAY SAMPLE PROGRAM    *
* * Statement:            Licensed Materials - Property of IBM    *
* * 63H9790                *                                    *
* (c) Copyright IBM Corp. 1995,1996                             *
* * See Copyright Instructions.                                  *
* * All rights reserved.                                       *
* * U.S. Government Users Restricted Rights - use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp. *
* * Status: Version 2 Release 0                                 *
* *
**********************************************************************

Figure 54 (Part 1 of 8). TECHMS BMS Source
INITIAL='Account Manager: Main Menu'

DFHMDF POS=(6,20), LENGTH=2, ATTRB=(PROT, NORM), INITIAL='1.'
INITIAL='Show account details'

DFHMDF POS=(8,20), LENGTH=2, ATTRB=(PROT, NORM), INITIAL='2.'
INITIAL='Add an account'

DFHMDF POS=(10,20), LENGTH=2, ATTRB=(PROT, NORM), INITIAL='3.'
INITIAL='Update an account'

DFHMDF POS=(12,20), LENGTH=2, ATTRB=(PROT, NORM), INITIAL='4.'
INITIAL='Delete an account'

DFHMDF POS=(17,9), LENGTH=38, ATTRB=(PROT, BRT), COLOR=GREEN,
INITIAL='Please select an option (1,2,3 or 4) :'

OPTION DFHMDF POS=(17,48), LENGTH=1, JUSTIFY=(LEFT, ZERO),
ATTRB=(UNPROT, NUM, NORM, IC), HILIGHT=UNDERLINE

MESS1 DFHMDF POS=(22,1), LENGTH=79, ATTRB=(PROT, NORM), COLOR=RED
INITIAL='-----------------------------------------------X--------------------------------'

----------'PF3=Exit'--------------------------'

--------------'PF12=Exit'

Appendix D. CICS Program Listings

Figure 54 (Part 2 of 8). TECHMS BMS Source
Figure 54 (Part 3 of 8). TECHMS BMS Source
FOR3  DFHMDF POS=(10,44),LENGTH=10,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(10,55),LENGTH=1,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(12,20),LENGTH=8,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM),INITIAL='Account:'
ACC3  DFHMDF POS=(12,44),LENGTH=5,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(12,50),LENGTH=1,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(14,20),LENGTH=8,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM),INITIAL='Address:'
ADD13 DFHMDF POS=(14,44),LENGTH=15,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(14,60),LENGTH=1,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(15,44),LENGTH=15,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
ADD23 DFHMDF POS=(15,44),LENGTH=15,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(15,60),LENGTH=1,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
ADD33 DFHMDF POS=(16,44),LENGTH=15,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(16,60),LENGTH=1,JUSTIFY=(LEFT,ZERO), X
  ATTRB=(PROT,NORM)
   DFHMDF POS=(18,20),LENGTH=8,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM),INITIAL='Balance:'
BAL3  DFHMDF POS=(18,44),LENGTH=4,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
   DFHMDF POS=(18,49),LENGTH=1,JUSTIFY=(LEFT,ZERO), X
   ATTRB=(PROT,NORM)
MESS3 DFHMDF POS=(22,1),LENGTH=79,ATTRB=(PROT,NORM),COLOR=RED
   DFHMDF POS=(23,1),LENGTH=79,ATTRB=(PROT,NORM),COLOR=YELLOW, X
   INITIAL='-----------------------------------------------X--------------------------------'
   DFHMDF POS=(24,1),LENGTH=16,ATTRB=(PROT,NORM),COLOR=YELLOW, X
   INITIAL='PF3/Enter=Return'
   DFHMDF POS=(24,71),LENGTH=9,ATTRB=(PROT,NORM),COLOR=YELLOW, X
   INITIAL='PF12=Exit'
T004 DFHMDF SIZE=(24,80),LINE=1,COLUMN=1,COLOR=NEUTRAL
   DFHMDF POS=(1,25),LENGTH=30,ATTRB=(PROT,NORM),HILIGHT=REVERSE, X

Figure 54 (Part 4 of 8). TECHMS BMS Source
Figure 54 (Part 6 of 8). TECHMS BMS Source
Figure 54 (Part 7 of 8). TECHMS BMS Source
Figure 54 (Part 8 of 8). TECHMS BMS Source
Appendix E. CICS Definitions

This appendix contains the CICS Definitions required to use our sample application.

E.1 Program Control Table Entries

<table>
<thead>
<tr>
<th>Update</th>
<th>Add</th>
<th>View</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit</td>
<td>Help</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FAAPCT2  Program Control Table-1

Transaction Code: MQ1
Group Name: SAMPLE
Program Name: AMQSCIC1

Secure: N (Y or N)
Can Be Purged: Y (Y or N)
Dump On Abend: Y (Y or N)
Priority: 0 (0-255)
Task Class: N (1-10 or N)
Resource Token: 
Use Alternate Screen Size: N (Y or N)

System ID: 
Remote Transaction Code: 

Description: SAMPLE MQM FRONT END PROGRAM

Enter F1=Help  F3=Exit  F10=Actions  F12=Cancel

18*C000  A

Figure 55. PCT Entry for MQ1 Transaction
<table>
<thead>
<tr>
<th>Update</th>
<th>Add</th>
<th>View</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit</td>
<td>Help</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FAAPCT2**  
Program Control Table-1

<table>
<thead>
<tr>
<th>Transaction Code</th>
<th>Group Name</th>
<th>Program Name</th>
<th>Secure</th>
<th>Can Be Purged</th>
<th>Dump On Abend</th>
<th>Priority</th>
<th>Task Class</th>
<th>Resource Token</th>
<th>Use Alternate Screen Size</th>
<th>System ID</th>
<th>Remote Transaction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>TECH</td>
<td>SAMPLE</td>
<td>TECHPROG</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>0</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CKTI</td>
<td>SAMPLE</td>
<td>AMQLTMC0</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>0</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description:** SAMPLE APPL FRONT END PROGRAM

**Enter F1=Help F3=Exit F10=Actions F12=Cancel**

**Figure 56. PCT Entry for TECH Transaction**

<table>
<thead>
<tr>
<th>Update</th>
<th>Add</th>
<th>View</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit</td>
<td>Help</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FAAPCT2**  
Program Control Table-1

<table>
<thead>
<tr>
<th>Transaction Code</th>
<th>Group Name</th>
<th>Program Name</th>
<th>Secure</th>
<th>Can Be Purged</th>
<th>Dump On Abend</th>
<th>Priority</th>
<th>Task Class</th>
<th>Resource Token</th>
<th>Use Alternate Screen Size</th>
<th>System ID</th>
<th>Remote Transaction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKTI</td>
<td>SAMPLE</td>
<td>AMQLTMC0</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>0</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description:** CICS MQ TRIGGER MONITOR TX

**Enter F1=Help F3=Exit F10=Actions F12=Cancel**

**Figure 57. PCT Entry for CKTI Transaction**
### E.2 Program Processing Table Entries

<table>
<thead>
<tr>
<th>Update</th>
<th>Add</th>
<th>View</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit</td>
<td>Help</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FAAPPT2 Processing Program Table-1

Program Name ............ : AMQLTMC0
Group Name .............. : SAMPLE
Program Type(P,M,D) ............. : P (P, M or D)
Resident(P,T,N) ............ : N (P, T or N)
System ID ............... :
Remote Program Name ........ :
Remote Transaction ID ........ :

Description ............. : CICS MQSERIES TRIGGER MONITOR

Enter F1=Help F3=Exit F10=Actions F12=Cancel

![Figure 58. PPT Entry for AMQLTMC0 Program](image1)

Program Name ............ : AMQSCIC1
Group Name .............. : SAMPLE
Program Type(P,M,D) ............. : P (P, M or D)
Resident(P,T,N) ............ : N (P, T or N)
System ID ............... :
Remote Program Name ........ :
Remote Transaction ID ........ :

Description ............. : MQM PROGRAM ON LOCAL

Enter F1=Help F3=Exit F10=Actions F12=Cancel

![Figure 59. PPT Entry for AMQSCIC1 Program](image2)

Appendix E. CICS Definitions 307
Figure 60. PPT Entry for TECHPROG Program

Figure 61. PPT Entry for VSAMSERV Program
Figure 62. FCT Entry for TECHBASE File

Figure 63. FCT Entry for TECHALT Alternate Index File
E.4 CICS System Definition Entries

```
***********************************************************************
* *
* IBM MQSeries for MVS/ESA
* CICS resource definitions for the CICS bridge
* *
***********************************************************************
*
DEFINE TRANSACTION(CKBP) GROUP(CSQCKB)
DESCRIPTION(MQSeries for MVS/ESA - CICS bridge - DPL bridge task)
  PROGRAM(CSQCBP00) TWASIZE(0) PROFILE(DFHCICST) STATUS(ENABLED)
  TASKDATALOC(ANY) TASKDATAKEY(USER) DYNAMIC(NO) PRIORITY(1)
  TCLASS(NO) DTIMOUT(NO) INDOUBT(BACKOUT) RESTART(NO) SPURGE(NO)
  TPURGE(NO) DUMP(YES) TRACE(YES) RESSEC(NO) CMDSEC(NO)
*
DEFINE PROGRAM(CSQCBP00) GROUP(CSQCKB)
 DESCRIPTION(MQSeries for MVS/ESA - CICS bridge - DPL program)
  LANGUAGE(C) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
  USELPACOPY(NO) STATUS(ENABLED) CEDF(NO) DATALLOCATION(ANY)
  EXECKEY(CICS) EXECUTIONSET(FULLAPI)
*
DEFINE PROGRAM(CSQCBP10) GROUP(CSQCKB)
 DESCRIPTION(MQSeries for MVS/ESA - CICS bridge - DPL abend handler)
  LANGUAGE(C) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
  USELPACOPY(NO) STATUS(ENABLED) CEDF(NO) DATALLOCATION(ANY)
  EXECKEY(CICS) EXECUTIONSET(FULLAPI)
*
DEFINE TRANSACTION(CKBR) GROUP(CSQCKB)
 DESCRIPTION(MQSeries for MVS/ESA - CICS bridge - Monitor task)
  PROGRAM(CSQCBR00) TWASIZE(0) PROFILE(DFHCICST) STATUS(ENABLED)
  TASKDATALOC(ANY) TASKDATAKEY(CICS) DYNAMIC(NO) PRIORITY(1)
  TCLASS(NO) DTIMOUT(NO) INDOUBT(BACKOUT) RESTART(NO) SPURGE(NO)
  TPURGE(NO) DUMP(YES) TRACE(YES) RESSEC(NO) CMDSEC(NO)
*
DEFINE PROGRAM(CSQCBR00) GROUP(CSQCKB)
 DESCRIPTION(MQSeries for MVS/ESA - CICS bridge - Monitor program)
  LANGUAGE(C) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
```

Figure 64 (Part 1 of 2). CICS/ESA Resource Definitions for the MQSeries - CICS DPL Bridge
USELPACOPY(NO)  STATUS(ENABLED)  CEDF(NO)  DATALOCATION(ANY)  
EXECKEY(CICS)  EXECUTIONSET(FULLAPI)  

*  
DEFINE PROGRAM(CSQCBTX)  GROUP(CSQCKB)  
DESCRIPTION(MQSeries for MVS/ESA - CICS bridge - messages)  
LANGUAGE(C)  RELOAD(NO)  RESIDENT(NO)  USAGE(NORMAL)  
USELPACOPY(NO)  STATUS(ENABLED)  CEDF(NO)  DATALOCATION(ANY)  
EXECKEY(CICS)  EXECUTIONSET(FULLAPI)  

Figure 64 (Part 2 of 2). CICS/ESA Resource Definitions for the MQSeries - CICS DPL Bridge
Appendix F. MQSeries Definitions

This appendix contains the MQSeries definitions required to use our sample application.

F.1 MQSeries Definitions on OS/2

Figure 65 to Figure 74 on page 315 contain the MQSeries definitions required on the OS/2 workstation to use our sample application.

```
DEFINE CHANNEL('SYSTEM.DEF.SVRCONN') REPLACE +
  DESCR('Server Connection Channel for Notes Server') +
  CHLTYPE('SVRCONN') +
  TRPTYPE('TCP') REPLACE DESCR('') +
  MCAUSER('nobody') +
  MAXMSGL(4194304)

Figure 65. Server Connection Channel Definition
```

```
DEFINE QLOCAL('SYSTEM.CICS.INITIATION.QUEUE') REPLACE +
  DESCR('MQSeries Default CICS initiation queue')

Figure 66. Initiation Queue Definition for CICS
```

```
DEFINE PROCESS('TECHPROC') REPLACE +
  DESCR('Starts the CICS-Transaction TECH') +
  * Application type (non-specific)
    APPLTYPE('CICS') +
  * Application identifier
    APPLICID('MQ1') +
  * User data
    USERDATA(' ') +
  * Environment data
    ENVRDATA(' ')

Figure 67. Process Definition to Start CICS TECH Transaction
```
DEFINE QLOCAL('EI.INPUTQUEUE') REPLACE +
   DESCR('This is the Local Queue for input to amqscic1.css') +
* Trigger control on
   TRIGGER +
* Trigger on all messages
   TRIGTYPE(EVERY) +
* Trigger on any priority
   TRIGMPRI(0) +
* Process name if trigger
   PROCESS(TECH) +
* Initiation queue name
   INITQ(SYSTEM.CICS.INITIATION.QUEUE)

Figure 68. Local Queue Definition for EI.INPUTQUEUE Input Queue

DEFINE QLOCAL('EI.OUTPUTQUEUE') REPLACE +
   DESCR('Local Queue for output from amqscic1.css') +
* Trigger control on
   NOTRIGGER +

Figure 69. Local Queue Definition for EI.OUTPUTQUEUE Output Queue

DEFINE QREMOTE('MQEI_DPL_BRIDGE') REPLACE +
   DESCR ('Remote Queue which is located on MVS') +
   RNAME ('MQEI_DPL_BRIDGE') +
   XMITQ('CSQ1') +
   RQMNAME('CSQ1')

Figure 70. Remote Output Queue Definitions for Queue Located on MVS
F.2 MQSeries Definitions on MVS

Figure 75 on page 316 to Figure 80 on page 317 contain the MQSeries definitions required on the MVS site to use our sample application.
DEFINE QREMOTE('EI.OUTPUTQUEUE') REPLACE +
  DSCR ('Remote Queue which is located on OS/2') +
  RNAME ('EI_OUTPUTQUEUE') +
  XMITQ('EITEST') +
  RQMNAME('EITEST')

Figure 75. Remote Reply Queue Definitions for Queue Located on OS/2

DEFINE QLOCAL ('EITEST') REPLACE +
  DSCR ('Transmission queue to QMGR EITEST on OS/2') +
  USAGE (XMITQ)

Figure 76. Transmission Queue Definition to QMGR EITEST on OS/2

DEFINE CHL ('MVS_KRYPTON') REPLACE +
  DSCR ('Sender channel on MVS to OS/2') +
  CHLTYPE(SDR) +
  TRPTYPE(TCP) +
  CONNAME(129.33.160.137) +
  XMITQ ('EITEST')

Figure 77. Sender Channel Definition on MVS to OS/2

DEFINE CHL ('KRYPTON_MVS') +
  CHLTYPE(RCVR) REPLACE +
  DSCR ('Receiver channel on MVS from OS/2') +
  TRPTYPE(TCP)

Figure 78. Receiver Channel Definition on MVS from QMGR EITEST on OS/2
DEFINE QLOCAL('MQEI_DPL_BRIDGE') REPLACE +
   DESCR('MQ-CICS DPL BRIDGE REQUEST QUEUE') +
   * FIFO Delivery
      MSGDLVSQ(FIFO) +
   * Sharing
      SHARE +
   * Persistent
      DEFPSIST(YES) +
   * Backout hardened
     HARDENBO

Figure 79. MQSeries DPL Bridge Request Queue Definition

DEFINE QLOCAL('IMSOTMA.QUEUE') REPLACE +
   DESCR('IMS OTMA Queue') +

Figure 80. IMS OTMA Queue Definition
Appendix G. Special Notices

This publication is intended to help information system architects, Lotus Notes developers, and enterprise developers who want to connect Lotus Notes to their existing enterprise applications. The information in this publication is not intended as the specification of any programming interfaces that are provided by MQSeries Enterprise Integrator for Lotus Notes. See the PUBLICATIONS section of the Lotus Programming Announcement for Domino.Connect which contains MQSeries Enterprise Integrator for Lotus Notes for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been...
reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>CICS</td>
<td>CICS OS/2</td>
</tr>
<tr>
<td>CICS/ESA</td>
<td>DB2</td>
<td>DProp</td>
</tr>
<tr>
<td>IBM</td>
<td>IMS</td>
<td>IMS/ESA</td>
</tr>
<tr>
<td>MQ</td>
<td>MQSeries</td>
<td>MVS</td>
</tr>
<tr>
<td>MVS/ESA</td>
<td>OS/2</td>
<td>PowerPC</td>
</tr>
<tr>
<td>System/390</td>
<td>VTAM</td>
<td></td>
</tr>
</tbody>
</table>

The following terms are trademarks of Lotus Development Corporation in the United States and/or other countries:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2-3</td>
<td>Approach</td>
<td>Domino</td>
</tr>
<tr>
<td>Freelance</td>
<td>InterNotes</td>
<td>Lotus</td>
</tr>
<tr>
<td>Lotus Notes</td>
<td>LotusScript</td>
<td>Notes</td>
</tr>
<tr>
<td>NotesPump</td>
<td>NotesSQL</td>
<td>SmartSuite</td>
</tr>
<tr>
<td>Word Pro</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following terms are trademarks of Apple Computer, Incorporated:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>AppleTalk</td>
<td>Macintosh</td>
</tr>
<tr>
<td>MacOS</td>
<td>MacTCP</td>
<td>System 7</td>
</tr>
</tbody>
</table>

The following terms are trademarks or registered trademarks of Microsoft Corporation.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>Visual Basic</td>
<td>Windows</td>
</tr>
<tr>
<td>Windows 95</td>
<td>Windows 95 logo</td>
<td>Windows NT</td>
</tr>
</tbody>
</table>

The following terms are trademarks of other companies:

AT&T is a trademark of American Telephone and Telegraph Company.

C-bus is a trademark of Corollary, Inc.

DEC, Digital, VMS, VAX are trademarks of Digital Equipment Corporation.

HP-UX is a trademark of Hewlett-Packard Company.

IPX, NetWare, Novell, InixWare are trademarks of Novell, Incorporated.
SAP, R/3 are trademarks of SAP.

Java, Solaris, Sun, SunOS are trademarks of Sun Microsystems, Inc.

Encina, Transarc are trademarks of Transarc Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Other trademarks are trademarks of their respective companies.
Appendix H. Related Publications

The publications listed in this appendix are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

H.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 327.

The other books of the Lotus Solution for the Enterprise Collection are:

- Volume 1 - *Lotus Notes: An Enterprise Application Platform*, SG24-4837
- Volume 2 - *Lotus Notes and Database 2 Common Server*, SG24-4918 (to be published in 1997)
- Volume 3 - *Using the IBM CICS Gateway for Lotus Notes*, SG24-4512

These publications are also relevant as further information sources:

- *CICS Clients Unmasked*, SG24-2534-01
- *Lotus Notes Release 4 in a Multiplatform Environment*, SG24-4649
- *Enterprise Integration with Domino.Connect*, SG24-2181

H.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

<table>
<thead>
<tr>
<th>CD-ROM Title</th>
<th>Subscription Number</th>
<th>Collection Kit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>System/390 Redbooks Collection</td>
<td>SBOF-7201</td>
<td>SK2T-2177</td>
</tr>
<tr>
<td>Networking and Systems Management Redbooks Collection</td>
<td>SBOF-7370</td>
<td>SK2T-6022</td>
</tr>
<tr>
<td>Transaction Processing and Data Management Redbook</td>
<td>SBOF-7240</td>
<td>SK2T-8038</td>
</tr>
<tr>
<td>AS/400 Redbooks Collection</td>
<td>SBOF-7270</td>
<td>SK2T-2849</td>
</tr>
<tr>
<td>RS/6000 Redbooks Collection (HTML, BkMgr)</td>
<td>SBOF-7230</td>
<td>SK2T-8040</td>
</tr>
<tr>
<td>RS/6000 Redbooks Collection (PostScript)</td>
<td>SBOF-7205</td>
<td>SK2T-8037</td>
</tr>
<tr>
<td>Application Development Redbooks Collection</td>
<td>SBOF-7290</td>
<td>SK2T-8042</td>
</tr>
<tr>
<td>Personal Systems Redbooks Collection</td>
<td>SBOF-7250</td>
<td>SK2T-8042</td>
</tr>
</tbody>
</table>
H.3 Other Publications

These publications are also relevant as further information sources:

- **MQSeries Publications**
  - *MQSeries Introduction to Messaging and Queuing*, GC33-0805-01
  - *MQSeries MQI Technical Reference*, SC33-0850-02
  - *MQSeries Planning Guide*, GC33-1349-04
  - *MQSeries Application Programming Reference*, SC33-1673-02
  - *MQSeries Application Programming Reference Summary*, SX33-6095-01
  - *MQSeries Application Programming Guide*, SC33-0807-06
  - *MQSeries Distributed Queuing Guide*, SC33-1139-07
  - *MQSeries Programmable System Management*, SC33-1482-04
  - *MQSeries Command Reference*, SC33-1369-07
  - *MQSeries Clients*, GC33-1632-03
  - *MQSeries for OS/2 Version 2.0.1 System Management Guide*, SC33-1371
  - *MQSeries for MVS/ESA System Management Guide*, SC33-0806
  - *MQSeries for MVS/ESA Version 1.2.0 Messages and Codes*, GC33-0819

  For the latest information about MQSeries, visit the MQSeries World Wide Web site at: [http://www.software.ibm.com/mqseries](http://www.software.ibm.com/mqseries)

- **CICS Publications**
  - *CICS/ESA V3.3 Resource Definition*, SC33-0666
  - *CICS Clients Version 2: Administration*, SC33-1792
  - *CICS Family: Client/Server Programming*, SC33-1435
  - *CICS Family: Object Oriented Programming for CICS Clients*, SC33-1639
  - *CICS Clients Version 2: Gateways - Version 2.0*, SC33-1821
For the latest information about CICS, visit the CICS World Wide Web site at: http://www.software.ibm.com/ts/cics

- **IMS Publications**
  - *IMS/ESA V5 Administration Guide: Transaction Manager*, SC26-8014
  - *IMS/ESA V5 Application Programming: Transaction Manager*, SC26-8017
  - *IMS/ESA V5 Installation Volume 1: Installation and Verification*, SC26-8023
  - *IMS/ESA V5 Installation Volume 2: System Definition and Tailoring*, SC26-8024
  - *IMS/ESA V5 Operator’s Reference*, SC26-8030
  - *IMS/ESA V5 Open Transaction Manager Access Guide*, SC26-8026

For the latest information about IMS, visit the CICS World Wide Web site at: http://www.software.ibm.com/data/ims

- **Lotus Publications**
  - *LotusScript Language Reference*, Part No. 12754
  - *Lotus Notes Administrator’s Guide*, Part No. 12755
  - *Lotus Notes Application Developer’s Guide*, Part No. 12750
  - *Lotus Notes Database Manager's Guide*, Part No. 12751
  - *Working with Lotus Notes and the Internet*, Part No. 12763

For the latest information about Lotus Notes, visit the Lotus World Wide Web site at: http://www.lotus.com
How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**
  - To get LIST3820s of redbooks, type one of the following commands:
    1. TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
    2. TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  - To get BookManager BOOKs of redbooks, type the following command:
    TOOLS CAT REDBOOKS
  - To get lists of redbooks, type one of the following commands:
    1. TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
    2. TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  - To register for information on workshops, residencies, and redbooks, type the following command:
    TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  - For a list of product area specialists in the ITSO: type the following command:
    TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE

- **Redbooks Web Site on the World Wide Web**
  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**
  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**

- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

- **Internet Listserv**
  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.
How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

<table>
<thead>
<tr>
<th>In United States:</th>
<th>IBMMAIL</th>
<th>Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>usib6fpl at ibmmail</td>
<td><a href="mailto:usib6fpl@ibmmail.com">usib6fpl@ibmmail.com</a></td>
<td></td>
</tr>
<tr>
<td>In Canada:</td>
<td>cbmbkz at ibmmail</td>
<td><a href="mailto:lmanix@vnet.ibm.com">lmanix@vnet.ibm.com</a></td>
</tr>
<tr>
<td>Outside North America:</td>
<td>dkibmbsh at ibmmail</td>
<td><a href="mailto:bookshop@dk.ibm.com">bookshop@dk.ibm.com</a></td>
</tr>
</tbody>
</table>

- **Telephone orders**

  - United States (toll free) 1-800-879-2755
  - Canada (toll free) 1-800-IBM-4YOU
  - Outside North America (long distance charges apply) (+45) 4810-1320 - Danish (+45) 4810-1020 - German
  - (+45) 4810-1420 - Dutch (+45) 4810-1620 - Italian
  - (+45) 4810-1540 - English (+45) 4810-1270 - Norwegian
  - (+45) 4810-1670 - Finnish (+45) 4810-1120 - Spanish
  - (+45) 4810-1220 - French (+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

  - IBM Publications IBM Publications IBM Direct Services
  - Publications Customer Support 144th Avenue, S.W. Sortemosevej 21
  - P.O. Box 29570 Calgary, Alberta T2P 3N5 DK-3450 Allerød
  - Raleigh, NC 27626-0570 Canada Denmark
  - USA

- **Fax** — send orders to:

  - United States (toll free) 1-800-445-9269
  - Canada 1-403-267-4455
  - Outside North America (+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States) or (+1)001-408-256-5422 (Outside USA)** — ask for:

  - Index # 4421 Abstracts of new redbooks
  - Index # 4422 IBM redbooks
  - Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

  - Redbooks Web Site http://www.redbooks.ibm.com

- **Internet Listserv**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserv. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).
IBM Redbook Order Form

Please send me the following:

<table>
<thead>
<tr>
<th>Title</th>
<th>Order Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

First name           Last name

Company

Address

City    Postal code    Country

Telephone number    Telefax number    VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date    Card issued to    Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.
**Glossary**

**Special Characters**

@Mailsend(). A predefined Lotus Notes function that sends the information supplied in the arguments list to the listed recipients.

A

**API.** Application programming interface. A set of calling conventions defining how a service is invoked through a software package.

**agent.** A Lotus Notes routine that automates tasks. Agents perform routine Lotus Notes tasks in the background. You can program agents, using LotusScript or the Lotus Notes formula language.

**application unit of work.** A set of actions within an application that the designer chooses to regard as an entity. It is up to the designer to decide how, if at all, an application should be subdivided into application units of work, and whether any application unit of work will consist of one, or many, logical units of work (LUWs). Typically, but not exclusively, an application unit of work corresponds to a CICS transaction.

**asynchronous.** Without regular time relationship; unexpected or unpredictable with respect to the execution of a program instruction. See synchronous.

B

**business process.** An entity-handling activity that is of limited duration, defined in scope, and set by business goals and policies, not by organization or implementation.

C

**certification.** In Lotus Notes, certification is the process of having an authorized person (the certifier) authenticate the identity of a user or server.

**CICS.** Customer Information Control System (CICS). A distributed online transaction processing system—an online system controller and some utilities that are capable of supporting a network of many terminals. The CICS family of products provides a range of application platforms on many operating system platforms.

**CLI.** call level interface. A callable API for database access, which is an alternative to embedded SQL API. In contrast to embedded SQL, CLI does not require precompiling or binding by the user but instead provides a standard set of functions to process SQL statements and related services at runtime.

**client.** As in client-server computing, the application that makes requests to the server and, often, deals with the interaction necessary with the user.

**client/server computing.** A form of distributed processing, in which the task required to be processed is accomplished by a client portion that requests services and a server portion that fulfills those requests. The client and server remain transparent to each other in terms of location and platform. See client, distributed processing, and server.

**commit.** An operation that applies all changes made during the current unit of work.


**D**

database. (1) A collection of interrelated data stored together with controlled redundancy according to a scheme to serve one or more applications. (2) All data files stored in the system. (3) A set of data stored together and managed by a database management system. (4) In Lotus Notes, a group of documents and their forms and views, stored under one file.

distributed processing. Distributed processing is an application or systems model in which function and data can be distributed across multiple computing resources connected on a local area network or wide area network. See client/server computing.

document. In Lotus Notes, a document is an object containing text, graphics, video, or audio objects or any kind of rich text data.

**E**

ECI. External call interface. An application programming interface (API) that enables a non-CICS client application to call a CICS program as a subroutine. The client application communicates with the server CICS program, using a data area called a COMMAREA.

electronic messaging. The creation, transfer, storage, and retrieval of text, graphics, image, or voice data by electronic means.

environment. The collective hardware and software configuration of a system.

**F**

formula. A Lotus Notes programming language that contains a set of built-in macros, functions, and commands.

**G**

GUI. Graphical user interface. A style of user interface that replaces the character-based screen with an all-points-addressable, high-resolution graphics screen. Windows display multiple applications at the same time and allow user input by means of a keyboard or a pointing device such as a mouse, pen, or trackball.

**H**

host. (1) In a computer network, a computer providing services such as computation, database access, and network control functions. (2) The primary or controlling computer in a multiple computer installation.

**I**

IMS. Information Management System (IMS).


intranet. A TCP/IP network that is entirely under the control of a private authority or company. The intranet may or may not have connections to other independent intranets (which would then be referred to as extranets) or the Internet. It may or may not be fully or partially visible to the outside.

Internet. The Internet consists of large international, national, and regional backbone networks that allow local and campus networks and individuals to access to global resources.
Lotus Notes. A groupware application that helps organizations communicate, collaborate, and coordinate business processes.

LotusScript. The Lotus cross-product BASIC scripting language.

LUW. Logical unit of work. An update that durably transforms a resource from one consistent state to another consistent state. A sequence of processing actions (for example, database changes) that must be completed before any of the individual actions can be regarded as committed. When changes are committed (by successful completion of the LUW and recording of the sync point on the system log), they do not have to be backed out after a subsequent error within the task or region. The end of an LUW is marked in a transaction by a sync point that is issued by either the user program or the CICS server, at the end of task. If there are no user sync points, the entire task is an LUW.

message. In MQSeries, a string of bytes that one program wants to send to another.

messaging. A communications model whereby the distributed applications communicate by sending messages to each other. A message is typically a short packet of information that does not necessarily require a reply. Messaging implements asynchronous communications.

middleware. A set of services that allows distributed applications to interoperate on a local area network or wide area network. It shields the developer or end user from the system complexity and enables delivery of service requests or responses transparently across computing resources.

MQSeries. The MQSeries family of products provides APIs that enables you to code indirect program-to-program communication by using messages queues.

NotesPump. A data distribution server that offers high-volume data exchange and synchronization with DB2, Oracle, Lotus Notes, ASCII/text, Sybase, and ODBC databases.

object. A program or a group of data that can behave like a thing in the real world.

ODBC. Open database connectivity. A Microsoft-developed C database API that allows access to DBMSs through callable SQL, and does not require the use of an SQL preprocessor.

OLTP. Online transaction processing. A style of computing that supports interactive applications in which requests submitted by terminal users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. An OLTP system supervises the sharing of resources for processing multiple transactions at the same time, minimizes compute time and duration of locks, and separates user think time from the use of storage and other resources.

OTMA. A component of IMS providing an access path and an interface specification for sending and receiving transactions and data from IMS. This transaction-based connectionless client/server protocol uses the MVS Cross System Coupling Facility as the communication vehicle.

queue. An MQSeries object. A queue is an area of storage set aside by the queue manager to hold messages on their way from one program to another.
recovery. The use of archived copies to reconstruct files, databases, or complete disk images after they are lost or destroyed.

recoverable resources. Items such as individual files and queues whose integrity CICS maintains in the event of a system error.

RPC. Remote procedure call. A communication model where requests are made by function calls to distributed procedures elsewhere. The location of the procedures is transparent to the calling application.

replication. A Lotus Notes procedure that updates and distributes copies (replicas) of the same Lotus Notes database that are stored on different servers.

rich text. A Lotus Notes field capable of storing a variety of type styles, graphics, and multimedia.

server. Any computing resource dedicated to responding to client requests. Servers can be linked to clients through local area networks or wide area networks to perform services, such as printing, database access, fax, and image processing, on behalf of multiple clients at the same time.

SQL. Structured query language. A standard set of statements used to manage information stored in a database. By using these statements, users can add, delete, or update information in a table, request information through a query, and display the result in a report.

stored procedures. A facility to execute procedures that are stored at the server. Stored procedures allow an application program to be run in two parts. One part runs on the client, and the other on the server. This allows one call to a remote database to execute a procedure that may represent several repetitive accesses to the database. The server procedure at the database runs within the same transaction as the client application.

synchronous. (1) Pertaining to two or more processes that depend on the occurrence of a specific event such as a common timing signal. (2) Occurring with a regular or predictable time relationship.

TCP/IP. Transmission Control Protocol/Internet Protocol. A set of communication protocols that support peer-to-peer connectivity function for both local and wide area networks.

transaction. A unit of processing (consisting of one or more application programs) initiated by a single request. A transaction can require the initiation of one or more tasks for its execution.

transaction manager. Provides the function to begin, end, commit, and roll back transactions.

transaction monitor. Provides a total environment for transactional applications. In addition to transaction manager functions, provides services to aid development, execution, and operation of transaction applications.

transaction processing. A style of computing that supports interactive applications in which requests submitted by users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. A transaction processing system supervises the sharing of resources for processing multiple transactions at the same time.

trigger. Defines an action to be executed when a given condition occurs.

two-phase commit. For a database, a protocol that is used to ensure uniform transaction commit or abort in a distributed data environment between two or more participants. The protocol consists of two phases: the first to
reach a common decision, and the second to implement the decision.

**U**

**unit of work.** A recoverable sequence of operations performed by an application between two points of consistency.

**W**

**Web.** See World Wide Web

**Web browser.** The client component of the World Wide Web. The Web browser is responsible for formatting and displaying information, interacting with the user, and invoking external viewers for data types that it does not support directly. Examples of Web browsers are Mosaic and Netscape.

**Web server.** The server component of the World Wide Web. It is responsible for servicing requests for information from Web browsers. The information can be a file retrieved from the server’s local disk or generated by a program called by the server to perform a specific application function.

**World Wide Web.** (WWW) (W3) (the Web) An Internet client/server distributed information and retrieval system based on HTTP that transfers hypertext documents across a varied array of computer systems.

**workstation.** A configuration of input and output equipment at which an operator works. A terminal or microcomputer, usually one that is connected to a mainframe or a network, at which a user can execute applications.
List of Abbreviations

**ACL**  access control list
**AIX**  Advanced Interactive Executive (IBM's flavor of UNIX)
**API**  application program interface
**ASCII**  American National Standard Code for Information Interchange
**BAPI**  business application program interface (SAP R/3)
**BASIC**  beginner's all-purpose symbolic instruction code
**BMS**  basic mapping support (CICS)
**CICS**  Customer Information Control System
**CLI**  call level interface
**COMMAREA**  communication area (CICS)
**CSD**  CICS System Definition file
**DBMS**  database management system
**DPL**  distributed program link (CICS)
**EBCDIC**  extended binary coded decimal interchange code
**ECI**  external call interface (CICS)
**EPI**  external presentation interface (CICS)
**ESA**  Enterprise Systems Architecture
**FCT**  file control table (CICS)
**GU**  get unique call (IMS)
**GUI**  graphical user interface
**HTML**  Hypertext Markup Language
**HTTP**  Hypertext Transfer Protocol
**IBM**  International Business Machines Corporation
**IMS**  Information Management System
**IOPCB**  input/output program communication block (IMS)
**IS**  information system
**ISRT**  insert call (IMS)
**ITSO**  International Technical Support Organization
**LAN**  local area network
**LSX**  LotusScript Extension
**LS:DO**  LotusScript:Data Object
**MPR**  message processing region (IMS)
**MQ**  message queue
**MQCIH**  MQSeries-CICS/ESA DPL bridge header (MQSeries)
**MQEI**  MQSeries Enterprise Integrator
**MQI**  message queue interface
**MQLSX**  MQSeries link LotusScript Extension
**MQMD**  message descriptor (MQSeries)
**MVS**  Multiple Virtual Storage
**NT**  Microsoft Windows NT (new technology)
**OCI**  Oracle call interface
**ODBC**  open database connectivity
**OLTP**  online transaction processing
**OS/2**  Operating System/2
**OTMA**  open transaction manager access
**PCT**  program control table (CICS)
**PPT**  program processing table (CICS)
**RACF**  Resource Access Control Facility
**RPC**  remote procedure call
**SMTP**  simple mail transfer protocol
**SQL**  structured query language
**SSL**  Secure Sockets Layer
**TCP/IP**  Transmission Control Protocol/Internet Protocol
**TCTTE**  terminal control table terminal entry (CICS)
<table>
<thead>
<tr>
<th><strong>URL</strong></th>
<th>uniform resource locator</th>
<th><strong>XCF</strong></th>
<th>cross system coupling facility (MVS)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VTAM</strong></td>
<td>Virtual Telecommunications Access Method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Index

Special Characters
/c 91
/p 53, 91
/u 53, 91
@Command 72, 82
@Created 9
@Functions 9
@MailSend 60
$QueryOpenAgent 88
$QuerySaveAgent 82, 88
$Return 88

Numerics
3270 support
   See CICS 3270 direct service

A
access control list
   See ACL
accessibility 3
AccountingToken 114
ACL 10, 53, 55
action in Lotus Notes 9, 46
add-in task 9, 60
administration
administration, security 10
agent
   authority 55
   event driven 46
   for the Lotus Notes client 70, 71
   for the Web client 50, 70, 80
   in Lotus Notes 8
   in the ITSO MQEI sample application 70
   scheduled 46, 55, 71, 78
   security 51
   starting 73, 82
AMQLTMCO 109
AMQSCI1 33, 104, 108, 109
AMQSCOMA.TST 108
anonymous access 52, 54, 55
application
   development 2
   model 43
   trigger-monitor 19
ApplIdentityData 114
asynchronous communication 16, 60, 106
atomicity 24
audit 2
authentication 28, 91, 98, 114
AUTHENTICATOR 99
Authenticator property 39, 53, 55, 91, 105, 114, 121
authority, agent 55

B
backup 2, 58
BAPI 12
BASIC 9, 12
basic mapping support
   See BMS
batch processing in Lotus Notes 46
blocking call 77, 86
BMS 25, 32, 42, 97

C
call level interface
   See CLI
CEDA 94, 101, 109, 116
certificate 52
CESN 53, 98, 100
channel
   definition 19
   used by CICS DPL via MQSeries service 115
   used by IMS via MQSeries service 123
   used by Native MQSeries service 109
CICS
   3270 applications and MQEI 92, 99
   and BMS 25
   and CICS 3270 direct service 99, 101
   and CICS DPL direct service 92, 93
   and CICS DPL via MQSeries service 116
CICS (continued)
and Native MQSeries service 109
authentication and CICS 3270 direct service 98
authentication in CICS DPL direct service 91
authentication in CICS DPL via MQSeries service 114
client 26, 53
controlling units of work with MQEI 59
converting BMS maps 42
description 24
DPL 25
MQEI installation requirements 126
MQSeries-CICS/ESA DPL bridge 27
system definition file 116
transaction 24
CICS 3270 direct service
and security 98
application access 15
CICS definition 101
clear screen 100
implementation 94
MQEI Definition database 96
MQEI Security database 98
CICS DPL direct service
and security 91
application access 15
CICS definitions 93
implementation 88
MQEI Definition database 89
MQEI Security database 91
CICS DPL via MQSeries service
and security 114
application access 14
CICS definitions 116
implementation 110
MQEI Definition database 113
MQEI Security database 114
MQSeries definitions 114
CICS gateway for Lotus Notes 60
CICS link extra for Lotus Notes 60
CICSCLI command 53, 91
CICSCLI.INI 90, 96
CKT1 104, 108, 110
clear screen 100
CLI 12
client access implementation 71
client/server
and CICS 27
and Lotus Notes 7
application 3
computing model 7
in today's business environment 1
ITSO MQEI sample application 65
mobile 3
command in IMS 30
COMMAREA 25, 28, 89, 92, 104
commit 24
CONFIG.SYS 109
ConnectionManager 89, 96, 113, 119
consistency 24
cross system coupling facility
See XCF
cross-platform support 3
CSD 116
CSQ1 115, 118, 122
CSQ4CKBM 115
CSQCBP00 112, 116
CSQCBP10 116
CSQCBR00 116
CSQCBTXE 116

D
data
access and copy 63
direct access 2
integrity 1, 2, 24, 29, 59
staged 1
database
See also MQEI Definition database
See also MQEI Security database
converting to the Web 10
discussion 10
in Lotus Notes 7
in today's business environment 1
mail-in 60
ODBC 12
security 53, 54
database management system
See DBMS
DB2 LSX 12, 59, 64
DBMS
and CICS 94, 102, 110, 117
and Domino.Connect 11
and NotesPump 64
DB2 29
IMS 29
delayed reply 50
DFHCSDUP 116
distributed program link
See DPL
distributed queuing 22
document
creating 72, 82
in Lotus Notes 7
in the ITSO MQEI sample application 70
Lotus Notes database 7
message identifier 107
redisplaying on the Web 86
security 54, 55
status 78
unique ID 87
unprocessed 57
updating 77
user interface 78
Domino
and Lotus Notes 9
and security 52
and the Web 9, 81
converting databases to the Web 10
coordination with enterprise applications 59
for Web user 50
standards support 10
translation of Lotus Notes forms 69
Domino.Connect 11
DPL 14, 25, 90, 93
durability 24
dynamic translation 9

E
ECI 15, 25, 26, 53, 60, 89, 91, 92, 93
EI.INPUTQUEUE 103, 104, 108
EI.OUTPUTQUEUE 104, 109, 115, 118, 119, 122
EIMessage 13, 40, 74, 83
EIMessageErrorHandler 74, 83
EIRC_SECURITY_FAILURE 92, 121
EIReceiveOptions 40, 74, 83, 107
EIReceiveOptionsErrorHandler 74, 83
EI.SendOptions 40, 60, 74, 83, 107
EI.SendOptionsErrorHandler 74, 83
EI.Service 13, 39, 53, 74, 83, 91
EI.ServiceErrorHandler 74, 83
EI.Session 38
EIUI_NOT_APPLICABLE 39
electronic messaging 7
enterprise application
and replication 56
and security 54
connecting to 75, 84
coordination with Lotus Notes applications 59
handling security with MQEI 54
receiving reply from 76, 85
retrieving user ID and password 75
sending request to 76, 85
supported by MQEI 39
environment variables 109
EPI 15, 26, 53, 95
ERP 11
error handler 74, 83
external call interface
See ECI
external presentation interface
See EPI

F
FCT 102
FieldGetText 76
file control table
See FCT
folder 8
form 8, 53, 68, 69
formula 9, 46, 72

G
GetItemValue 75, 85
groupware 7
GU call 30, 31, 118, 120
handling errors 74, 83
hotspot 8, 82
HTML 9, 50, 69, 82, 87
HTTP 10, 81, 86, 87
Hypertext Markup Language
  See HTML
Hypertext Transfer Protocol
  See HTTP
IMS
  and IMS via MQSeries service 117, 123
  and security 15, 121
  authentication failure 121
  command 30
  controlling the unit of work with MQEI 59
  description 29
  message 29
  MQSeries-IMS bridge 30
  transaction 30
IMS via MQSeries service
  and security 121
  application access 14
  authentication failure 121
  implementation 117
  IMS definition 123
  MQEI Definition database 119
  MQEI Security database 121
  MQSeries definitions 122
IMSOTMA.QUEUE 118, 119, 122
InboundConnection 104, 113, 119
initialization file 65
initiation queue
  See queue
input queue
  See queue
input/output program communication block
  See IOPCB
INSTALL/IVP 33, 123
Internet 1, 3, 11, 50, 53
intranet 3, 11
IOPCB 120
isolation 24
ISRT call 30, 118, 120
ITSO MQEI sample application
  components 32
  description 65
  downloading 127
  sources 33
IVTNO 117
Java 10
KRYPTON_MVS 112, 116, 118, 123
log file 2, 58
logging 2
lookup 63
Lotus Notes
  action 9, 46
  and client/server 7
  and Domino.Connect 11
  and MQEI 5
  and MQLSX 61
  and security 54
  coordination with enterprise applications 59
  developing applications with 8
  groupware 7
  integration with OLTP 1
  name translation to enterprise user ID 54
  server security 54
  starting an agent 73
  transactional support 59
LotusScript 9
LotusScript extension
  See LSX
LS:DO
  accessing enterprise application 59
  component of Domino.Connect 12
  data access 63
LSX 9
mail-in database 60
message
definition 18
descriptor 28
IMS input and output 29
MQEI definition 74, 84
queue 18, 108
used by CICS 3270 direct service 97
used by CICS DPL direct service 90
used by CICS DPL via MQSeries service 114
used by IMS via MQSeries service 120
used by Native MQSeries service 105
middleware 2, 53, 54
mobile
and client/server 3
and transaction 3
Lotus Notes support 47
support 3
MPR 118
MQI 104, 108, 263
MQACT_NONE 114
MQEI
and Lotus Notes 5
and MQLSX 5, 62
and replication 56
and security 15, 52, 54
application access 14
CICS requirements 126
code level utility 42
converting BMS maps 42
creating the objects 74, 83
definitions for CICS 3270 direct service 96
definitions for CICS DPL direct service 89
definitions for CICS DPL via MQSeries service 112
definitions for IMS via MQSeries service 119
definitions for Native MQSeries service 104
description 5, 37
disk space 126
initialization file 65
LotusScript Extension 37
MQSeries requirements 125
other bridge solutions 60
system structure 12

MQEI (continued)
tracing 42
user type 46
using a three tier model 45
using a two tier model 43
utilities 42
Web access implementation 80
MQEI Definition database
creating an MQEI object 39
defining enterprise services 68
definition 40
MQEI object characteristics 13
used by CICS 3270 direct service 96
used by CICS DPL direct service 89
used by CICS DPL via MQSeries service 113
used by IMS via MQSeries service 119
used by Native MQSeries service 104
MQEI Security database
access control list 56
creating an MQEI object 39
defining the users 66
definition 41
implementation 53
used by CICS 3270 direct service 98
used by CICS DPL direct service 91
used by CICS DPL via MQSeries service 114
used by IMS via MQSeries service 121
used by Native MQSeries service 105
MQEI_DPL_BRIDGE 111, 115
mqei.ini 66
MQGET 104, 118
MQI 17, 62, 108
MQIIH 62
MQLSX
and MQEI 5, 62
description 61
MQMD 28, 62, 114
MQPUT 103, 111, 112, 118
MQPUT1 118
MQSeries
and MQLSX 61
bridge 23
channel 19
client 21, 53
default and system objects creation 108
definitions for CICS DPL via MQSeries service 114
MQSeries (continued)
definitions for IMS via MQSeries
service 122
definitions for the Native MQSeries
service 108
description 15
distributed 21
message 18
MQEI installation requirements 125
MQSeries-CICS/ESA DPL bridge 27
MQSeries-IMS bridge 30
platform 16
process definition 19, 103
server-connection channel 109
trigger 19, 103
user ID and password 105
MQSeries link extra for Lotus Notes 60
MQSeries link for Lotus Notes 60
MQSeries Link LotusScript Extension
See MQLSX
MQSeries-CICS/ESA DPL bridge 15, 27, 34, 114
MQSeries-IMS bridge 14, 30, 118, 121
MQSERVER, environment variable 109
MVS_KRYPTON 112, 116, 118, 123

N
Native MQSeries service
and security 105
application access 14
CICS definitions 109
delayed receipt 106
implementation 102
message definition 105
MQEI Definition database 104
MQEI Security database 105
MQSeries definitions 108
navigator 8
NotesDocument 73, 87
NotesDocumentCollection 79
NotesPump 12, 64
NotesSession 79
NotesUIDocument 73

O
ODBC 12, 63
offline processing in Lotus Notes 46
OLTP
and Domino.Connect 11
IMS 29
integration with Lotus Notes 1
online connection 47
online transaction processing
See OLTP
open database connectivity
See ODBC
Oracle LSX 12, 59, 64
OS/2, environment variables 109
OTMA 30, 118
OutboundConnection 96, 104, 113, 119
output queue
See queue

P
parallel processing 16
passticket 28
PCT 102
point of consistency 24
PPT 94, 101
print command 86
process
used by Native MQSeries service 109
program control table
See PCT
program processing table
See PPT

Q
queue
initiation 103, 108
input 108, 115, 122
message 18
output 109, 115, 122
transmission 112, 115, 118, 123
used by CICS DPL via MQSeries service 115
used by Native MQSeries service 108
used in IMS via MQSeries service 122
queue manager
   used by IMS via MQSeries service 122

R
RACF 15, 53
reader access authority 55
ReceiveMessage 76, 85, 92, 121
recovery 2, 58
referential integrity 59
reliability 3
remote procedure call
   See RPC
replica 58
replication 56
reply, delayed 50
ReplyToQueue 28, 104, 112
restart 58
rollback 2
RPC 25

S
SAP R/3 11
Secure Sockets Layer
   See SSL
security
   administration 10
   agent 51
   and CICS 3270 direct service 98
   and CICS DPL direct service 91
   and CICS DPL via MQSeries service 114
   and IMS via MQSeries service 121
   and Internet 53
   and Lotus Notes 54
   and MQEI 15, 52, 54
   and Native MQSeries service 105
   and the Web 51
   considerations for Lotus Notes clients 56
document 55
SendMessage 76, 85
Service definition 39, 74, 84
ServiceStep 90, 96, 113, 119
SQL 63
SSL 10, 52, 54
standards support 3

synchronous connection 106
syncpoint 2
SYSTEM_AUTHENTICATOR 99, 106
SYSTEM.CICS.BRIDGE.QUEUE 115
SYSTEM.CICS.INITIATION.QUEUE 103, 108
SYSTEM.DEF.SVRCONN 103, 109, 111, 115, 118, 123
SystemName 39, 92, 105, 114, 121

T
TCTTE 26
TECH 95
TECHALT 94
TECHBASE 89, 94
TECHPROC 109
TECHPROC 32, 34, 95, 102
terminal emulation 1
three tier model 45
time-independent communication 16
transaction
   CICS 24
   IMS 30
transid 97
transmission queue
   See queue
transparency 3
toggle
   definition 19
   message 108
   monitor task 104, 108
   used in Native MQSeries service 103
triggered task 60
two tier model 43

U
unit of work 24, 59
UnitOfWork property 60
UniversalID 87
Userld property 39, 53, 55, 105, 114, 121
UserIdentifier 114
utilities in MQEI 42
V
view 8, 53
VSAMSERV 32, 89, 112

W
WaitInterval 77, 86
WaitType 77, 86
Web
  access implementation 80
  agent 50
  and Domino 9
  and user interface object 88
  converting Notes databases with Domino 10
  creating a document 87
  displaying text 88
  form, creating 82
  starting an agent 82, 87
  submit button 87
  user 50
Windows NT, environment variables 109

X
XCF 30
ITSO Redbook Evaluation

Lotus Solutions for the Enterprise, Volume 4 Lotus Notes and the MQSeries Enterprise Integrator
SG24-2217-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. Please complete this questionnaire and return it using one of the following methods:

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____
If no, please explain:

_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

What other redbooks would you like to see published?

_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

© Copyright IBM Corp. 1997