Implementing IBM Lotus Enterprise Integrator 6

Architecting and designing enterprise solutions using LEI 6

Installing and administering LEI 6

Getting the most out of the Advanced RealTime activities

Andre Guirard
John Bergland
Deb Landon
Ian Barrack
Kim Greene
Shu Sia Lukito
Rajat Prashar

ibm.com/redbooks
Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (December 2003)

This edition applies to IBM Lotus Enterprise Integrator for Domino (LEI) Release 6.0.1 (product number 5733-LEI).

© Copyright International Business Machines Corporation 2003. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Chapter 9. Getting Started with the LC LSX

9.1 History of the LC LSX .................................................. 192
9.1.1 Migrating LEI LSX code from Previous LEI Versions .......... 192
9.2 The LSX Classes ......................................................... 193
  9.2.1 LCSession Class .................................................. 193
  9.2.2 LCConnection Class .............................................. 193
  9.2.3 LCField Class ..................................................... 194
  9.2.4 LCFieldlist Class ................................................ 194
  9.2.5 LCStream Class ................................................... 195
  9.2.6 LCDatetime, LCNumeric, LCCurrency Classes ............... 196
9.3 Basic database operations ........................................... 196
  9.3.1 Connect to a database ........................................... 196
  9.3.2 Handle LC errors ................................................. 196
  9.3.3 Select and fetch all the records in a table ..................... 197
  9.3.4 Select records that meet certain criteria ....................... 198
  9.3.5 Insert a new record .............................................. 200
  9.3.6 Delete records that meet certain criteria ....................... 201
  9.3.7 Update a record using writeback ................................ 201
  9.3.8 Update a record using a selection key .......................... 202
  9.3.9 Control Commit and Rollback ................................... 203
  9.3.10 Call a stored procedure ....................................... 204
  9.3.11 Using database timestamps .................................... 204
  9.3.12 Execute SQL that you create in your program ................. 205
  9.3.13 Work with binary data ......................................... 206
9.4 Using the Notes Connector ........................................... 207
9.5 Connection pooling .......................................................... 208
9.6 Performance tips ............................................................ 210
  9.6.1 Performance code samples ........................................... 212
9.7 Tips and tricks ............................................................... 218
  9.7.1 Using the Trace metaconnector ................................. 218
  9.7.2 Map by Name .......................................................... 219
  9.7.3 Sorting data ............................................................ 220
  9.7.4 Updating key fields ................................................ 221
  9.7.5 Offload database access to server .............................. 222
9.8 LEI power programming .................................................. 223
  9.8.1 LEI Programming 1: Transfer data from profile documents . . . 223
  9.8.2 LC LSX Example 2: Update Virtual Fields in Notes views .... 226
  9.8.3 LC LSX Example 3: Fix unvirtualized documents ............... 228
9.9 Notes client coding considerations .................................... 229
9.10 Other APIs ................................................................. 229
  9.10.1 LS:DO ................................................................. 229
  9.10.2 JDBC ................................................................. 230
  9.10.3 @DbLookup and @DbColumn functions ......................... 230
9.11 Additional information about the LC LSX ............................ 230

Appendix A. Removing Lotus Enterprise Integrator (LEI) .............. 233
  Considerations prior to removing LEI ..................................... 234
    Access to data ............................................................ 234
    LEI clustered environment .............................................. 234
    Domino Enterprise Connection Services (DECS) .................... 234
  Process of removing LEI .................................................. 235

Appendix B. Troubleshooting .................................................. 243
  Using the LEI log (leilog.nsf) ......................................... 244
  Using the Domino log (log.nsf) ....................................... 246
  Enabling log conflicts in replication activities ....................... 247
  Logging SQL statements .................................................. 248
  Using the trace metaconnector ......................................... 249
  Using Notes Server Diagnostic (NSD) ................................. 249

Appendix C. Additional material ............................................ 251
  Locating the Web material ............................................... 251
  Using the Web material .................................................. 251
    How to use the Web material ......................................... 251
  Further useful downloads ............................................... 252

Glossary ................................................................................. 253

Related publications ............................................................. 255
  IBM Redbooks .................................................................. 255
  Other publications ......................................................... 255
  Online resources ............................................................ 256
  How to get IBM Redbooks ................................................. 256

Index ................................................................................. 257
Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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

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

<table>
<thead>
<tr>
<th>AS/400®</th>
<th>IBM®</th>
<th>OS/400®</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domino Designer®</td>
<td>ibm.com®</td>
<td>PartnerWorld®</td>
</tr>
<tr>
<td>Domino®</td>
<td>iNotes™</td>
<td>QuickPlace®</td>
</tr>
<tr>
<td>DB2 Connect™</td>
<td>iSeries™</td>
<td>Redbooks™</td>
</tr>
<tr>
<td>DB2 Universal Database™</td>
<td>Lotus Enterprise Integrator®</td>
<td>Redbooks(logo)™</td>
</tr>
<tr>
<td>DB2®</td>
<td>Lotus Notes®</td>
<td>Sametime®</td>
</tr>
<tr>
<td>@server™</td>
<td>Lotus®</td>
<td>zSeries®</td>
</tr>
<tr>
<td>Everyplace®</td>
<td>Notes®</td>
<td></td>
</tr>
</tbody>
</table>

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

This IBM® Redbook helps you to implement Lotus® Enterprise Integrator® (LEI) 6. It is targeted for system administrators who plan to implement or upgrade to LEI 6 in their organization. This book provides tips and techniques to help you successfully deploy and administer LEI 6.

We begin by providing a brief introduction to what's new with LEI 6 and cover some architectural scenarios of how LEI 6 infrastructure could be implemented in a customer's environment. We then provide detailed information on how to install and administer LEI 6, as well as how to migrate or upgrade from LEI 3.x to LEI 6. Tools, techniques and utilities for debugging activities are described in depth, for testing connectivity to relational databases and testing connection documents in the LEI Administrator. Working examples are provided to illustrate how to use the Advanced RealTime activities that are new in LEI 6. It also provides examples of some of the batch activities previously available in LEI 3.x such as a scripted activity and a replication activity. Finally, the book concludes with a discussion of the LC LSX, the LotusScript programming interface to the Lotus Connectors which lets you program any functionality you need that isn't available through the LEI forms.

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, Rochester and Cambridge Centers.

Andre Guirard is a member of the Enterprise Integration development team at IBM Lotus Software, which is the organization that develops the products this book is about. He is a contributor to The View, LDD Today, and SearchDomino, and an occasional speaker at Lotus conferences. His main areas of expertise are enterprise integration and Domino® development.

John Bergland is a project leader at the International Technical Support Organization, Cambridge Center. He manages projects that produce Redbooks about Lotus Software products. Before joining the ITSO in 2003, John worked as an Advisory IT Specialist with IBM Software Services for Lotus (ISSL), specializing in Notes® and Domino messaging and collaborative solutions.
Deb Landon is a Certified IT Specialist in the IBM International Technical Support Organization, Rochester Center. Her current area of expertise is focused on Domino for iSeries™ and related Lotus products, such as Sametime® and QuickPlace®. Debbie has been with IBM for 19 years working first with the S/36 and then the AS/400®, now IBM @server iSeries Server. Before joining the ITSO in November of 2000, she was a member of the PartnerWorld® for Developers, iSeries team, supporting business partners in the area of Domino for iSeries.

Ian Barrack is Chairman of Simban Holdings Pty Ltd, an Australian based professional services firm. A Certified Lotus Instructor, he has 14 years experience in providing consultancy services in both private and government arenas. From a defense background, Ian has worked with developing solutions utilizing IBM/Lotus environments since 1993 based on Intel®, Sun, and IBM @server zSeries® and iSeries platforms. His expertise is in supplying corporate solutions incorporating systems/data security, enterprise integration, systems design and troubleshooting. The center of Ian’s attention is on providing appropriate solutions to organizations of all sizes. Ian can be contacted at ibarrack@simban.com.

Kim Greene, President of Kim Greene Consulting, Inc., specializes in Domino for iSeries consulting, development, customized education, and performance. Kim has over six years of experience with Domino and 14 years of experience with the AS/400 and iSeries platform. Kim specializes in iSeries Domino performance analysis, system and application tuning, enterprise integration with back-end applications and data, and Domino application development. Other services offered include providing customized iSeries Domino education and training for customers and business partners. For more information on any of these services, see www.kimgreene.com. Kim is also a frequent writer for technology magazines and speaks at many conferences worldwide. Previously she was employed at IBM where she worked in the PartnerWorld for Developers (PWD) organization helping IBM business partners incorporate Domino into their existing applications. Prior to working in PWD she worked in several areas of OS/400® performance in the AS/400 development laboratory.

Shu Sia Lukito is an Advisory IT Specialist with IBM Software Services for Lotus (ISSL) with the Bethesda, Maryland office. Her current area of expertise is focused on Domino design and development and related Lotus technologies, such as enterprise integration, Sametime and QuickPlace.
Rajat Prashar is an IT Architect with IBM Software Service For Lotus (ISSL) working out of the Toronto Lab. As an IT Architect with ISSL his main role is the design and development of strategic e-business solutions for IBM customers. Rajat has over 10 years of experience in technology consulting and holds a degree in MIS from Concordia University. His areas of expertise include Web application development, enterprise integration, collaboration, knowledge management, and e-commerce.

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

Peter Janzen
Product Manager, Lotus Software

Sarah Boucher,
Manager, Lotus Enterprise Integration Team, Portsmouth, NH

Thanks for putting up with Andre Guirard, spending inordinate amounts of time creating examples, and getting the details right.

Members of the extended LEI team at Lotus:

Tully Derby, Scott Morris, Russ Owen, Pam Gilday, Gary Mulford, and Carol Eaton

Thanks for their willingness to spend time answering questions and reviewing drafts, which has made this a better book than it otherwise would have been.

Tim Bishop
Steve Kluck
Gary Mulford
Domino for iSeries development team, IBM Rochester

Mike Gilley
IBM AS/400 Support Center, Lotus Domino team

Brian Hallstrom
Deb McGuire
Lotus Support Center

Mike Pascoe
Enterprise Support Services, IBM Rochester

Jeff Huebert
DRDA/DDM development team, IBM Rochester

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.
Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

**Comments welcome**

Your comments are important to us!

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

- Use the online **Contact us** review redbook form found at:
  
  ibm.com/redbooks

- Send your comments in an Internet note to:
  
  redbook@us.ibm.com

- Mail your comments to:
  
  IBM Corporation, International Technical Support Organization
  
  1 Rodgers Street
  
  Cambridge, MA 02142
Chapter 1. Introduction to IBM Lotus Enterprise Integrator (LEI) 6 for Domino

This chapter provides an overview of the IBM Lotus Enterprise Integrator (LEI) 6 product. You will learn the following by reading this chapter:

- What is LEI?
- How LEI can be used to solve business problems
- What is Domino Enterprise Connection Services (DECS) and Domino Connection Resources (DCRs) and how do they compare to LEI 6?
- What functions the LEI 6 product provides
- What is new in LEI 6.0.1?
- What is new with LEI 6 on the iSeries?
- A discussion of the terminology differences between Domino and DB2® Universal Database™ (DB2 UDB)
- Data type mappings between Domino and DB2 UDB on the iSeries
1.1 A business case for LEI

Enterprise integration is a critical requirement for almost every business customer. Most organizations have legacy information and active systems based on several different database systems. Coordinating data between systems can be a major challenge, requiring a great deal of custom coding and in-depth knowledge of the database systems in question.

To get productivity gains and improve collaboration and customer service, it's also increasingly important to integrate your back-end systems, including ERP and CRM, into a user-friendly interface for your employees, partners and customers, without breaking the bank on development costs.

LEI is a separately acquired Lotus product for the Domino server. It provides the ability to perform high-volume data transfers, synchronize disparate data sources, and provide real-time integration with back-end data sources.

LEI provides easy-to-use tools to migrate or replicate data between different back-end relational databases, or between relational databases, enterprise systems, or Lotus Notes®. Advanced Realtime also lets you use the Domino Designer® to quickly create applications that directly access data in your back-end systems. The familiar interface that lets your users read their e-mail, share documents, and participate in discussions, also lets them access and update enterprise data in real time, transparently. This lets you roll out new applications with much less user training required, than if you developed a custom system.

Since a Lotus Notes application can be easily made into an Internet or intranet application by activating the HTTP function of the Domino server, LEI also lets you tie in your back-end systems to intranet and Internet users. The Domino Everyplace® Server and Domino Global Designer further extend your reach to mobile devices, and people in different parts of the world. And you can easily build Lotus Sametime awareness into your Notes/Domino applications, tying in your enterprise instant messaging capability with your enterprise data.

Many businesses use LEI to let their employees work in a disconnected mode while still having access to enterprise data. This supports sales personnel who work remotely and are rarely connected to a server. The sales person can replicate the Domino application and data to a laptop. They can have a local copy of just that part of the back-end data that concerns them, automatically updated whenever they connect in. They can also make changes to their local data (if they’re authorized) and have those changes immediately reflected in the back-end systems when they replicate. For instance, they could enter a new order locally, that replicates directly into the tables of a DB2 order tracking database.

Lotus’ enterprise integration strategy is shown in Figure 1-1. Lotus’ strategy provides full access to Domino from any client to any data. This is critical in today’s business environment. Many businesses have data in various types of data sources, including relational databases, transactional systems, ERP systems, and many others. These back-end resources need to be accessed from diverse client types, including Web browsers, Lotus Notes clients, wireless devices such as phones or pagers, the iNotes™ client, and so on.

1 Depending how fancy you want to get about using advanced browser features.
Most application architectures today have three tiers:

- Presentation
- Application logic
- Data

Application development teams have different skills to create and maintain each tier of the application. The presentation layer requires HTML and JSP skills, application logic requires LotusScript, Java™, servlets, C, and other skills, while data access requires database administration skills, SQL, data normalization, and possibly stored procedure knowledge.

In the Lotus architecture, the Lotus Connectors provide a uniform way to access the data. There's a separate connector for each type of database, plus an ODBC connector for access to any databases where we don’t support a “native” connector. Others, such as the SAP R/3 connector\(^2\), provide access to application-specific data in a relational-like way. There's a base set of functionality common to all of these connectors. There are also differences between connectors, because as much as possible, the connectors give you access to the unique features of the back-end database. However, there's enough commonality that Lotus can write other parts of the system — the timestamp-based replication in LEI, for instance — without knowing in advance what types of databases you will be moving data between.

LEI eases the strain placed on businesses by making it easier to integrate with various back-end systems. Because it's forms-based, LEI lightens the burden on application developers by not requiring the developer to have specific database administration skills or SQL skills. The LEI administrator can easily tie together tables, views, and stored procedures developed by others, without knowing the innermost details of the database system. Even when writing code with the LC LSX, the Domino programming interface to the connectors —

\(^2\) Sold separately.
in nearly all cases you can read and write data, even create tables and indexes, without writing any SQL. The programming interface lets you deal with the data as abstract objects, and the connector generates SQL (or whatever command language the back-end system uses) and deals with character set conversions automatically and efficiently.

The application logic can be coded as stored procedures and validation rules in the back-end database, or as part of your Notes forms, or in LotusScript using the LC LSX. Or a combination of these, depending on the needs of your application and where you have the most development expertise.

For security, likewise, you can leverage existing authentication and user access of a back-end database, or you can take advantage of the highly secure user authentication, granular read and write access controls and encryption capabilities that Domino provides — or combine these approaches.

While there are several other ways to access back-end data sources from Domino — notably, the LS:DO — LEI provides very powerful enterprise integration options with a minimum of custom coding, maximum performance, and the ability to abstract the task and let the connectors deal with the details.

Domino 6 extends the rich functionality of LEI by incorporating data real-time through a new technology called virtualization that makes integration with back-end data sources seamless. One of the biggest inhibitors to using Domino Enterprise Connection Services (DECS) R4.6/R5 or Domino Connection Resources (DCR) technology in Domino 6 is the problem of having to synchronize keys between DB2 UDB and Domino data sources for a Virtual Fields activity\(^3\). These keys quickly become out of sync as non-Domino applications, such as RPG and COBOL programs, insert or delete records from these same back-end DB2 UDB data sources. If the insertions and deletions are not happening exclusively through Domino, DECS and DCRs are not aware that new keys are available and that some keys have been eliminated in the back-end data store. There are workarounds to this issue, but they require programming skills to resolve.

LEI 6 eliminates most of these workaround requirements by providing real-time access to back-end data stores through its virtualization technology. This new technology offers the ability to integrate with back-end data stores without having to store any keys in the Domino environment. Virtualization is almost magical. It provides the functionality of having all the data stored outside of Domino, while still allowing an end user to work with that data as if it were stored locally in the Domino database. This means the virtualized data can participate in views, offering a very rich set of functionality not previously available with real-time data access mechanisms.

### 1.2 Which tools come with Domino?

In the previous section, both DECS, DCRs and the LC LSX were briefly discussed. Along with various Lotus Connectors, these components are part of the Domino server, included at no extra cost. Let's look at these integration options in further detail so you understand how they compare with the functionality provided by LEI; which applications require LEI, and which can you do without paying extra?

#### 1.2.1 DECS

DECS has been a component of the Domino server since Release 4.6.3. It offers developers a non-programmatic way to create applications that provide live, native access to enterprise

\(^3\) Virtual Fields were called “RealTime Notes” in earlier versions of LEI/DECS.
information. DECS provides real-time access to data in a back-end data source such as DB2 UDB. DECS can be used to visually map data between Domino and back-end data sources such as DB2 UDB, Notes, text, stream files, and ERP systems.

The relationship between a Domino application and an enterprise data source on the back-end is defined in a Domino database known as the DECS Administrator. The DECS administrator provides a visual wizard and online documentation to assist the application developer in defining the external data source connections to the back-end data source. Fields in this data source and the Domino application are visually identified and mapped in activity documents which are created in the DECS Administrator database. With these connection and activity documents, Domino applications can operate on enterprise or external data as seamlessly as if they were located in Domino. With DECS, external, back-end data can be viewed, created, updated, or deleted directly and transparently by the Lotus Notes or Web browser client.

The biggest inhibitor for businesses to utilize DECS in their Domino applications requiring enterprise integration is the issue described in the previous section. This is the issue of the keys becoming out of sync when applications external to Domino access the same relational databases that Domino is accessing.

DECS capabilities are a subset of the features available in LEI. Most importantly, LEI adds

- **Virtual Documents**: let the Notes user interface transparently present data stored in a relational back end, with no “stub” documents in Notes.
- **Virtual Agents**: let Notes users execute stored procedures though the Notes client interface or Web browser.
- **Virtual Attachments**: Store file attachments in a relational database instead of in Notes documents.
- **Integrated Credentials**: let you easily apply your relational database's access controls to Notes users.
- **Replication Activity**: keeps relational data in sync with data in other systems, including Notes. This is the missing piece to keep your “stub” documents up to date with additions and deletions in the source data for a Virtual Fields activity.

### 1.2.2 DCRs

DCRs (Data Connection Resources) were introduced in Domino 6. They bring the technology of DECS into the Domino Designer client. From Domino Designer, you can define a connection to an external data source, such as a relational database. You can then use the connection to link the fields that are contained in a form to fields that are found in the external source. DCRs are reusable in an application and can be shared across applications. You can use DCR technology to access data in enterprise systems and then take advantage of the power of a Domino application to replicate, share, secure, and manage the data.

DCRs have the same issue as DECS, where keys can become out of sync if there are applications external to Domino manipulating the back-end DB2 UDB databases.

For more information on DCRs, see the article “Lotus Domino Designer 7 Technical Overview” on the Lotus Developer Domain LDD Today Web site. This article can be found at: [http://www.lotus.com/ldd/today.nsf](http://www.lotus.com/ldd/today.nsf)

Additional information about both DECS and DCRs can be found in the Domino Designer Client Help database. Both DECS and DCR support comes with the base Domino 6 code. They are not separate products you need to purchase.
1.2.3 LC LSX

The LC LSX is the LotusScript programming interface to the Lotus Connectors. This is the tool you would use to program any aspects of the integration that DECS or LEI couldn’t support through their configuration screens.

The LSX is covered in detail in Chapter 9, “Getting Started with the LC LSX” on page 191.

It is also possible to access the Lotus Connectors by writing C or C++ code. While this offers the possibility of better performance, most high-volume data transfers can be set up using the LEI activity forms, and LotusScript code is easier to integrate with a Notes/Domino application than C code is.

1.3 LEI defined

The LEI product has been providing core integration technology between Domino and back-end data sources for a number of releases now. The product was first introduced as NotesPump. As popularity in this tool grew, the NotesPump product was rewritten and enhanced to become LEI.

LEI is a server based data distribution tool that provides customers with a forms based, no programming required method to move information between legacy and relational server sources and Domino applications. It is the perfect tool for solving business problems that require data stored in different types of data sources to tightly integrate. LEI has several functions that allow for this integration. These functions are described in 1.3.3, “Activity documents” on page 8.

LEI has an incredibly rich set of functionality for integrating Domino with numerous back-end data sources. LEI is a separate product that is purchased from Lotus.

In the next two sections we discuss the two main components for using LEI, connection and activity documents.

1.3.1 The LEI Administrator

All the administration functions of LEI, including the creation and management of connections and activities, are done through the LEI Administrator Notes database. In LEI 6, this database is named decsadm.nsf, and it is often, but not always, on the same Domino server as the Notes databases you want to connect to relational data. To access the LEI Administrator from the Lotus Notes client, select File Open, choose the server, and either type “decsadm.nsf” in the File field, or select “LEI Administrator” from the list.

1.3.2 Connection documents

Connection documents define the data sources that will be used by LEI activities. A connection can be used as either a source or a destination, or both. They define not only connections to relational databases, but also to Notes databases, text files, or file systems, and can include special application-based connectors such as SAP R/3.

The information provided in a connection document includes the type of data source being connected to (for example, “DB2”), user authentication information for connecting to that data source, the data source name, and connector specific information, such as whether the data source is being journaled or not in the case of DB2 UDB.
There are a core set of connectors that ship with the Domino server code. Connectors provide the strategic “plumbing” for enterprise integration. They deliver native connectivity, via a consistent object model, to external data sources. Connectors allow Domino applications to connect, authenticate, and translate data that resides within relational databases, enterprise resource planning, and transaction processing systems.

There are two sets of connectors, the base connectors and premium connectors. The base connectors are included in the Domino 6 server code and include the following:

- DB2 (7.x and 8.1)
- Oracle (7, 8i, and 9i)
- Sybase (11)
- OLE-DB (SQL Server 2002 and Access 2000)
- ODBC
- File
- Text
- Notes

Lotus Domino is a cross platform product, and not all of these connectors are supported on every platform. In particular, the current version of Domino for iSeries excludes all relational database connectors except for DB2.

Premium connectors provide access to ERP back-end data sources. Because these connectors are provided by the ERP provider themselves, they are a chargeable feature. It is very important to use an ERP connector rather than manipulating the back-end ERP files directly. ERP systems have many sets of business rules that need to be applied to the back-end data being manipulated.

At the time this redbook was written, only one ERP provider has a connector available to work with Domino 6. That ERP provider is SAP R/3 Version 1.7. Also during the writing of this redbook, the SAP R/3 connector did not support the new Advanced RealTime activities; only the batch LEI activities are supported with LEI 6.

For the latest information on what premium connectors are available to work with LEI, refer to the Lotus Enterprise Integration Web site at:

http://www.lotus.com/lei

For details on creating connection documents, see Chapter 6, “Connection documents” on page 73.

**Metaconnectors**

Metaconnectors are special LEI connection documents that provide additional data processing on data returned from a specified connector. There are four different metaconnection types in LEI 6.0.1:

- **Collapse/expand**
  
  Provides the capability to take multiple records from one data source table and collapse them into a single form field. This metaconnection document also performs the reverse operation to expand data into multiple records. This is useful in converting Notes multivalue fields into multiple rows in a relational table.

- **Meter**

  Provides a way to collect statistical usage data, to identify and quantify data access patterns.

- **Order**
Lets you sort data in a connector-independent way as it is read. This metaconnection is especially helpful to order data sets returned from Domino to match the order alphanumeric data is returned from a DB2 connection.

- Trace
  This metaconnector is used to debug problems with connection documents and activities; it produces debugging output, including the SQL statements used to manipulate the relational database.

For more details on metaconnection documents, see 6.6, “Metaconnectors” on page 97.

1.3.3 Activity documents

There are several types of Activity documents, based on the type of function LEI will perform. Based on this function, you select the appropriate source and target data sources and additional details about the activity, such as which fields in the source correspond to which fields in the target. Create an activity by filling out the appropriate form. You specify the source and target by selecting Connection documents that you created previously.

For the Advanced RealTime activities, there is a user assistant which prompts you for input via dialog boxes for completing the form. User assistance is enabled by default. If you would like to disable this feature, click the Turn User Assistant Off button in the LEI Administrator main screen. By disabling the user assistant wizard, you will be taken directly to the Advanced RealTime activity form rather than going through the dialog boxes. The batch activities in LEI do not utilize the user assistant. When creating one of these activity documents, you are taken directly to the form to fill in the appropriate parameters.

Following is a quick introduction to each of the different types of activities that can be created within LEI. The activities are grouped into two categories, batch and Advanced RealTime activities.

Batch LEI activities

Batch LEI activities have been available with the LEI product from its original version. These activities continue to be available in LEI 6 and include:

- Admin-backup
  Creates a backup copy of the LEI Administrator database (decsadm.nsf) and optionally, the LEI Script Vault database (leivlt6.nsf).

- Admin-purge log
  Purges the LEI log database (leilog.nsf) of documents that are older than a specified number of days.

- Archive
  Used to reduce the number of documents in a specified database. The archive activity moves data from one database to another. As the records are moved into the target database, they are deleted from the source database.

- Command
  This activity allows you to execute an operating system command, a database command, or SQL commands.

- Direct transfer
  The direct transfer activity copies data from one database to another. The target database can be created during execution of the direct transfer. For example you could create a Domino database from a DB2 UDB database with this activity.
Chapter 1. Introduction to IBM Lotus Enterprise Integrator (LEI) 6 for Domino

- **Java**
  Allows LEI to invoke a Java application. This Java application can call application logic outside of LEI, allowing you to perform more extensive manipulations on the data being transferred.

  **Note:** The LC Java classes have been withdrawn from the Domino 6 product and are not available for the Java activity in LEI 6.

- **Polling**
  Polling allows certain conditions to be monitored in a database. When the conditions are met, a subordinate activity is triggered.

- **Replication**
  Replication is a very powerful feature of LEI. It allows two different types of data sources to be synchronized. The replication activity is often used to synchronize a Domino database with a relational database, or two relational databases with each other.

  For an example of a replication activity, see 7.7, “Replication activity” on page 130.

- **Scripted**
  A scripted activity provides the function of executing LotusScript (LSX) commands. Using the LotusScript classes, you can extend the functionality available to other LEI activities such as direct transfer, polling, and replication. Scripted activities allow you to create customized routines that provide more control over source and destination data transfers.

  **Note:** The LEI LSX that was previously available with the LEI 3.x products is no longer supported in LEI 6. These functions have been merged into the LC LSX.

  For an example of a scripted activity, see 7.5, “Scripted activity” on page 125.

**Advanced RealTime activities**

Advanced RealTime activities are new to LEI 6. These new activities rely on the virtualization technology to provide real-time access to data external to Domino. The new Advanced RealTime activities available with LEI 6 include:

- **Virtual agents**
  Virtual agents are a new activity introduced in LEI 6 that enables stored procedures to be invoked and executed as if they were Domino agents. Virtual agents can be scheduled, triggered, or invoked manually. They can be run with or without parameters.

- **Virtual documents**
  Virtual documents are also new with LEI 6. They allow external records to appear to Domino as if they are actual documents stored in the Domino database. Virtual documents greatly expands the real-time capability that was provided in previous releases of LEI and DECS. This level of virtualization allows external data to participate in Domino views, greatly enhancing the functionality of Domino applications that are dependent on data stored in a back-end data source such as DB2 UDB.

- **Virtual fields**
  This activity, previously called “Real-time Notes”, actually is available in previous LEI versions and DECS, but has been renamed to correspond to the other virtual activities. Virtual fields connect Domino forms to back-end data sources, allowing you to open, create, update, and delete external system data directly through the Domino application. The biggest difference between virtual documents and virtual fields is that virtual field
activities require a key document to be stored in the Domino database. With virtual documents, no data is stored in the Domino database, it is 100% virtualized.

For details and examples of using Advanced RealTime activities, see Chapter 8, “Advanced RealTime activities” on page 151.

1.3.4 Current limitations with using LEI 6

LEI 6 has come a long way in providing truly robust enterprise integration with the addition of virtual documents and virtual agents. The goal of virtualization is to provide seamless data transfers between Domino and RDBs even when other applications external to Domino make changes to the external data source.

However, this ideal must give way somewhat to the realities of network and performance issues. To work efficiently with outside data, LEI must have a way to tell which records in the relational database were recently changed or deleted.

To do this, and to store the header information that Notes requires as part of a document, a virtual document requires either additional fields, or a separate reference table, stored in the DB2 UDB alongside the records to be virtualized. Any outside process that updates the data, must also update the timestamp in the EIMODIFIED field; if not, LEI will not know that the record has been updated and the change will not be reflected in Notes views until the view index is rebuilt.

Likewise, when a record is deleted by an external process, the LEI accounting information must be updated in a specific way, or else the records will remain in the views. Refer to 8.6.6, “Deletion involving virtual documents” on page 177 for details of document deletion.

Even if this is not done, updated records will be shown correctly when they are opened in a document window, and deleted records will give an error if users try to open them from a view.

If you require Notes views to be up to date and do not have the option of updating the timestamp, use Domino Designer to select the Notes view option to discard the index after every time it’s used; that guarantees that the view will always be up to date. However, depending on the size of your database, this may be slow.

1.4 A comparison of DECS, DCRs, and LEI

Figure 1-2 provides a summary of the functions that are available with DECS and LEI, depending on which release of Domino you are working with. As you can see, real-time access to back-end data was available in both DECS in Domino R4.6/R5.x and LEI 3.x. The LEI product, separately purchased, provides not only real-time access to back-end data, but also provides the rich functionality of data management activities such as replication, direct transfer, and the additional activities you see listed.

In Domino 6, DECS continues to provide real-time access to back-end data with the core Domino server code, and additionally this real-time access can be accessed directly via field properties in the Domino Designer client through the new DCR technology support.

The LEI 6 product continues to provide the batch data management activities that were provided with LEI 3.x. LEI has been enhanced significantly in Domino 6 to provide the new Advanced RealTime activities support for virtualizing fields, documents, and agents. These new activities also provide some Advanced RealTime features that include virtual attachments and integrated credentials.
In Figure 1-2 we see that both virtual attachments and integrated credentials are Advanced RealTime features. This means you cannot create an LEI activity of a type virtual attachment or of a type integrated credentials. These features are additional enhancements available with the new virtualization technology added into LEI 6. You can only utilize the features of virtual attachments and integrated credentials through the three new Advanced RealTime activities of virtual fields, virtual documents, or virtual agents.

### 1.4.1 Advanced RealTime feature - virtual attachments

The virtual attachment Advanced RealTime feature allows files and other objects to be stored in an external data source and appear as if they are Domino attachments. The actual attachment is stored in an external system but is indistinguishable from a native attachment to the end user and the developer. These attachments are stored as binary or BLOB data types in the RDB.

Users and applications can perform any operation with virtual attachments that Domino supports for attachments it stores locally. This includes opening, saving, and also the new Domino 6 capability to edit attachments in place. Virtual attachments allow Domino databases to remain at a much more stable size because the attachments are stored in an external data storage device. They also make it easier to keep track of and organize attachments.

For an a more complete discussion and example of using virtual attachments, see 8.2, “Virtual Attachments” on page 156, as well as 8.3, “Example 1: Virtual Fields activity” on page 158.

**Important:** Virtual attachments are not generally stored byte-for-byte the same as the source file. If the user creates the attachment with compression enabled (which is the default), the compressed data are stored in the BLOB. The original file cannot be recreated except by Notes/Domino.
1.4.2 Advanced RealTime feature - integrated credentials

This new Advanced RealTime feature allows Domino IDs to be associated with back-end userids (credentials). The LEI connection document contains a single userid and password. Using this ID for all activities can create problems.

- When the password for this userid expires, all activities that use the connection will cease working.
- It's difficult to track who really caused an update, delete, or insert to happen in a back-end database. Often, businesses need to be able to track this level of detail as part of their security auditing practice.
- All users have the same access to the back-end data; you can't use differing security attributes of different users to control, for instance, who may write to a particular table.

This security issue can be handled by adding Authors fields and editing history lists to your Notes forms. However, in many cases this puts you to extra work duplicating an already-existing security infrastructure, and for virtual documents it would require you to add fields to your data tables.

Integrated credentials let you leverage your existing relational database security to prevent unauthorized viewing and editing via Notes. You must store user credentials in a Lotus Notes database, which uses document Reader access to prevent users from accessing the credential information of other users. You may let users update their own credential records, or use an LEI Replication activity to update the information from user tables in your relational database (somewhere you would need a table of Notes usernames matched with their RDB userids).

Once the credentials database is in place, LEI can use the user's Notes ID as a key to look up their user ID and password in the external system, allowing them to view and edit only those records they are authorized to, without prompting them for an additional login. In addition, this lets the database system track which user made a revision.

LEI is slower if you use integrated credentials, because:

- The server must do extra work of looking up the credentials for each user on every transaction.
- The server must maintain more active connections to the data source (one for each user as opposed to one shared by all users), so less time is saved by reusing persistent connections.

For more information, see 2.2.2, “Integrated credentials” on page 28.

1.5 What’s new with LEI 6.0 through 6.0.2

LEI 6 was first introduced with Lotus Domino 6. The LEI product has received some enhancements in the follow-on release of LEI 6.0.1. The following section details what's new with LEI 6 in general. We also include a section that details what is different for LEI 6.0.1 on the iSeries server.

1.5.1 What’s new in LEI 6

- LEI product tied to release boundary

Starting in Domino 6.0, the LEI product is bound to the release level. This is a change from past releases of LEI, where the LEI 3.1 product, for example, would run on multiple
releases of the Domino R5.x server. Starting in Domino 6.0, the level of LEI must match the release level. This means that LEI 6.0 will only work with Domino 6.0. Additionally, LEI 6.0.1 will only be able to be installed on a Domino 6.0.1 server, and so on.

► Addition of Advanced RealTime activities

Advanced RealTime activities were first introduced in this chapter in 1.1, “A business case for LEI” on page 2 and are made possible through a new technology called virtualization. Virtualization allows data external to Domino to appear to the end user of the application as though it were native Domino data. There are three new Advanced RealTime activities:
- Virtual documents
- Virtual fields
- Virtual agents

► Server-side browsing

Previous versions of LEI used client-side browsing, which required LEI administrators to install back-end connectivity software on both the Domino server and the LEI development client. For the iSeries, this meant installing a DB2 connect client on the PC where the LEI administration client used to create the connection and activity documents resided. This requirement goes away in LEI 6 due to server-side browsing.

► Development Client retired

Previous LEI versions also required developers to install a separate LEI Development client. Because of server-side browsing, a separate development client is no longer required.

► Improved user interface for LEI Administrator, activity documents, and connection documents

For those of you that used both the DECS and LEI 3.x products, you were used to slightly different user interfaces, including which symbols were used to indicate the status of various activities. What you will find with LEI 6 is a user interface more similar to that available with DECS. As a result, the symbols used are more intuitive than those previously used by LEI.

Additionally, the connection and activity documents have been modified to make them more intuitive as well. You will find these documents easier to fill out and more logically flowing.

► Retirement of the LEI.INI file

Previous versions of LEI used a combination of the LEI.INI file and the NOTES.INI file for environment variable settings. The LEI.INI file has been retired in LEI 6 and all LEI-related variables now exist in the NOTES.INI file. Additionally, some LEI environment variable names have been changed and others have been retired. Following is a list of the variables that have been retired:
- LCINIDelete
- KitType
- LCINIGetString
- Directory
- LEIDirectory
- RemoteConsole

► LCTEST superseded by DCTEST

Previous versions of LEI used a connectivity test tool called LCTEST. With the retirement of the LEI LSX, this connectivity test tool has been replaced with the new DCTEST tool. DCTEST is a client-side tool that verifies connectivity to a back-end data source.

► LotusScript changes
The Scripted activity allows for manipulation of data sent to or returned from a connector. Previous versions of LEI provided a set of classes called the LEI LSX. This set of APIs is no longer available, all scripting needs to be done through the LC LSX classes. Any LotusScript code that contains the Uselsx “lsxlei” syntax will need to be updated to Uselsx “lsxlc”.

- **Timestamp polling**
  This is a new feature allowing polling to be based on a timestamp. This new feature is enabled in the Polling activity document by selecting to enable timestamp polling and then selecting a timestamp field associated with the connection the polling activity is configured to use.

- **Retirement of EDA/SQL connector**
  The EDA/SQL connector is no longer supported in the LEI 6 product.

- **Retirement of Connection Broker metaconnection form**
  As a simplification of the LEI Administrator UI, you no longer create a Broker metaconnection in the LEI Administrator. Instead, this capability can be enabled directly from the various Activity forms using the Integrated Credentials option. The Broker metaconnector still exists for use from LC LSX code, however.
  
  For details, see 2.2.2, “Integrated credentials” on page 28.

- **Retirement of CGI interface**
  The LEI CGI interface has been deprecated in LEI 6. The nleicgi.exe and leicgi files are no longer supported.

- **LEI administrator file name change**
  Past releases of LEI had an LEI administrator file name of leiadm.nsf by default. Users had the option of changing the file name to whatever they liked. In LEI 6, this is no longer the case. The name of the LEI administrator file has changed to decsadm.nsf, and the name of the file cannot be changed by the user. If the name of the file is changed, LEI 6 will not function correctly, it must remain as decsadm.nsf. The LEI template used to generate this file is leiadm.ntf.

- **LEI script vault file name changes**
  The LEI script vault name has changed as well. It is now leivlt6.nsf in LEI 6. The template used to generate this database is leivlt.ntf.

- **Migration utility**
  There is a migration utility built into the LEI 6 product. This utility is used to migrate LEI 3.x connection and activity documents to LEI 6. This migration is required for connection and activity documents created in previous versions of LEI to work with the LEI 6 product.
  
  The migration utility can be run during the installation and configuration of the LEI 6 product or it can be run separately at a later time of your choosing. This utility can also migrate scripts found in the Release 3.x LEI script vault (leivlt.nsf) to the LEI 6 script vault (leivlt6.nsf) in order to convert their format from using the LEI LSX classes to the LC LSX classes.
  
  For details on using the migration utility see Chapter 4, “Migrating from Lotus Enterprise Integrator 3.x” on page 57.

- **Addition of browsing capabilities in metaconnectors**
  Metaconnectors are special connections that can be built over connection documents. They wrap around a base connection document and provide additional functionality over the base connection. There are four metaconnectors available:
  
  - Order
Browsing functionality has been added to Metaconnectors in LEI 6. To use this new feature, you must select a metadata object (table, form, view, and/or procedure) in every connection that is used with the metaconnector. If connections referenced by the metaconnector do not have metadata selected, a message appears notifying you that no metadata has been selected in the connection reference in the metaconnector.

- Return of output parameters upon update or insert using Direct Transfer activities and LEI Advanced RealTime activities

LEI 6 supports the return of output parameters on update or insert into the specified data source. Fields designated as OUT or IN/OUT in a stored procedure can be returned to LEI upon completion of its task. You can enable output parameter support in order to update a field in the data source when the data transfer to the target is successful.

If the stored procedure is a designating metadata object, LEI 6 allows the stored procedure to send output back to the source metadata through parameters. This feature is implemented by each connector uniquely. At the time of writing of this redbook, the only connector that supports this feature is the DB2 connector. The SAP R/3 Release 1.7 connector is planning to support this feature in a future version of the connector.

- Ability to e-mail activity log files as plain text

Some of the previous versions of the LEI product allowed you to e-mail activity log files to designated recipients through a Notes rich text format as a doclink. In LEI 6, these activity log files can additionally be sent in plain text format. This is a nice feature for recipients who are receiving these messages from a pager or PDA device.

- Addition of integrated credentials database

One of the Advanced RealTime features provided with LEI 6 is the usage of integrated credentials. Integrated credentials allows you to map a Domino user ID to a user ID and password on the back-end data source server. The lcired.ntf template is used to create this database.

For more details, see 2.2.2, “Integrated credentials” on page 28.

- IBM iSeries support

LEI 6.0 is the first version available for the iSeries platform. There are significant differences between this first iSeries release and other supported platforms. For details, consult the redbook Implementing IBM Lotus Enterprise Integrator 6 on the IBM @server iSeries Server, SG24-6941.

1.6 Terminology

We have all found by working on different technologies that they each have their own unique descriptions and terms when describing essentially the same functions and features. This also true of the terminology between relational databases, LEI, and Lotus Notes. Notes is not a relational database system, so for some concepts there’s not an exact corresponding entity in the other system. This section explains and correlates the terminology between SQL, LEI and Lotus Domino. There may in addition be terminology differences between the SQL for accessing a relational database, and the physical representation of that database on the end system (for instance, what SQL calls a “Table” corresponds to a “Physical file” in DB2 UDB), but there are too many database systems to cover this terminology here.
In addition to this comparative description, also check the “Glossary” on page 253, and the index, for any terms you need explained.

Table 1-1 shows a side-by-side comparison of the different environments. Starting at the highest level of containment type of object going down to the lowest level of object, we list corresponding concepts on the same row.

Table 1-1  Terminology across RDB, LEI and Lotus Domino

<table>
<thead>
<tr>
<th>Relational database/SQL</th>
<th>LEI</th>
<th>Lotus Notes/Domino</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection, Owner, or</td>
<td>Owner</td>
<td>Database (.nsf file)</td>
</tr>
<tr>
<td>Schema</td>
<td>Metadata</td>
<td>Form or View</td>
</tr>
<tr>
<td>Table</td>
<td>Connection document</td>
<td>Document collection</td>
</tr>
<tr>
<td>View</td>
<td>Connection document</td>
<td>View entry collection</td>
</tr>
<tr>
<td>Index</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Row</td>
<td>Row</td>
<td>Document or view entry</td>
</tr>
<tr>
<td>Column</td>
<td>Field</td>
<td>Field</td>
</tr>
<tr>
<td>Procedure</td>
<td>Stored procedure</td>
<td>Agent</td>
</tr>
<tr>
<td>Statement</td>
<td>Statement or Command</td>
<td>Formula</td>
</tr>
</tbody>
</table>

Let’s take each of the rows in Table 1-1 and examine them in more detail.

**Collection, schema, owner, database**

In the relational world, a collection, scheme, or owner is a way of logically grouping metadata, indexes, and other data entities. When you refer to an entity in SQL, you would write `schema.entityname`.

Within the context of LEI, the terminology used is *owner*. The owner is generally the RDB userid of the person who created the objects that you’re connecting to in an LEI connection document.

For Lotus Domino, the nearest comparable structure is a Domino database. This is the highest level of container in the Domino world. However, you don’t use SQL for access to Domino, and you don’t refer to Domino entities using a “.” notation.

**Metadata, form or view**

Relational databases contain information that describes their tables and views, listing the field names, data types, and other attributes. This information is accessible via the Catalog command in LEI.

In Notes, the nearest corresponding object is the form. A form has a name, like the metadata name in a relational database, and contains fields and datatypes, like the columns defined in metadata. The Notes form also combines this with some business logic (translations, validations, actions, and access controls), and presentation (the layout of the fields on the form, static text, decorations). From an LEI standpoint, we don’t care about presentation, but you may optionally make use of the translations and validations.
LEI can access Notes data either by finding all documents that use a particular form, or by reading the columns from a Notes view. So in a sense, a Notes “view” design element is also metadata, but in LEI terminology, a Notes view corresponds to a relational index.

Table, connection document, document collection
Going the next step down, we look at the actual data described by the metadata or form.

A collection of Notes documents is roughly comparable to the rows in a table. Each element of the collection has a set of field values associated with it. Notes documents created with the same form will have generally the same set of fields stored in them.

However, a Notes document is actually a free-form container. Any Notes document can in theory contain any set of field values, depending how it was created. A Notes agent, for instance, can easily create documents that have fields not on any form, or fail to create all the fields that are on the form. LEI forces Notes access into a relational model, but it's important to remember Notes' differences from relational databases also.

Unlike data stored in separate relational tables, Notes documents are not segregated based on their form. A Notes database is a container of documents. If a Notes database contains five forms, all documents are in the same database no matter that form they were created with. When you issue a request to Notes for documents of a particular form, therefore, the Notes connector has to filter through all the documents in the database to find the ones that match. This makes “Select” operations against Notes document collections inherently less efficient than relational selects. To get better performance, you can use a view, which is already pre-filtered and sorted.

LEI doesn’t itself contain any data, but just mediates access between data sources, so there’s no direct corollary to this data store within LEI terminology. But a connection document is the entity in LEI that allows access to a table.

View, connection document, view entry collection
In relational database parlance, a view is a logical organization of the data in one or more tables. It may filter the table rows so that you only see ones that meet certain criteria. You can use it as if it were a table, except that you might not be able to write to it.

A Notes view is similarly a way to organize Notes documents. It includes a “selection formula” that controls which documents appear in the view. Frequently, selection formulas use the Form attribute of the document to include only documents that use a particular form. Columns can be sorted, and one or more usually are, so the view also has the attributes of a RDB index.

Just as you can save time by selecting records from a relational view instead of searching the whole database, you can save time in selecting Notes documents by basing your selection on a Notes view.

Notes views do not support “join” operations — all the data on each row comes from a single document. However, since Notes documents can contain multivalued fields, the data in a single Notes document may well correspond to data in a relational system you would have to join multiple tables to get. The examples in Chapters 7 and 8 demonstrate this.

Row, document or view entry
Continuing down the levels of granularity, an entry in a relational table or view is called a row. LEI also refers to this as a row. The nearest parallel in Notes is a document or view entry (depending whether the Notes connection document selects to read data from a document or from view columns).
There is also an option to make a single Notes document behave like multiple rows if some of
its fields are multivalued. This is used when relating a Notes document to a relational table in
a one-to-many relationship. Chapter 7, “Batch activities” on page 105 provides specific
examples which illustrate this concept in detail.

Column, field
LEI refers to a single value from a single row as a “field”. This is also the terminology used in
Notes. In the LEI context, a field may have multiple values (for instance, if it comes from a
multivalued Notes field). You have options to store such values in a relational database as
separate rows in a table, as a delimited string, or as an exact binary image of the field value.

Stored procedure, agent
Agents written in Domino Designer can operate on Notes documents in much the same way
as a stored procedure in a relational setting. They are dissimilar in that a Notes agent can’t
return a data set (although it could store information for LEI to read). Agents also don’t take
parameters. The current Notes connector doesn’t support executing an agent, but LEI
includes a “Scripted activity” tool that lets you run a Notes agent on a schedule; refer to 7.5,
“Scripted activity” on page 125.

Statement, formula
Data is extracted, manipulated, added to, or removed from a relational database with an SQL
statement. In an LEI context, these are frequently called commands, as in the Command
activity. 7.2, “Example 1: Setting up a Command Activity” on page 106 provides a detailed
example of a command activity.

Lotus Domino doesn’t support SQL; its command language is the macro formula. Depending
on context, this may be a selection formula used to filter documents for reading, or what LEI
refer to as a “transformation” formula, used to change the values of document fields.

The Notes formula language doesn’t support document creation or deletion (at least not in the
contexts where LEI uses them). The Notes connector implements these using Notes’ internal
API.

1.7 Data type conversions/mappings

One of the most difficult parts of any enterprise integration project is understanding how data
types in one data source map to data types in a disparate data source. Different database
platforms can have different names that correspond to similar types, or special types for which
there is no exact match in another system.

Since the Lotus Connectors and LEI let you execute any SQL statement in the syntax
accepted by whatever back end you’re connected to, you do have the ability to create tables
containing fields of any type the RDB supports. But when you read and write data, the fine
distinctions between, say, the DB2 INTEGER and SMALLINT types are somewhat blurred,
because all the types are represented within LEI as one of seven basic internal types. These
LEI Data types are shown in Table 1-2.

<table>
<thead>
<tr>
<th>LEI Data Type</th>
<th>Description / Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCY</td>
<td>Represents any value of a Currency type.</td>
</tr>
<tr>
<td>DATETIME</td>
<td>Date and time, date only, or time only.</td>
</tr>
</tbody>
</table>
The Domino Designer Help contains a chart for each RDB connector listing the types supported by that RDB and showing how those types correspond to the LEI types. In the chapter outline, look under Lotus Connectors, Connectivity Guide, and the chapter heading for your database system.

When trying to copy information from one system to another, LEI will do its best to convert between the different types, insofar as this is possible. For instance, it will convert between a Notes Currency field and a DB2 NUMERIC, or between any two vaguely similar types. When defining an activity, you have the option to say what happens if the value can't be copied exactly. For instance, if the destination stores fewer digits of numeric precision than the value contains (precision loss) or if a string field is too small to contain the string (data loss). You may opt to have the activity generate an error in such cases, or to accept the data or precision loss.

The redbook *Implementing IBM Lotus Enterprise Integrator 6 on the IBM iSeries Server*, SG24-6941, contains a much more comprehensive data conversion guide geared to DB2 specifically.

<table>
<thead>
<tr>
<th>LEI Data Type</th>
<th>Description / Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>Integer numbers (32-bit)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Floating point numbers (Double size)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>Decimal numbers up to 88 digits precision.</td>
</tr>
<tr>
<td>TEXT</td>
<td>Any and all string data, using any character set, fixed or variable size.</td>
</tr>
<tr>
<td>BINARY</td>
<td>BLOB or binary field types, Notes rich text fields, Notes multivalue fields, and so on. BINARY field types may also store text.</td>
</tr>
</tbody>
</table>
Chapter 2. Architectural scenarios and considerations

This chapter discusses different possible architectural scenarios for implementing Lotus Enterprise Integrator (LEI) 6. Considerations for security and performance when designing a solution based on LEI 6 are also covered.
2.1 Architectural scenarios

The architectural scenarios presented in this chapter are examples of how an LEI 6 infrastructure could be implemented. They are not a complete list of how LEI could be set up; rather, they are intended to show typical architectures designed to address specific needs and network conditions. Once you understand the constraints and performance characteristics imposed by the different setups, you can certainly come up with your own configurations.

Since LEI works more or less identically with several different operating systems and database platforms, these examples omit mention of specific databases and operating systems — instead, we just refer to all relational databases as RDB, for instance.

The only significant exception is the iSeries platform, which at the time of this writing supports no relational connectors except DB2. The redbook *Implementing IBM Lotus Enterprise Integrator 6 on the IBM @server iSeries Server*, SG24-6941, discusses specific configurations for dealing with this.

2.1.1 Scenario 1: Single Domino/LEI server

This scenario, as shown in Figure 2-1, is a simple example with all components on one server or (for OS/400) a single logical partition (LPAR).

This type of architecture is most practical for test or pilot environments. There is a performance advantage with this architecture because all components are on the same physical machine, which eliminates overhead from network communication.

The one server hosts the Domino server, a Notes/Domino application, an LEI server, and the relational database that is accessed by LEI. For virtual documents or virtual fields, when an end user accesses a Notes document through a Web browser or Notes client, the relational data will be accessed in real time and displayed in the document. For data transfer activities, the data can be been directly transferred or replicated between Domino and the RDB.
**Description of components**

The following describes the components in this scenario:

- **End users**: Users access the Notes/Domino application from a Web browser or Lotus Notes client. If you’re just testing LEI functionality for learning, you can even install the Notes client on the same box, if it’s a Windows® OS.
- **Domino 6.0.1 server**: The Domino server hosting this Notes/Domino application.
- **LEI 6.0.1**: The LEI server runs either as a Domino server task, or in a separate “console” window.

The LEI Administrator database is on the Domino server and is used to create and maintain LEI connection and activity documents. Several connection documents can exist to various forms in multiple Notes/Domino applications (though only one is shown here), and to various tables and views in the RDB. For more information on connection documents, see Chapter 6, “Connection documents” on page 73.

Several activity documents may also exist between the Domino applications and RDB. It’s not unusual to have multiple activities to the same Notes form, as the examples in Chapters 7 and 8 show.

- **RDB**: The data required by the Notes/Domino application may be distributed in different tables in the relational database.

**2.1.2 Scenario 2: Remote database servers**

The architectures discussed in this section are the most common in real world situations since there is normally a physical separation between LEI and your RDB servers. These designs may be used when there is a business need to get data from various remote locations into one central point like a Notes/Domino application. The reverse may also be true where the data is initially entered into multiple Notes/Domino applications and is then moved to one or many RDB files.

The examples in this section show architectures that access relational databases remotely. To access remote databases, the LEI server box must have a remote database client installed. For instance, to access a DB2 database you could use the DB2 Connect™ client from IBM, or the DataDirect ODBC drivers.

In this diagram, the same relational client software is used to connect to multiple external databases. It’s also possible to install several different RDB clients on the same LEI server and use, say, Oracle and DB2 data in connection with different fields on the same Notes form.

![Figure 2-2  LEI access to remote databases](image-url)
2.1.3 Scenario 3: LEI access to multiple relational back-ends

Since LEI activities also support data movement between two relational back-ends, there doesn’t need to be a Notes/Domino application involved. You might just replicate data between a DB2 table on one server and a Sybase table on another, for instance.

![Diagram of LEI setup for replication between diverse databases](image)

The Domino server might well be used to host Notes applications also, and perhaps some of these applications have LEI activities associated with them that connect the Notes applications to DB2 or Sybase. But it might also be the case that the Domino server is dedicated to LEI replication, in which case there wouldn’t need to be any Notes applications or clients (except the Notes client used to access the LEI Administrator application).

**Note:** In cases where they just need to replicate remote databases, some customers have set up LEI to run on a workstation with just the Notes client installed — no Domino server. If no Advanced Realtime activities are used, LEI can run without Domino.

2.1.4 Scenario 4: LEI 6 Advanced Realtime plus Notes replication

Fields and documents that are “virtualized” by a Virtual Fields or Virtual Documents activity, work like regular Notes data for purposes of replication to other Notes servers. This is a powerful feature for sharing the information on other Notes servers and for providing your sales force and other mobile users with portable copies or subsets of your enterprise data. Since Notes R5 servers and clients can replicate with Notes 6 servers, this also lets you take advantage of LEI’s new features without needing to upgrade your entire infrastructure to Notes 6.

To take advantage of this ability, set up other Domino servers to replicate the information, just as you would for a regular Notes application. As Figure 2-4 shows, you can adjust the replication interval independently for each server, based on the level of need for the information to be current and the available network bandwidth.

In addition, you can arrange for each replica to receive only those documents needed at that site. This saves storage and bandwidth, especially important to those laptop users who have to phone in. Notes replication includes other space and time saving features, such as the ability to receive truncated documents which can be retrieved in full on request (the request can be made while offline, to be completed in the next replication).
By default, Notes replication is bidirectional, so changes made in any of these replicas will eventually flow back to the RDB. Notes access controls are effective in all replicas, including local replicas\(^1\).

This decentralized architecture also makes your application more resistant to outages. Even if Server 1 is down, users can continue to work in the replicas at their site, and when Server 1 comes back up, it will receive their changes via replication.

This architecture also lends itself well to bidirectional Replication activities using a timestamp. Refer to 7.7.2, “Timestamp replication” on page 131 for additional details.

Note: For Advanced Realtime, a Notes Connection document is not required. Refer to Chapter 8, “Advanced RealTime activities” on page 151 for additional details.

2.1.5 Scenario 5: Decentralized Advanced RealTime access

This configuration allows real-time access to the same relational data across distributed replicas of a Notes/Domino application. In this scenario, all replicas access the same relational data directly via a Virtual Fields activity. This scenario would be appropriate when there is a Notes form in an application that requires real time access to data.

An example of this scenario would be an expense reporting application that has a form with current exchange rates approved by the company. The approved rates are stored in a RDB table indexed by currency type. The distributed Notes/Domino application needs instant access to the approved exchange rates that are stored in the table.

\(^1\) The “Consistent access control” database access property extends full access control to local replicas. Even if this option is not enabled, however, users cannot receive documents by replication which they don’t have access to view, nor replicate changes to a server unless the server recognizes their access to make those changes.
This is a good application for a Virtual Fields activity, because the list of currencies — the keys — changes relatively infrequently, although the data associated with those keys may change daily or even hourly. When new key documents are added, they can be created in one Domino database and allowed to replicate to the others. Different LEI servers running "the same" activity can detect whether a document update that arrives by replication has already been written to the relational table, so you should not get unnecessary updates or duplicate keys by doing this. Note, however, that if the relational data are updated by an outside process (not through Notes), then updates will not necessarily replicate to systems that aren't running the activity (such as the laptop shown in this diagram). For details about this, refer to 8.6.7, "Domino replication involving virtual documents" on page 178.

To be considered identical, two Virtual Fields activities on different servers must have identical database connection properties. The name of the activity does not need to match, but you should probably copy and paste the connection document from one LEI Administrator database to another, and make sure all the activity options are the same.

A Virtual Documents activity might also be used in this context, but in that case each replica must use its own external key table, and they must not replicate these documents with each other, or there will be duplicates.

The architecture shown in Figure 2-5 can be used to distribute an application that uses virtual fields or virtual document activities. Replicas of the Notes application on every Domino server each have real-time access to the relational data. Contrast this with 2.1.1, "Scenario 1: Single Domino/LEI server" on page 22 where a single server has live access to the data, and uses Domino replication to synchronize other replicas. With this scenario, all replicas would be instantly synchronized if data in the application on any server changes. To do this, every server needs to have LEI installed, with its own copy of the LEI Administrator database (not a
replica copy). Every server has a virtual field or virtual document activity running against its
own replica copy of the Notes/Domino application. All replicas point back to the same RDB
table.

Note:
- In this scenario, the LEI Administrator database (not a replica) must be installed on
each Domino server with a Virtual Fields activity defined.
- For Advanced Realtime, a Notes Connection document is not required. Refer to
Chapter 8, “Advanced RealTime activities” on page 151 for additional information.

2.2 Security considerations

For LEI to connect to most relational databases, the LEI server must pass a valid database
user ID and password to the RDB server. This user must have access rights to select, insert,
and delete data in certain tables. With LEI 6 Advanced Realtime, there are two options for
authenticating users:
- A generic user ID and password in the connection document.
- Integrated credentials, where each Notes or Web user can have their personal database
login used automatically.

2.2.1 Generic user ID and password

The LEI Administrator's connection forms for every different relational database, all have
fields to enter the userid and password.

By default, activities that use this connection document, will use the authentication
information you enter here.

That means only one database login is used for all activity in the database. This has
advantages and disadvantages. The advantages are:
- It's simple.
- It's more efficient because a single persistent connection can be used for all users.

The disadvantages are:
- It bypasses the database access controls, giving all users the same access to read and
write all tables accessed through that connector.
- You can't use the RDB's change tracking to see what transactions were done by what
users.
- The userid and password are stored in the connection document. In case LEI has more
than one person administering it, they might like to keep this information private from the
other administrators.

Passwords may optionally be encrypted using an encryption keys you create and share
with other trusted administrators. The access control list (ACL) of the LEI Administrator
database should also be set to prevent non-administrators from accessing this database.

Tip: Read access to individual connection documents and activity documents can be
controlled using database ACL roles and the Security tab of the document properties
dialog. Be careful not to delete your own read access to a document, as you may find it
difficult to recover. Also, the Notes ID used to run LEI must have access to these
documents.
2.2.2 Integrated credentials

Integrated credentials is a new feature in LEI 6. It allows for the mapping of Notes user IDs to user IDs and passwords on external systems.

**Note:** Integrated credentials is only supported for the Advanced RealTime activities (virtual agents, virtual documents, and virtual fields) and not for the batch LEI activities such as data transfer or replication.

With integrated credentials, a user opens a Notes/Domino application using their personal Notes user logon. When they do a transaction that requires accessing relational data, LEI first looks up their name and the database type and name in an integrated credentials database. Use the LEI Credentials template (leicred.ntf) to create this database. You may have multiple credentials databases on a single server, and you may use Notes replication to synchronize them between Domino servers.

The activity document specifies which credentials database to use. If the user’s name is found there, and their records contain a match for the database they’re trying to use, LEI uses the userid and password stored in that record to log them in automatically.

Optionally, you may set the activity to use the credentials stored in the connection document if a match is not found. This lets you establish a base level of access for all users — read-only, perhaps — and issue personal accounts only to users who need access to edit the data.

Integrated credentials only work for the three Advanced RealTime activities (virtual agents, virtual documents, and virtual fields), because these are the only activities initiated by user requests. Batch activities use the credentials stored in the connection document. You should enter a username and password in the connection document even if you plan to use integrated credentials, both to allow batch activities to run, and so that you can browse metadata while creating connections and activities.

The advantages of integrated credentials are:

- You can leverage your existing RDB security, instead of having to duplicate it using Notes Readers and Authors fields.
- You can use the RDB tools to track which users make which transactions.

Disadvantages include:

- The contents of the credentials database must be kept up to date. This might be done by users editing their own records (Readers fields in the documents prevent users from seeing records other than their own), or through manual editing by administrators. If you have a relational table somewhere associating Notes usernames with database userids, you can probably program a replication activity to keep it up to date. However you do it, though, it's extra work.
- Integrated credentials give generally poorer performance than generic credentials, as explained in 2.3.2, “Performance spectrum scenarios” on page 34.

**Configuring integrated credentials**

To use integrated credentials you must create an integrated credentials database, then add users to the database. Once this is done, you can then reference this integrated credentials
database from the LEI Advanced RealTime activity documents. Perform the following steps to create and setup the integrated credentials database:

1. Create a new Domino database from the LEI Credentials template (leicred.ntf). See Figure 2-6.

2. Open the new database, and use File / Database / Access Control to open the access control dialog. Set the accesses appropriately based on your local database access standards. The access level shown in Figure 2-7 lets users edit their own documents.

If you update this database using an automatic process, then you can set the default to No Access. The minimum access required for the credentials database to be usable, is Reader.
access for the userid used to run LEI (the KeyFileName or ServerKeyFileName ID in the notes.ini file).

3. Use the **Create Credential Entry** button to create user mappings for each user (Figure 2-8).

![Figure 2-8 Integrated credential database](image1.png)

3. Use the **Create Credential Entry** button to create user mappings for each user (Figure 2-8).

4. For each user mapping, enter the Lotus Notes user ID, as well as the relational database user properties. These properties include the connector (that is, DB2 UDB, Oracle, and so on) external database name, external user ID and password. The password is not visible in the document. See Figure 2-9.

![Figure 2-9 Completing the LEI integrated credentials form](image2.png)

4. For each user mapping, enter the Lotus Notes user ID, as well as the relational database user properties. These properties include the connector (that is, DB2 UDB, Oracle, and so on) external database name, external user ID and password. The password is not visible in the document. See Figure 2-9.

5. When prompted for the external system password, as shown in Figure 2-10, enter the password for the external user ID. Notice the entered password is not visible.

![Figure 2-10 Setting the external system password](image3.png)

5. When prompted for the external system password, as shown in Figure 2-10, enter the password for the external user ID. Notice the entered password is not visible.

6. Once the integrated credentials document is complete, it is displayed in the main view of the integrated credentials database. This view is categorized by Lotus Notes username (Figure 2-11).
Using integrated credentials in Advanced RealTime activity documents

Once the integrated credentials database has been setup, you can use it in any of the three Advanced RealTime activity documents.

In the Advanced RealTime activity document, under the Integrated Credentials tab, click the Lookup Credentials option. You then have to fill in the following options as shown in Figure 2-12:

- **Missing Credentials**: This field allows you to choose whether you want to use the generic user ID and password that is specified in the connection document if no match is found for the current user in the integrated credentials database.
- **Credentials DB Filepath**: Relative file path of the integrated credentials database.
- **Connection Cache Size**: This allows you to specific the maximum distinct connections that are active at any one time. This is an important setting for performance, since establishing a connection can take a noticeable amount of time with many database servers. If the setting is too low, connections will be too often discarded to make room for other users' connections, causing a delay for connection before their next operation. If too high, you're wasting memory.

Considerations when implementing integrated credentials

The following points are architectural considerations when using integrated credentials:

- Is a generic user ID and password a sufficient security mechanism for your application?
- Management overhead. Some one must create and maintain the user ID and passwords in the integrated credentials database.
- Performance overhead of integrated credentials.
- Optimum setting for the Connection Cache Size field in the LEI advanced realtime activity document.

**Note**: Even when you use integrated credentials, your connection document must have a valid userid and password to support activity initialization.
2.2.3 Fields and public key encryption

If a field in a Notes document is encrypted using Public Key Encryption, the corresponding fields in the RDB will be nulled. Notes users with the correct encryption key will be able to see the encrypted data as usual in Notes. Make sure that the relational column that you map to a field that has encryption enabled, does not have a constraint preventing null values, or uniqueness constraints.

This feature is also important to keep in mind if you want to have any programs directly accessing the data in the external system. For more information on this topic see “Virtual Documents and Public Key Encryption Key” in Chapter 13 of the LEI Activities and User Guide.

2.3 Performance considerations

The performance of a Domino application is a very complex topic and the addition of LEI adds another layer of complexity. The following sections provide some tips and hints on what can be done to improve the performance of your LEI applications. We also point out items to lessen the impact to the performance of LEI applications.

For information on general Domino application performance refer to the redbook Performance Considerations for Domino Applications, SG24-5602, which can be found at:

http://www.redbooks.ibm.com

2.3.1 LEI considerations

This section reviews performance considerations relative to LEI activities.

Advanced RealTime activities

Following are performance considerations for some of the Advanced RealTime activities, specifically virtual fields and virtual documents:

- **Persistent connections**: For Advanced RealTime activities, the Max Connections parameter in the activity document sets the number of persistent connections. A larger number of connections may improve performance if there are many users using the Notes application concurrently. This parameter defaults to 2, which may be less than ideal in some production environments. Lotus recommends setting this parameter to 2 or 3, and if users experience significant delays then to increase the number.

  For more information, refer to 2.3.4, “Persistent connections” on page 35.

- **Integrated credentials**: If integrated credentials are used, there may be a performance cost because the user ID and password passed to the back-end data source is always changing. This lessens the performance improvement normally gained with persistent connections. See 2.2.2, “Integrated credentials” on page 28 for more details.

- **Internal keys versus external key table**: Virtual documents that use internal keys will perform better than those that use an external key table. If internal keys are used, the control fields are added to the DB2 UDB table that has a virtual document processing over it. In this case, only the table being virtualized needs to be access to perform the virtualization.

  If an external table is used to hold the control fields, a join must be performed each time data is virtualized in the Domino application. This requires extra processing and will perform less optimally than using internal keys. This is particularly true if the file that is being virtualized contains many records.
For more information on virtual documents, see 8.6, “Virtual documents activity” on page 172.

- **Index the three internal key fields**: If using internal keys for Virtual Documents, index the EINOTEID, EIUNID, EIMODIFIED fields. Indexes will not automatically be created over these control fields; you need to create them manually.

- **Index keys for external key table**: If an external key table is used for a virtual document activities, the join key fields in both the external key table, as well as key fields in the data table should be indexed. Every time a virtual document is accessed a SQL join is done on the two tables. Indexing the key fields that the join operation is done on may substantially improve performance.

  The built-in “Create external key table” utility in the virtual document activity, will automatically create the recommended indexes.

- **First time start on virtual document activity**: Initial population of the internal keys or the external key table may be slow and make the application appear to be slow.

  **Note**: If the virtual document activity is accessing a larger relational table, it’s best to start it for the first time during non-production hours. The scheduling options permit you to arrange this in advance so that you need not be present.

- **Remote database server**: Accessing a remote database over a network will be slower then accessing a database that is local to the LEI server. Local database access removes all network overhead. When architecting your application solution, you need to keep in mind the how much data will be transferred and at what rate to determine if the network bandwidth will be a concerning factor.

**Batch activities**

Following are performance considerations for the batch activities, specifically direct transfer and replication:

- **Direct transfer, Number of Records to Transfer Concurrenty**: This parameter is set to 1 by default. It enables the buffering of records before a transfer is done. Setting this number higher then 1 will generally improve performance for direct transfer activities. A rule of thumb is to set this to be 10% of the number of total expected rows of data you expect to transfer. For example if you expect to transfer 1000 rows of data, you can set this parameter to 100. If the relational database supports multi-row operations, this option may improve performance by reducing “overhead” — fewer fetch requests and responses need to be sent, even though each response contains more data.

- **Timestamp replication**: For the replication activity, timestamp replication will normally be faster than primary key replication. Timestamp replication only needs to look at records that have a timestamp greater than the last time the replication activity ran. This greatly reduces replication times for most environments.

  With primary key replication, LEI reads each record in both the source and target data sources and compares those with matching keys for differences. Only the records that have changed will then be replicated.

  Primary key replication can also be slowed if the systems being replicated have different text sorting algorithms. If the key values don’t sort in the same order, the replication activity will do extra inserts and deletions because it fails to match records that are already identical. Use an order metacommector (page 101) if needed to ensure consistent sorting.

- **Remote database server**: Accessing a remote database over a network will be slower then accessing a database that is local to the LEI server. Local database access removes all network overhead. When architecting your application solution, you need to keep in
mind the how much data will be transferred and at what rate to determine if the network bandwidth will be a concerning factor.

2.3.2 Performance spectrum scenarios

To help pull all of this information together, the following table shows which features and options slow or speed up Advanced Realtime activities.

<table>
<thead>
<tr>
<th>Slower</th>
<th>Faster</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web browser or mobile client</td>
<td>Lotus Notes client</td>
<td>The server must do 2-3 times more work to support a Web user (also true for non-LEI applications).</td>
</tr>
<tr>
<td>Remote database</td>
<td>Local database</td>
<td>Depending on server load, generally it’s faster to reference local data than to read over the network.</td>
</tr>
<tr>
<td>Integrated credentials</td>
<td>Generic userid from connection document</td>
<td>Integrated credentials require extra time for credentials lookup, and more connections are created for the same number of users.</td>
</tr>
<tr>
<td>External key table</td>
<td>Internal keys</td>
<td>This factor applies to Virtual Documents only.</td>
</tr>
</tbody>
</table>

This is not to say you should always use the faster option — there are functional differences, so sometimes your application may require an option that makes it slower. Even an application that chooses all the slowest options won’t necessarily have unacceptable performance. Much depends on other factors, such as how many users there are, network speeds, database optimization or lack thereof, and number of records in database.

You may also trade off performance versus timeliness of information. If your remote RDB is accessed over a slow connection, your users will get much better performance if you use a Replication activity (see 7.7, “Replication activity” on page 130) to migrate changes between RDB and Notes, instead of using Virtual Fields or Virtual Documents over that connection. However, the Notes information may be slightly out of date if other applications are making changes to the relational data.

The design of your Notes application may also have significant effect on performance. Consult the Domino Designer documentation and the redbook Performance Considerations for Domino Applications, SG24-5602.

2.3.3 RDB performance considerations

In addition to looking at the overall architecture of the Domino LEI application as we did in 2.3.1, “LEI considerations” on page 32, there are relational database performance tips you should take into consideration as well.

- **Indexes**: Indexing fields that are queried often or used in SQL joins, can significantly improve performance. The white paper Indexing Strategies for DB2 UDB for iSeries contains good pointers for tuning DB2 — many of these are applicable to relational databases generally. This paper can be found at:

Views versus tables: Logical views may perform better than accessing a table directly. For example, if you have a table with 20 columns and 500 rows, but LEI only needs to access 5 columns and 100 rows, you will most likely get better performance if you access a view that filters out the unneeded rows and columns.

For more detailed information on DB2 UDB performance factors, you can reference the following iSeries performance resources Web page at:

As regards other database systems, consult the vendor's documentation for tuning tips.

2.3.4 Persistent connections

After an Advanced RealTime activity starts up, the first time someone requests some data through that activity, LEI creates a connection to the data source for that activity. LEI keeps this connection open as long as the activity is running, to more quickly process subsequent requests.

LEI supports some parallelism, to process multiple requests of the same activity simultaneously. Processing an Advanced Realtime operation can involve significant I/O waits; while waiting, there's no reason LEI can't be setting up and sending out a request for data for another user. However, that requires a second connection to the same data source. Advanced Realtime activity forms contain a setting for “Max Connections.” This controls how many requests the activity can process simultaneously. If the existing connections are already in use, LEI will create an additional connection to the same data source, up to the limit you specify. None of these will be disconnected until the activity shuts down.

If using Integrated Credentials, the activity maintains an active connection for each user, from when they first request information from the database, until the activity shuts down.

The idea of persistent connections is related to, but not the same as, the “connection pooling” feature used by the LC LSX (see 9.5, “Connection pooling” on page 208). The connection pool collects unused connections that originate from any script – not just from a single activity.
Installing and configuring the LEI Administrator

LEI is implemented on several platforms, and there are some differences in the install process between the platforms. This chapter shows where to find the install documentation for your platform, and discusses those issues of installation and administration that are the same on all servers.

This chapter covers the following topics:

- Pre-installation activities
- Installation of the LEI product code
- Administration of LEI
3.1 Where to find the install documentation

Since there are some differences among the supported platforms and possibly from version to version also, this book doesn't try to document the installation process for each server. For the most up to date information specific to your platform, you must consult the documentation that comes with the installation kit. The file leiig.nsf is a Notes Help database containing a chapter on each platform, giving detailed installation steps.

To read leiig.nsf, first copy it to your local Notes data directory (generally C:\Notes\Data or C:\Program Files\Notes\Data) or the Domino data directory on the server. From the Notes client, use File -> Open Database to open it.

If installing on the IBM iSeries server, also consult the redbook Implementing IBM Lotus Enterprise Integrator 6 on the IBM @server iSeries Server, SG24-6941, which has a very thorough, step-by-step guide to installation written specifically for iSeries.

3.2 Pre-installation activities

To work properly, the Notes IDs used to run LEI and the LEI Administrator need certain permissions on the Domino server. The following sections describe how to assign these permissions.

3.2.1 Domino server document changes

You can find the Domino Server document easily in the Configuration tab of the Domino Administrator client. Otherwise, look for it in the server's Domino Directory under Configuration / Servers / All Server Documents.

Edit the Domino server configuration as follows, before installing and configuring LEI.

1. As displayed in Figure 3-1, modify the Programmability Restrictions section, located under the Security tab of the Domino server document. Add the user Enterprise Connector Products/Lotus Notes Companion Products to the following fields:
   - Run unrestricted methods and operations
   - Run restricted LotusScript/Java agents

   This gives the LEI Administrator server agents, access to run on the Domino server. Otherwise, you must re-sign all the design elements with an ID that already has this access. If you don't do this, you will be unable to browse metadata from the LEI administrator, making it extremely difficult to create any activities.

<table>
<thead>
<tr>
<th>Programmability Restrictions</th>
<th>Who can -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run unrestricted methods and operations:</td>
<td>Enterprise Connector Products/Lotus Notes Companion Products, leidemanITSG</td>
</tr>
<tr>
<td>Sign agents to run on behalf of someone else:</td>
<td></td>
</tr>
<tr>
<td>Sign agents to run on behalf of the invoker of the agent:</td>
<td></td>
</tr>
<tr>
<td>Run restricted LotusScript/Java agents:</td>
<td>Enterprise Connector Products/Lotus Notes Companion Products, leidemanITSG</td>
</tr>
</tbody>
</table>

*Figure 3-1 Domino server document - Security tab - Programmability restrictions section*
2. In the Server Access section of the Security tab, grant access for the user *Enterprise Connector Products/Lotus Notes Companion Products* to use the server.

Figure 3-2 displays the required addition to the Access server field. Ensure the user name *Enterprise Connector Products/Lotus Notes Companion Products* and the Domino server name of the LEI server are both included in this field.

<table>
<thead>
<tr>
<th>Server Access</th>
<th>Who can –</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access server:</td>
<td>✔️ users listed in all trusted directories and Enterprise Connector Products/Lotus Notes Companion Products, leidomservTSO</td>
</tr>
</tbody>
</table>

*Figure 3-2 Domino server document - Security tab - Server access section*

3. Go to the Ports tab, Notes Network Ports subtab, and check the address in the Net Address field. Make sure it contains a correct DNS host name or IP address to the server. See Figure 3-3.

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Notes Network</th>
<th>Net Address</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP</td>
<td>TCP</td>
<td>NETWORK1</td>
<td>9.5.32.79</td>
<td>ENABLED</td>
</tr>
</tbody>
</table>

*Figure 3-3 Domino server document - Notes network ports*

### 3.2.2 ID modifications

The steps in this section insure that the Domino server will not pause and display a dialog asking for a password or for authorization to execute code.

The KeyFileName parameter in the Notes.ini file on the server, specifies which ID will be used to run LEI operations. If KeyFileName is not specified, the server ID will be used instead (the ID listed in the ServerKeyFileName INI parameter).

Find this ID file, and do the following steps:

1. Get a copy of the ID file from the Domino server and put it on your Notes client workstation.

2. In the Notes client, use the menu **File -> Security -> Switch ID** to switch to that id.


4. On the User Security window, under the Security Basics tab, as displayed in Figure 3-4, select **Don’t prompt for a password from other Notes-based programs (reduces security)** checkbox.
5. Next click the **What Others Do** tab, and the subtab **Using Workstation**.

6. Click the **Add** button and enter **Enterprise Connector Products/Lotus Notes Companion Products**.

7. For this new entry, select the following options as displayed in Figure 3-5:
   - **Allow access to:**
     - File system
     - External code
     - Current database
     - Environment variables
   - **Allow ability to:**
     - Read other databases
     - Modify other databases
3.3 Installation of LEI

This section describes the detailed process of installing LEI. We begin by providing a helpful checklist to review, to ensure you have proper access to documentation, required databases, and that the proper tasks on the server are running.

3.3.1 Installation Checklist

Before proceeding with installation, make sure you have done the following:

- Review the readme documentation that is available with the LEI product.

Note: For AS/400 systems, make sure the ID file is still owned by QNOTES after you upload it.

3.2.3 Other preparations

Refer to the specific installer documentation for your OS Platform for any additional steps.
Follow pre-install directions for your server platform in the LEI Installation Guide Notes database (leiig.nsf). This database comes with the install kit.

Confirm that your server and workstation satisfy the hardware and software requirements identified in the LEI Installation guide.

iSeries install, please refer to the redbook *Implementing IBM Lotus Enterprise Integrator 6 on the IBM iSeries Server*, SG24-6941, for detailed steps.

Confirm that the appropriate access to the servers, through the Domino server document identified in 3.2.1, “Domino server document changes” on page 38 have been granted.

Confirm that the required modifications to the ID file discussed in 3.2.2, “ID modifications” on page 39 have been applied.

Confirm that the Domino server has been restarted and is running.

Confirm that DECS is not running as a Domino server task.

3.3.2 Process of installation

Details vary based on platform and version, so you should consult the LEI Installation Guide database and follow the detailed steps listed there. Here are some things to note, however:

Selecting installation servers

At some point, you will be prompted for the names of your LEI server and Domino server, as shown in Figure 3-6 (this screen shot is from iSeries; yours might not match exactly). It defaults with the name of the Domino server on which LEI is being loaded.

This is where you would set up LEI Clustering, also called Remote Administration. If you wish to put the LEI Administrator database on a different Domino server than the one on which you are installing LEI, for purposes of centralizing administration tasks, this screen lets you do that. In the first field, enter the name of the Domino server on which to install LEI. In the second field, enter the name of the server which will host the LEI Administrator Notes database. Some LEI functionality is unavailable if you use remote administration; refer to “LEI clustering” on page 53 for details.

Note: The entry in the Domino Server Name field must match what is listed in the Mail Server field located at the bottom of the Basics tab in the Domino server document. This will be corrected in a future release of LEI. For details, refer to the following technotes:


Enabling Advanced Realtime

If you don't intend to use Advanced RealTime activities, you can avoid some overhead by disabling this functionality during installation, in the screen shown by Figure 3-7.

If you need Advanced Realtime later, you can enable it by editing the server's notes.ini file. The ini variable to enable Advanced Realtime is:

```
EXTMGR_ADDINS=decsext
```

If the EXTMGR_ADDINS line already exists, add decsext to the end of the comma-delimited list.
Implementing IBM Lotus Enterprise Integrator 6

Figure 3-7 InstallShield - Enable/Disable RealTime activities

Install documentation/Create script vault options

The screen shown in Figure 3-8 lets you choose whether to install the optional documentation and script vault.

As noted on this screen, unless you tell it to create a script vault, you can't automatically migrate the scripts in your existing script vault.
Chapter 3. Installing and configuring the LEI Administrator

Figure 3-8   InstallShield - Optional install of documentation, samples and script vault

Running the Migration Tool during installation

After all the LEI files have been copied by the installer, the screen shown in Figure 3-9 gives you the option to launch the migration utility. If you choose not to run the migration utility during installation, you may also run it later. Refer to Chapter 4, “Migrating from Lotus Enterprise Integrator 3.x” on page 57 for information on migration.
3.4 Administration of LEI

On the Win32 and UNIX® platforms, there are two ways to run LEI — as a standalone process with its own console window, or as a server add-in task with no separate console window (any LEI-related messages then appear on the Domino server console).

On the iSeries server, there is no standalone option — LEI is installed only as a Domino addin task.

The LEI process has a command language; the commands are listed in Table 3-1. If LEI runs in its own window, you can enter the commands there.

If LEI is loaded as an add-in, you can use the same commands, but you must prefix them with `tell lei`.

Note:
- The Migration tool does not work well when installing a remote administered LEI server (as described in “Selecting installation servers” on page 42). For details, refer to 4.1.3, “LEI clustering and remote administration” on page 59.
- Support recommends the migration process be launched on its own instead of being done as part of the LEI installation process. The LEI log file is unreadable when migration is done as part of the installation process.
Table 3-1  LEI console commands

<table>
<thead>
<tr>
<th>LEI command</th>
<th>Domino server console command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>tell lei help</td>
<td>Displays a brief description of the LEI server commands.</td>
</tr>
<tr>
<td>List</td>
<td>tell lei list</td>
<td>Displays a list of activities that are currently being executed by this LEI server. The number associated with each Activity name must be used with the Close and Kill commands issued from the LEI Administrator.</td>
</tr>
<tr>
<td>Status</td>
<td>tell lei status</td>
<td>Displays basic configuration information about this LEI server, as well as server time and status.</td>
</tr>
<tr>
<td>Quit</td>
<td>tell lei quit</td>
<td>Shuts down the LEI server and all LEI Activities</td>
</tr>
</tbody>
</table>

3.4.1 LEI Administrator database

The console command set described in the previous section, lets you get status and shut down the server. For anything else, you must use the LEI Administrator database, decsadm.nsf, which lets you:

- Define connection information to your external data sources
- Configure Activities that use those connections
- Configure LEI servers controlled by that LEI Administrator
- Activate and deactivate configured activities
- View which servers and activities are operating
- Stop or shutdown an LEI server controlled by that LEI Administrator
- View logs for servers or activities controlled by that LEI Administrator
- Access reference material

The LEI Administrator database might not be on the same Domino server where LEI is running, as explained in “LEI clustering” on page 53.

Figure 3-10 shows the LEI Administrator database used to control the operation of the LEI server. The administrator has three distinct areas of activity:

- Navigator frame
- Working frame
- Active View frame
Navigator frame

The Navigator area, on the left, catalogs the different kinds of information available from the LEI Administrator. Going from top to bottom, you can:

- View configuration documents (connections, activities, and server docs).
  These are views within the LEI Administrator database.

- Organize projects
  If you use Create / Folder from this screen, the folder will show up here. You can use the folders to organize any documents in the LEI administrator into projects — drag and drop or use Actions / Move To Folder. A document may be a member of multiple folders.

- View reference material
  The Help section contains links to the LEI documentation databases and the LEI Administrator database About document.

- User assistance
  The user assistance feature guides you through the creation of Advanced Realtime activities with a wizard interface. It also displays additional Help text on connection documents when you create or edit them.

Experienced users may wish to disable this feature. Turn user assistance on or off by clicking the button labeled Turn User Assistance On or Turn User Assistance Off.
Access log entries

Log entries associated with LEI activities are stored in a separate log database; the “Log” link takes you to that database, where the log entries are sorted by server, date, and activity name.

Tip: The “Current Activity Execution Log” control, shown in Figure 3-10, goes directly to the latest log entry for the highlighted activity.

Working frame

The working frame, shown in Figure 3-12, displays the contents of the view selected in the navigator frame. At the top of the working frame is an action bar that is present regardless of what is displayed in the body of the frame. The action bar provides access to create new connection and activity documents.
Active View frame

The Active View frame, displayed in Figure 3-13, provides the administrator the ability to view all live servers and activities controlled by LEI. Through the action bar the administrator can stop live activities or view the associated log entries.

3.4.2 Configuration

Clicking on Configuration in the Navigation frame lets you configure certain aspects of the LEI server and administrator by accessing both the LEI Server Configuration document and the LEI Administrator Configuration document. Figure 3-14 displays the LEI Server Configuration document. There should be one of these for each server administered through this LEI Administrator database.
Table 3-2 shows the parameters in the LEI Server Configuration document and how the values modifies operation of the server.

Table 3-2  LEI Server Configuration document parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Status</td>
<td>Specifies the status of the server. This information cannot be modified directly. The status is Active or Inactive.</td>
</tr>
<tr>
<td>Last Broadcast</td>
<td>Specifies the date and time that the Administrator last received a status update from the server. Like the Current Status, this information cannot be modified. The frequency of broadcasts is the one entered in the LEI Administrator Configuration document. See Figure 3-15.</td>
</tr>
<tr>
<td>Server Name</td>
<td>Specifies the name of the LEI server given at install time. This information cannot be modified.</td>
</tr>
<tr>
<td>Poll Interval</td>
<td>Specifies the interval at which the LEI server polls the Administrator database to see if there are any activities that it needs to execute.</td>
</tr>
</tbody>
</table>

Note: If Advanced RealTime is being enabled after completion of the installation process by selecting the Advanced RealTime checkbox displayed in Figure 3-14, review the Domino server’s Notes.ini file. Add the entry, `EXTMGR_ADDINS=DECSEXT`. If the `EXTMGR_ADDINS` line already exists, add `DECSEXT` to the comma-delimited list.

Figure 3-14  LEI Server Configuration document
The parameters in the LEI Administrator Configuration document, displayed in Figure 3-15, modify how the LEI Administrator operates. There should be only one of these documents.

### LEI Administrator Configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Number of Activities</td>
<td>Specifies the greatest number of concurrent activities that the server will run. Once the server is running the maximum number of activities, it postpones additional activities until existing activities are concluded. Zero stands for no maximum.</td>
</tr>
<tr>
<td>Maximum Duration of Activities</td>
<td>Specifies the longest duration, in minutes, that the server will run any single activity. The server automatically closes any activity that exceeds this duration. Zero stands for no time out.</td>
</tr>
<tr>
<td>Maximum Consecutive Failures</td>
<td>Specifies the greatest number of consecutive failures that a single activity can have before the server no longer schedules it for execution. Zero stands for no disabling</td>
</tr>
<tr>
<td>Activity Shutdown Request TimeOut</td>
<td>Specifies the amount of time in seconds that an activity has to respond to a close command before it is terminated. Zero stands for no time out.</td>
</tr>
<tr>
<td>Domino Server</td>
<td>Name of the server that hosts Domino. It may be the same as or different from the server that LEI has been installed on.</td>
</tr>
<tr>
<td>Advanced RealTime Enabled</td>
<td>Select this checkbox to notify LEI that the specified Domino server can run Advanced RealTime activities.</td>
</tr>
</tbody>
</table>

Log Database: LEILOG.NSF

LEI Help Database: help/leidoc.nsf

Scripted Agent Database: leivi8.nsf

Status Broadcast Interval: 80 minutes

---

Table 3-3 describes the effect of these settings.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Database</td>
<td>Specifies the file path and name of the LEI log database on the Domino server, relative to the Domino server’s data directory.</td>
</tr>
</tbody>
</table>
LEI clustering
An LEI clustered environment uses one LEI Administrator to control multiple LEI servers. This is distinct from a Domino cluster that keeps replica databases on clustered Domino servers synchronized for load balancing and fail over. The benefit of an LEI cluster is the ability to have a single point of administration for multiple LEI servers.

**Note**: To refer to the administration of servers that are in an LEI clustered environment and their LEI Administrator database is stored on a remote server, the term Remote Administration is often employed. This is to reduce confusion between LEI clustering and Domino clustering.

LEI servers that are remotely administered cannot be used for Advanced RealTime activities. In an LEI clustered environment the only server that is able to operate Advanced RealTime activities is the LEI server that hosts the LEI Administrator database, decsadm.nsf, operating locally. If a remotely administered server has the Advanced RealTime checkbox enabled, displayed in Figure 3-14, all Advanced RealTime activity documents are likely to have difficulty being started.

The only method of configuring an LEI cluster is through the installation procedure. As discussed in “Selecting installation servers” on page 42, the installer displays a window requesting the name of this LEI server and which Domino server to create the LEI Administrator database on. If you want to create a “cluster” of servers with a single LEI administrator, you should first install or upgrade LEI on the administration server. When you install that server, enter the same name for the LEI server and Domino server, as displayed in Figure 3-6 on page 43.

If this is an additional LEI server in an already existing cluster, perform the following steps:

1. Enter the new LEI server name in the field labeled **LEI Server Name**. In the field labeled **Domino Server Name**, enter the name of the Domino server that hosts the LEI Administrator database. See Figure 3-16. Click **Next** to continue.

**Note**: Currently the entry in the Domino Server Name field must matched what is listed in the Mail Server field located at the bottom of the Basics tab in the Domino server document. This will be rectified in a future release of LEI. For details on this current limitation, refer to the following technotes:


<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEI Help Database</td>
<td>Specifies the file path and name of the LEI Help databases on the Domino server, relative to the Domino server's data directory.</td>
</tr>
<tr>
<td>Scripted Agent Database</td>
<td>Specifies the file path and name of the optional repository for Scripted Agents, relative to the Domino server's data directory.</td>
</tr>
<tr>
<td>Status Broadcast Interval</td>
<td>Specifies the interval, in minutes, at which each running LEI server must notify the Administrator database of its status. Any LEI server that fails to update its status in three of these time periods is assumed to have terminated abnormally. The interval value must be entered in minutes and defaults to a value of 60. The status of the server is reported in the LEI Server Configuration document along with the date and time of the last broadcast received. See Figure 3-14.</td>
</tr>
</tbody>
</table>
Implementing IBM Lotus Enterprise Integrator 6

Figure 3-16  InstallShield - LEI server identification and LEI Administrator database location

2. The next window that appears, displayed in Figure 3-17, informs the user that it has detected that there is already an LEI Administrator on the server listed. This is to be expected as we are telling it to add the new LEI server configuration details to an already existing LEI Administrator in the cluster. Click **Next** to continue.

Figure 3-17  InstallShield - Informational message
3. You are now asked if the Domino server is on the machine that we are installing to. Click **No** to continue with the installation process.

![Figure 3-18 InstallShield - Confirmation message](image)

4. With completion of the LEI installation process and the initial starting of the LEI server, the LEI Administrator Active View frame will display multiple operational servers as shown in Figure 3-19 (provided the LEI servers are running).

![Figure 3-19 Active View frame showing an LEI clustered environment](image)

### 3.4.3 Starting and stopping the LEI server

There are a number of ways to start and stop the LEI server. This section reviews how to control the operation of the LEI server through:

- Notes.ini file, providing automatic loading
- The Domino console
- The LEI Administrator

#### Automatic loading

The installation process automatically places LEI in the ServersTasks line of the Domino server’s Notes.ini file on the iSeries platform. If LEI is not to be loaded automatically then the Notes.ini file should be edited and the LEI entry removed from the ServersTasks= line.

Similarly if LEI has been removed and it is desired to load LEI automatically, edit the Domino server’s Notes.ini file and add the LEI entry to the ServersTasks= line.

#### Domino console

To start LEI via the Domino server console enter the following command:

```
load lei
```

To stop the LEI server via the Domino server console enter the following command:

```
tell lei quit
```

#### LEI Administrator database

Through the LEI Administrator there are two options for quitting the selected LEI server. Both are located under the Actions menu, as displayed in Figure 3-20.
Selecting the pull-down menu options **Actions -> LEI Server Administration -> Shutdown Server** stops the selected LEI server after all live activities have successfully shutdown.

Selecting the pull-down menu options **Actions -> LEI Server Administration -> Stop Server** stops the selected LEI server immediately, without waiting for live activities to successfully shutdown.

![Image](image_url)

*Figure 3-20   Lotus Administrator, stopping the LEI server*
This chapter provides an overview of the issue of migration for organizations that are moving from Lotus Enterprise Integrator (LEI) 3.x to LEI 6. There are a number of issues that require addressing with any software migration. In an attempt to review these issues, this chapter discusses the following:

- Considerations to address when migrating to LEI 6
- What pre-migration activities need to be addressed
- The process of migration

The LEI Installation Guide (leiig.nsf), which comes with the installation kit, may contain more up-to-date information than this guide, particularly regarding bug fixes. Consult the chapter on migration in that guide for the latest information.
4.1 Migration considerations

With LEI, migration is often synonymous with upgrading from the a previous release. While they are often talked about together, it is important to separate the tasks of upgrading the LEI server and of migrating activity documents, connection documents and LotusScript “LC LSX” code.

The process of upgrading the LEI server is taken care of by installing the new LEI code, discussed in Chapter 3, “Installing and configuring the LEI Administrator” on page 37. This process is only performed once and replaces the 3.x version of LEI with LEI 6.0.1.

The new LEI Administrator file is decsadm.nsf, the same filename as the DECS Administrator, which it replaces during install. The install process makes a backup copy of the old decsadm.nsf — you specify the backup filename during installation.

Previous LEI versions used a different filename, leiadm.nsf, for their LEI Administrator. If you had an leiadm.nsf from LEI 3.x, it is not removed during installation, which makes it available for the process of migration.

If you install LEI on a server where you’re using any version of DECS, the activities and connections of the DECS activities will automatically be migrated to LEI 6 as part of the install process. Migration of DECS documents happens as part of the installation, not as part of the LEI Migration Utility.

The migration process will copy and update activities and connections from the old leiadm.nsf database. You can run the migration process multiple times, but take care about documents with duplicate names (see page 61), and agents from the old leivlt.nsf database, which will be duplicated if you migrate them a second time (see page 64).

4.1.1 Migrating scripted documents

The process of migration also migrates agents stored in the script vault (leivlt.nsf) to the LEI 6 script vault (leivlt6.nsf). The LEI LSX used in scripts for LEI 3.x was retired and now the standard LC LSX is used in scripted agents. The process of migration will alter the entry Uselsx "*lsxlei" to Uselsx "*lsxlc". This will do it for scripted activity agents; other LotusScript may require special handling. Refer to 9.1.1, “Migrating LEI LSX code from Previous LEI Versions” on page 192.

The migration will only migrate agents that are stored in a script vault named leivlt.nsf. A scripted activity can call a script stored in any Notes database, so if you have some existing scripts stored elsewhere, you will need to edit them yourself to make this change.

The process of migration will not migrate scripts stored in a script library, even if the script library is in leivlt.nsf. These script files must be copied and altered manually.

If you operate the process of migration multiple times, duplication of script names are not checked, so be careful. You may want to delete agents from the old leivlt.nsf after you confirm successful migration.

Important: The LEI 3.x script vault that is being used for the script migration must be named leivlt.nsf.
4.1.2 Migrating metaconnection documents

Metaconnection documents in LEI 3.x did not require that the underlying connection document to have metadata selected. With LEI 6 it is required that the underlying connection document of a meta connector has metadata selected. This new requirement will prevent migrated metaconnection documents from operating properly.

To correct the issue, after migration, open the underlying connection documents of all metaconnection documents and define the metadata. This should not prevent existing activities that use that connector from working.

For a complete explanation of metaconnectors, refer to 6.6, “Metaconnectors” on page 97.

4.1.3 LEI clustering and remote administration

An LEI clustered environment can cause complexities in relation to the process of migration. If the server that is being upgraded is being made part of an LEI cluster where the LEI Administrator database resides on a remote server, it is recommended to complete the installation prior to performing the process of migration outlined in 4.3, “Process of migration” on page 60.

If the LEI Administrator database resides on the same LEI 6 server, the migration process is straightforward and can be undertaken without additional activity. When the LEI 6 Administrator database is on a remote server to the one being migrated, perform the following steps:

1. Manually copy the LEI 6 Administrator database to the machine to where the LEI 6 server has been installed.
2. Invoke the migration tool as described in 4.3, “Process of migration” on page 60.
3. Manually copy the LEI 6 Administrator database back to the server where the LEI 6 Administrator database was originally installed.
4. Open the LEI Administrator and select the pull-down menu options of Actions -> RefreshAllDocuments.

Step 4 is used to recompute all the computed fields from the documents that have been migrated. Before the migrated documents are used for the first time these computed fields are required to be updated.

4.2 Pre-migration activities

Before running the migration tool, you should prepare as described in this section.

Find out Domino server’s data directory
If you run the migration tool yourself (not as part of the installation), on iSeries, you will be asked for the path of the Domino data directory.

If you don’t know it, you can find it in the server’s notes.ini file; look for a line that begins Directory=. The redbook Implementing IBM Lotus Enterprise Integrator 6 on the IBM @server iSeries Server, SG24-6941, shows some alternate techniques.

Backup environment
As always when installing software on a server, it’s smart to make a backup of the environment, to restore from in case the install goes wrong.
The most important files to backup are leiadm.nsf, decsadm.nsf, leivlt.nsf, lei.ini and notes.ini.

**Upgrade Domino**

LEI and Domino releases are now linked. Before you install LEI, you must first upgrade the Domino server to the same point release as the LEI version you plan to install.

**Install LEI 6**

Once the Domino server has been upgraded and is working, the upgrade of the LEI server from LEI 3.x to LEI 6 can be addressed. Chapter 3, “Installing and configuring the LEI Administrator” on page 37 addresses the requirements for installing LEI 6 and the process of installation.

### 4.3 Process of migration

The process of migration can be performed at the time of installation or after installation is complete and LEI is operational.

**Note:** Lotus Support recommends the migration process be launched on its own instead of being done as part of the LEI installation process. The LEI log file is unreadable when migration is done as part of the installation process.

To run the migration outside of install, the Domino server should be running, but the LEI server does not need to be.

For Win32 servers, execute the program file run_migration.exe. For UNIX servers, execute the program file run_migration. You can find this file in the Domino executable directory.

On an iSeries LEI server, you should use the program iSetupMigration.exe, which is on your installation media. The startup on iSeries is a little different; the redbook *Implementing IBM Lotus Enterprise Integrator 6 on the IBM iSeries Server*, SG24-6941, contains detailed screenshots.

When the Migration Utility welcome screen appears, as displayed in Figure 4-1, the operator will have to enter relevant information as to the server performing the migration. The window requests the following information:

- Domino server name of the server performing the migration.
- The file name of the LEI 3.x Administrator database.
- The name of the Domino server that performs the Advanced RealTime activities.
- Is it desired to backup the present LEI 6 Administrator database.
- Is it desired to migrate the ACL entries across from the 3.x LEI Administrator database.

**Note:** Current versions of the migration tool do not work well if the “Source server” and the “Domino server” are not the same (that is, if LEI is administered remotely). See 4.1.3, “LEI clustering and remote administration” on page 59.

Provide the requested information and click **Migrate** to continue.
1. Should a conflict exist between the names of connection documents used in the 3.x LEI Administrator database and the names use in existing connection documents in the LEI 6.x Administrator database, the screen shown in Figure 4-2 is displayed.

The operator can select one of the following options:

- Rename the LEI 3.x connection documents appending the suffix of their choosing to all conflicted documents by entering a value in the Chose Conflict Suffix box.
- Overwrite existing LEI 6.x connection documents with the LEI 3.x versions of the connection documents.
- Skip migrating the connection documents that have a conflict with their names.

The option selected applies to all conflicts; there is no option to modify a subset of the connection documents. Click **Continue Migration**.
2. Should a conflict exist between the names of activity documents used in the LEI 3.x Administrator database and the names used in existing activity documents in the LEI 6.x Administrator database, the operator will be presented with the display shown in Figure 4-3.

The operator can select one of the following options:
- Rename the LEI 3.x activity documents appending the suffix of their choosing to all conflicted documents by entering a value in the Chose Conflict Suffix box.
- Overwrite existing LEI 6.x activity documents with the LEI 3.x versions of the activity documents.
- Skip migrating the activity documents that have a conflict with their names.

The option selected applies to all conflicts, there is no option to modify a subset of the activity documents. Click **Continue Migration**.
Figure 4-3   Migration utility - Conflict with activity document names

Note: If there are no conflicts between names of activity documents, Figure 4-3 will not be displayed.

3. The next window, shown in Figure 4-4, is the progress monitor for the migration utility.

Figure 4-4   Migration utility - Progress Indicator
4. If a scripted activity document was found in the LEI 3.x Administrator database, the window displayed in Figure 4-5 is shown. It will appear for the first agent in the script vault then dependent on the response selected will decide whether it is displayed again.

If Migrate is clicked, it will migrate that agent and prompt again when the next agent is being processed.

If Don’t Migrate is clicked, it will not migrate that agent and then will prompt for the next agent to be processed.

If Migrate All is clicked, the all agents will be processed and the migration utility will not prompt with this window again.

**Note:** If you migrate an agent more than once, the Migration Utility will create duplicate agents.

5. If an agent uses script stored in a script library a window as shown Figure 4-6 is displayed, indicating that agents in script libraries will have to be migrated manually. This is just an informational window indicating that after the migration is completed any scripts located in a script library, as discussed in 4.1.1, “Migrating scripted documents” on page 58, will have to be migrated manually. Click OK to continue.
6. Upon completion, Figure 4-7 is displayed. Click **Finish** to complete the migration.

![Migration Utility](image)

Figure 4-7  Migration utility - Completion window

7. After migration has completed, open the LEI Administrator and use the menu **Actions** -> **RefreshAllDocuments**.

   As mentioned earlier, this is done to recompute all the computed fields from the documents that have been migrated. Before the migrated documents are used for the first time, these computed fields must be updated.

8. Any scripts that are stored in the LEI 3.x script vault should now be manually copied to the new script vault, `leivlt6.nsf`. Do not forget to modify where appropriate.

9. Test all migrated activity documents to verify successful migration.

### 4.4 Troubleshooting

If migration is not operating as expected, review the following areas:

- LEI 6 has been successfully installed and operating as expected.
- Confirm that the LEI Administrator database is on the server being migrated.
- If the LEI 3.x Administrator has been copied using a Notes client, was the process of creating a new copy told to copy the ACL? It is important not to delete ACL entries prior to migration.
Review the Domino server’s log.nsf file, Miscellaneous Events view. Find the log entry corresponding to the time of your installation attempt and see whether there are any helpful messages there.

Review migration log (migration.log). This is discussed in the next section.

4.4.1 Migration log file

To review how the migration process has performed, review the migration log file (migration.log). On Win32 and UNIX platforms, you will find this file in the Domino executable directory. On iSeries, look for it in the following Integrated File System (IFS) path:

/QIBM/ProdData/LOTUS/NOTES

Figure 4-8 shows an example of the migration.log contents. It lists the actions undertaken by the process of migration. For instance if the operator was seeking if there was a difficulty with the transfer of a connection document, they would search for the line with the connection document’s name to see any associated message.

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/30/2003 04:47:40</td>
<td>The LEI administrator has been backed up to decsadm_1.nsf</td>
</tr>
<tr>
<td>05/30/2003 04:47:40</td>
<td>Duplicate ACL entry, -Default-. ACL entry will not be added</td>
</tr>
<tr>
<td>05/30/2003 04:47:40</td>
<td>Duplicate ACL entry, OtherDomainServers. ACL entry will not be added</td>
</tr>
<tr>
<td>05/30/2003 04:47:40</td>
<td>Duplicate ACL entry, CN=Ian Barrack/O=ITSO. ACL entry will not be added</td>
</tr>
<tr>
<td>05/30/2003 04:47:40</td>
<td>Duplicate ACL entry, LocalDomainServers. ACL entry will not be added</td>
</tr>
<tr>
<td>05/30/2003 04:47:40</td>
<td>Duplicate ACL entry, CN=Enterprise Connector Products/O=Lotus Notes Companion Products. ACL entry will not be added</td>
</tr>
<tr>
<td>05/30/2003 04:47:40</td>
<td>Duplicate ACL entry, CN=LEI3DomSvr/O=ITSO. ACL entry will not be added</td>
</tr>
<tr>
<td>05/30/2003 04:47:41</td>
<td>Connection AS06 DB2 Employee has been transferred</td>
</tr>
<tr>
<td>05/30/2003 04:47:41</td>
<td>Connection DB2 Connection has been transferred</td>
</tr>
<tr>
<td>05/30/2003 04:47:41</td>
<td>Connection Employee Profile has been transferred</td>
</tr>
<tr>
<td>05/30/2003 04:47:41</td>
<td>Connection Notes Connection to Casefile.nsf has been transferred</td>
</tr>
<tr>
<td>05/30/2003 04:47:41</td>
<td>Activity Direct Emp has been transferred</td>
</tr>
<tr>
<td>05/30/2003 04:47:42</td>
<td>Activity Direct Transfer of Case Files has been transferred</td>
</tr>
<tr>
<td>05/30/2003 04:47:42</td>
<td>Activity Employee has been transferred</td>
</tr>
<tr>
<td>05/30/2003 04:47:43</td>
<td>Scripts not migrated -- Unable to open LEI 6 Script Vault (leivlt6.nsf)</td>
</tr>
<tr>
<td>05/30/2003 04:47:43</td>
<td>Scripts not migrated, unable to open Source R5 script vault</td>
</tr>
</tbody>
</table>

Figure 4-8  Migration.log file
Chapter 5. Diagnostic tools and techniques

This chapter describes utilities for testing connectivity to relational databases, for testing a connection document in the LEI Administrator, and tools and techniques for debugging activities.

If you can't establish a connection to a relational database, use these tools to narrow down the problem:

- DCTEST shows which Lotus Connectors are installed on your server, lets you choose a database type, and tests whether it's possible to communicate with that database using the database client's API.

- CONTEST operates one level higher than DCTEST. Rather than calling the database APIs directly, CONTEST uses the appropriate Lotus Connector and an LEI Connection Document.

Before you go to a great deal of trouble testing the connection, however, look for error messages that the server displays on the console or logs in the LEI Activity Log. You may be able to rule out a connection problem based on that.

For example, if your activity is failing with the error “Invalid Metadata” (meaning the table you're trying to use does not exist or you have no access to it), then there's no need to test the connection. Without a connection, you couldn't get far enough to determine that the table doesn't exist. Refer to 5.4, “Activity debugging techniques” on page 71 for some activity debugging tips.

Note: If you have connection problems using DB2 UDB on iSeries, consult the redbook *Implementing IBM Lotus Enterprise Integrator 6 on the IBM @server iSeries Server*, SG24-6941, which includes a thorough guide to configuring DB2 connectivity.
5.1 Test Connectivity using database vendor tools

If you're having trouble connecting to an outside data source, it's best to start tracking down the problem at the lowest level you can. Whatever client software you're using to connect to the data source will typically have the ability to create a test connection to a database of that type. For instance, using DB2 Connect client on Windows 2000, you can use the Client Configuration Assistant to check whether it's possible to establish a connection.

![Configuration Assistant](image)

*Figure 5-1  Using the DB2 Configuration Assistant to test a connection*

If you have access to a tool from the database vendor, use that first. Once you can connect that way, move on to the next section.

5.2 Test connectivity using the DCTEST utility

Once you've used the vendor tools to make sure the database you're trying to connect to is functional (or if you have no access to such a tool), the next step is to use DCTEST to test the native APIs that the Lotus Connectors use to access that type of database.

DCTEST ensures that proper connectivity has been configured to access a specific data source. If the vendor's connection test works and DCTEST fails, the likeliest explanation is that the database client API is a version that's incompatible with LEI's database code.

**Tip:** LEI includes separate connectors for Oracle (that is, Oracle 7) and Oracle 8. Make sure you select the right one!

The DCTEST utility is installed in the Domino program directory. Set that as your working directory, then to run DCTEST:
On Windows 32, type `ndctest`
On UNIX, type `dctest`
On iSeries, call `qnotes/dctest`

This will display a screen such as the one shown in Figure 5-2.

```
C:\Lotus\Domino> ndctest

Lotus Connector Server Connection Verification Test

Copyright 2002 IBM Corporation
--------------------------------------------
This utility will verify connectivity from this
machine to the selected type of server.

At the prompt, enter the number of the test
you would like to run, or enter 0 to exit.

0 - Exit this program
1 - Lotus Notes
2 - Oracle Server
3 - ODBC
4 - Sybase Server
6 - DB/2
7 - OLE DB
8 - Oracle8 Server

Run test number: [0]
```

*Figure 5-2  The DCTEST utility*

From this point, you can select the database type you want to test, and enter the connection
details. DCTEST will try to connect and disconnect, just to prove it can be done. If any errors
occur, they will be displayed; you may need to consult the vendor's documentation to interpret
the message. Figure 5-3 shows a successful test.
5.3 Test the Connection Document using CONTEST

Once you’ve used DCTEST to establish that it’s possible to connect to your target database, use CONTEST to check that a connection document in the LEI Administrator database is correctly defined.

The CONTEST is in the Domino program directory. Open a command line window, and set that as your working directory. To invoke:

- On Windows 32, type `ncontest "connection name"`
- On UNIX, type `contest "connection name"`
- On iSeries, `CALL PGM(QNOTES/CONTEST) PARM('connection name')`

In place of `connection name`, type the name of a connection document from the LEI Administrator database.

You will not get detailed error information from this test; just a pass/fail indication, as shown in Figure 5-4. You can use the “-p” option to get a list of connector properties if the connection succeeds, but this doesn’t help in diagnosing a failure.
There’s another way to test a connection document which gives a more useful error message. Open the document in the LEI Administrator, place it into edit mode (or create a new connection document), enter the connection information you think is correct, then click the button beside the Owner or Name field. This tries to connect to the database to get a list of owners or table names for you to select from. If that fails, you get an error message such as is shown in Figure 5-5.

This error message is specific to the database system you’re trying to use, and you may need to consult the vendor documentation for an explanation. In this case the problem is fairly obvious — we got the username or password wrong.

5.4 Activity debugging techniques

Once you’ve established that the connection is working, here are some tips for figuring out why an activity might be failing.

- Check the activity log. Highlight the activity in the LEI Administrator, and click the Current Activity Execution Log button. Any error messages or warnings you may find there may be a good clue.
- If the failure is intermittent, you might try checking past activity logs for the same activity. Click the Logs link, near the bottom left corner of the LEI Administrator main window, to open a view of log entries that you can search.
- Look for error messages on the LEI Console (if running LEI standalone).
- Look for error messages on the Domino Console and in the log.nsf database, Miscellaneous Events view.
- The Connection forms for relational databases contain an option, “Output SQL Statements to Log”. Check this option — you may have to stop and restart the activity for it to take effect. Every SQL statement executed will be written out to the activity log in the leilog.nsf database. Often, seeing the SQL it was trying to execute when it got the error may give you a hint.
In situations where the “Output SQL Statements” option doesn't work or doesn't provide enough information, you might try a Trace metaconnector. Refer to 6.6.4, “Trace MetaConnector” on page 103. You will have to stop the activity and temporarily reprogram it to use the metaconnector instead of the connector it was using.

**Note:** SQL statements generated by the Lotus Connectors may contain placeholders instead of actual data — the data are passed to the database client separately to be plugged into the SQL. For instance, `select * from dbo.empkeys where empid = %1` might seem to be an error, because %1 is not a valid numeric employee ID, but actually this is OK. Different database systems may use different codes to represent their placeholders.
Connection documents

A connection document is a Notes document stored in the LEI Administrator, which contains the information necessary for making a connection to a particular database and (optionally) table, view or stored procedure. This includes the database name, userid, password, and so on. Different connection types may require different information; for instance, the Sybase connection form asks for a SQL server name, but DB2 doesn’t require a server name.

Not all connection documents that are available in the LEI Administrator database are supported by LEI on the iSeries server. The design of the LEI Administrator is the same on all platforms, however, so you can create a connection document for any database type, even if you’re on iSeries or if you haven’t installed the client software for that database.

A special class of connection documents, the Metaconnection, can be combined with a regular connection document to provide enhanced functionality to sort data, meter usage, debug SQL statements generated by LEI, and deal with multivalued field data.

Supported connection document types are:

- DB2
- Oracle (that is, Oracle 7)
- Oracle 8
- Sybase
- ODBC
- OLE DB
- Notes
- File system (allows access to folder contents)
- Text (read or write records in a text file)
- Metaconnectors
  - Collapse/Expand
  - Meter
  - Order
  - Trace
6.1 DB2 connection documents

The different types of connection documents to relational databases are discussed in detail, field by field, in the Domino Designer Help and the LEI Connectors and Connectivity Guide (same information in both places). Space does not permit covering each one here in such detail, but by way of example, the DB2 connection document is presented here. While there are minor differences among the different RDB connection forms, overall they are very similar.

To create a connection document, open the LEI Administrator database (decsadm.nsf) on the LEI server. Click on the Add Connection button and select the database type, in this case DB2. See Figure 6-1.

![Figure 6-1 Creating a DB2 connection document](image)

In the DB2 connection document there are three sections that can be filled in. These are:

- Connectivity - which includes the connection parameters for the database and any options that you can configure for that database type, for instance whether you want to commit all transactions immediately.
- Table selection - This is where you define the exact library/owner, table, and columns the connection will access. This table selection can be a table, a logical view or a stored procedure.
- Other - Where you can categorize the connection document in the LEI Administrator database, and add descriptive comments about the connection.

These sections and the fields in the sections are described in detail below.

6.1.1 Connectivity section

The Connectivity section of the connection document is where you to enter the data needed to connect to the database. See Figure 6-2 for an example.

The fields in this section may vary considerably depending on the database type. You will always specify a database, username and password; apart from that, consult the LEI Connectors and Connectivity Guide (lccon6.nsf) for a detailed description of the options for the connection type you’re creating.

For DB2, this section includes:

- Name: any name you want to give to your connection document — must be unique.
Database: the name of the database that you want to connect to. For DB2, you would generally use the Configuration Assistant of the DB2 Connect client to set up a database name locally, providing the IP address of the server, and so on. Then you can use that name here.

User Name: a username that DB2 will recognize, which activities will use to access the database. This user must have enough authority to do whatever you need this connection for.

Password: the password for the username you entered. In LEI 6.0.2 and later, the password will not be displayed. However, in any version, anyone with access to this document can see the password using the Document Properties dialog.

If you want to prevent other administrators of the LEI Administration database from seeing the password, you can encrypt it using the “key” button. You must first establish an encryption key using the Notes client, and share it with other administrators who are authorized to use this password. For more information on creating secret encryption keys see “Creating secret encryption keys” in the Domino Designer Help.

Read access to connection documents and activity documents can be controlled using database ACL roles and the Security tab of the document properties dialog.

Note: If you encrypt passwords, the Notes userid used to run LEI must have a copy of the key you used for the encryption. If you put reader controls on the documents, this userid must have reader access; otherwise, it can’t read the information it needs to create the connection.

Data Journalling: You must tell LEI whether the DB2 files that will be accessed by this connection document are journalled. You should check with the database administrator to see if the files are journalled and set this parameter accordingly.

Note: The Data Journalling radio button does not turn journalling on or off in the DB2 tables. The selection in the connection document must match this property of the tables that you will access.
Under the Transaction Options tab (Figure 6-3) there are two settings that define how database updates and inserts are handled. The two fields are:

- **RollBack**: The RollBack upon error check box tells LEI that if any error occurs when inserting or updating data in DB2, the state of the database should be put back to what it was after the last commit was done. For example, if you are inserting 20 rows into a table and you do a database commit after every 5 inserts, and on the 7th insert you encounter an error. The database after disconnection will be put back to the state it was in after the 5th insert and the commit.

- **Commit**: This field allows you to configure when commits to the database will be done. There are three options:
  - **Commit at Disconnect**: This option is to commit only just before you disconnect. This means that every row of data would be inserted or updated and only after all rows are complete then a commit is done. The danger in using this setting is that if an error is encountered then the database would be rolled back to before the activity was started and good data would not have been added to the database. You would have to fix the error and re-start the activity from the beginning. For large data transfers this setting may become problematic.
  
  - **Commit Every N Operations**: This option allows you to specify the number of transactions after which you want to commit. The setting of this field would be dependant on how many rows of data are projected to be inserted or updated.

  - **Commit Every Operation**: This option is to commit after every operation. This would commit changes to the database after every insert or update. Using this setting may have a negative impact on performance if a large number of records are being put into the database (for example, a direct transfer activity).
Use the Table Creation Options tab (Figure 6-4) to make LEI automatically create a target table for you. For example, if you want to transfer a set of data from a Notes application to a new DB2 table, you can allow LEI to create the target table for you. The field names and data types in the new relational table will be based on the field names and types of the Notes fields. There is one option in the Table Creation Options tab:

- **Size Cutoff for Create NOT LOGGED COMPACT**: This setting allows you to enter a size cutoff for large object data types (LOB), where if a column exceeds this size it will be created with the NOT LOGGED COMPACT option.

**Note**: When LEI creates tables it will use the most generic data mapping. These mappings may not be the most efficient. For example, LEI will map all Notes text fields to CLOB in DB2 UDB. All number fields in Notes will be mapped to floats in DB2 UDB, and all binary fields will be mapped to BLOBS. Creating the DB2 tables manually may be more efficient.

The Logging Options tab (Figure 6-5) lets you choose whether the SQL statements that LEI send to DB2 are copied into the LEI activity log. This setting is very useful when trying to debug errors. Refer to 5.4, “Activity debugging techniques” on page 71. All relational connectors have this option.

The Map Nulls Option tab (Figure 6-6) allows for fields which resolve to a null value to be mapped to a default value.

---

**Note**: The Commit and Rollback options do not apply to Advanced Realtime activities. Since these activities use a persistent connection, and must maintain a consistent database state among multiple tables, the activity will decide for itself when to commit. These options are applicable to batch activities.
6.1.2 Table selection section

The Table Selection section (Figure 6-7) lets you specify what metadata to use in the selected database. This section is the same for all relational databases. There are three fields:

- **Selection Type**: lets you choose whether you’re connecting to a table, view, or stored procedure.
- **Owner**: This is the name of the library, owner, schema (choose your term based on what database type you’re using) that you want to access.
- **Name**: This is the table, view, file name that you want to access.

The Column(s) field is automatically filled.

Entering information in these fields is optional. It’s only for your convenience to not have to specify the table name everywhere you use the connection — the name you specify here will be filled in automatically. Also, as shown here, you may specify the table name and leave the Owner with a value of <Any>. It’s up to you whether you want to be able to select from all the tables in the data source, or would prefer to see only a single library.

6.1.3 Other section

The Other section (Figure 6-8) lets you enter a category that you would like the connection document to be listed under in the Activities / By Category view. You may also enter a comment for your own reference and for the other administrators. The fields in this section include:

- **Category(s)**: Category where you want the DB2 connection document to appear in the LEI Administrator By Category view. To create a new category, just type it here.
- **Comments**: Comments describing the purpose and relevant details about the DB2 connection document.
6.2 Test your connection

If you can click the Owner or Table button in a relational database connection document, and browse a list of owners or table names, then your connection document is able to connect to your data source. This is a quick test to see whether the connection is usable. Refer to Chapter 5, “Diagnostic tools and techniques” on page 67 for other connection tests and problem diagnosis steps.

Of course, having a working connection is no guarantee that the options in the Connectivity section are all set appropriately for your application. That depends on how you plan to use the connection in an activity. It’s not unusual to create more than one connection document to the same table, so that you can use different options in different circumstances. We suggest you establish a naming convention for connection documents that will let your administrators select the right connection document by name.

6.3 Notes connection document

You need a Notes connection document to run “batch” activities that involve a Notes database. For instance, a one-time Direct Transfer of records from a DB2 table into Notes would require a Notes connection document and a DB2 connection document. Advanced Realtime activities don’t need a Notes connection, because these activities access the Notes database directly, through the Notes API, not by way of the Notes Connector.

Before creating the connection document, you should confirm that all the base connectivity to the local or remote Domino server is setup by running the DCTEST program as shown in 5.2, “Test connectivity using the DCTEST utility” on page 68.

To create a Notes connection document, open the LEI Administrator Database (decsadm.nsf) on the LEI server. Click on the Add Connection button and click Notes. See Figure 6-9.
In the Notes connection document there are two sections that can be filled in. These are:

- **Connectivity**: This section includes the connection parameters for the Domino server and database as well as General options, Document Selection, Field Selection, Data Transformation, and Data Creation.

- **Other**: This section allows you to categorize your connection documents in the LEI Administrator database, as well as add descriptive comments about the connection.

These sections and the fields in the sections are described in detail below.

### 6.3.1 Connectivity section

The Connectivity section (Figure 6-10) of the Notes connection document allows you to enter all data required to connect to a Notes database. The fields in this section include:

- **Name**: any unique name you want to give to your connection document.

- **Domino Server**: any Domino server which is accessible over the network, and to which the LEI userid has access. Usually, it will be the LEI server. You can use either the fully qualified name or the "common name" (provided the common name is unique).

- **Notes Database**: the filepath of the Notes database you want to connect to. This filepath is relative to the Notes data directory on the selected server. Once the Domino server name is entered, you can use the pull-down arrow to see all Notes databases on that server.

**Note**: Make sure the user ID you use to log into Notes and access the LEI Administrator has at least reader access to the Notes database you want to connect to, otherwise you may not be able to access other contents of the database such as available forms.
6.3.2 General Options section

When you select the General Options tab, the following options appear as shown in Figure 6-11. These options can apply to the source or destination database, depending on whether the connection is used by the LEI activity as the source or target connection.

➤ Binding Options control whether and how the Notes form is used when writing documents:

– Do Not Enforce Form Design: Specifies that the LEI activity not execute any Notes form and field-related formulas. Computed fields, such as the current date, are not computed, and fields that are set as Readers or Authors on the form, will not have their usual effect as Readers and Authors fields. This is the most efficient option.

– Enforce Field Flags Only: Specifies that the LEI activity set all field flags as defined in the form but not execute any formulas. As a result, special types, such as reader and author fields, are assigned in new documents as indicated in the form. This option avoids the overhead of computing field formulas.

– Enforce Form Design: Apply the field flags, and in addition, calculate formulas for default value, input translation, and computed field, as part of the LEI activity. This option may cause slower data transfer as a result of formula calculations. This option does not execute @function names that begin with Db. It does enable field flags.

– Enforce Form Design and Perform Validation: Calculates all Notes field-related formulas (default value, input translation, and computed field) and invokes all Notes field validation formulas which ensure that the data is within the parameters for the field in the Notes form. Any field validation errors are recorded in the LEI activity log, and will prevent that Notes document from being written.

➤ Server Port: Specifies the name of the port to use when dialing a server remotely, for example, COM1.

➤ File Path for File Attachments: Specifies the directory where file attachments are stored if the option “Extract File Attachments” is selected. If no path is specified here, file attachments are copied to the current directory. If you use Extract File Attachments, we
strongly recommend specifying a directory here that does not contain any valuable files (otherwise they may be overwritten by attachment files of the same name).

- Use Computed Subforms: Allows you to include all computed subforms by name (non-computed subforms are included automatically). The fields in the computed subforms listed here will be accessible in the field mapping dialog box from a LEI activity using this connection document.

You can “cheat” and use this field even when you don’t have computed subforms, to allow access to fields that exist in the document but aren’t on the main form (for example, they were created by an agent). Create a subform that contains definitions for the extra fields, and enter its name here.

You can also use this technique to define fields that don’t already exist in the Notes document, but will be created on the fly by the Data Transformation formulas (page 86).

Note: This field is disabled if you Enforce Form Design.

![Figure 6-11 Notes connection document, General Options tab](image)

### 6.3.3 Document Selection section

The Document Selection section (Figure 6-12) is used to configure the document level options used by the LEI activity. These are used to limit the documents that the activity selects from the source. Several options can be combined, but the filtering is cumulative. A document must satisfy all options to be selected. These options can apply to the source or destination database, depending on whether the connection is used by the LEI activity as the source or target connection, and applied to the creation of a result set.

- Form To Use: Enables you to select a form from the database specified in the Notes Database field in the Connectivity section.

- Fields To Use From Form: Once you select a form, the fields contained on that form are displayed. Check the field names that you wish to use. Leave blank to select all fields.

Note: Any field selection here is ignored when this connection is used in a Replication activity. To tailor field selection for replication, explicitly map the fields in the Mapping section of the Replication activity document.

- Document Options:
  - Include View Responses: Add to the result set any documents that are responses to the other selected documents; this is similar to adding “| @AllDescendants” to a Notes view selection formula. This option cannot be used along with the Copy Response Hierarchies option. When used in a Notes to Notes transfer, the result in the destination database is not hierarchical and all documents are copied at the same level.
– **Copy Response Hierarchies (Notes to Notes only):** This option ensures that response documents retain their hierarchical relations to the parent document. This option cannot be used with the Include View Responses option. To use this option from an LEI activity with manual field mapping, add a field LCXHIER to both field lists (this is not necessary for automatic field mapping by name or position).

– **Include Documents of Other Forms:** By default, the result set contains only documents associated with the source form identified as the metadata in the activity document (which might not be the same as Form to Use on this screen). Use this option to include documents of all forms in the result set, filtering them through the source form. That means that you can only use the fields on those documents whose names match the source form.

**Note:** If the Fetch View Column Data option is used for selection, this option is required and will be automatically selected.

– **Case-Insensitive View Searches:** Makes view searches case-insensitive (they are case-sensitive by default). An example of where to use this is in timestamp replication with the case-insensitive string comparison option, where case-insensitive searches are needed in Notes.

– **Full Text Search Query to Execute:** The full text query that you enter here is executed against the current result set and only the resulting documents (those that are contained the searched text) are used by the activity. The query is executed on the result set produced by either the command statement in the activity or the View to Use for Selection entry, plus any defined agent to execute.

**Note:** For a description of the Notes full-text query syntax, see the document “Refining a search query using operators” in the Notes Client Help (not the Domino Designer Help), or IBM technote 7003210 at:


The database selected in the connectivity section must be full-text indexed, and a view must be selected in View to Use for Selection option (located under the Field Selection tab) to support this option. Note the difference between a view index and a full-text index.

– A view index exists automatically whenever you have a view. It contains just the data displayed in the view. You can search it only by its sorted columns, and only by their exact values or the characters at the beginning of the column. You can scan through the indexed documents in view order.

– A full-text index applies to the entire database. It doesn’t exist unless you create it. It contains information about every field in every document, including rich text fields. You can search for terms that appear anywhere in any field, or you can restrict the search to specific fields.

This option only affects document selection through an explicit Notes formula, such as a direct transfer activity or the LC LSX Connection.Execute method. It does not affect generic selection such as those done by replication activities.
6.3.4 Field Selection section

The Field Selection section (Figure 6-13) is used to limit the document fields (or view columns) used by the LEI activity to create the result set.

- View to Use For Selection: If the name of a Notes view is entered, the documents appearing in that view will be selected by the activity. This option replaces and ignores any command statement (that is, selection formula) specified in the activity document. By default, only top-level documents are included, although other options allow the view contents to be altered. Documents not in the view are ignored.

When the connection is used for replication activities, this field indicates the name of a view that contains the documents in the right order for replication; it must match the sorting of the other connection that you’re replicating with. Supplying a view name makes your replication activities run much faster. If you don’t specify a view here, LEI will create a temporary view every time the activity runs, and will write a message to the activity log suggesting use of a permanent view name.

When the view is the wrong format or doesn’t exist, checking the Allow View Creation/Modification option (see below) lets the activity create or overwrite that view in its own format. This option overwrites the view when modifying an existing view.

- Allow View Creation/Overwrite: Checking this option allows the activity to modify or create the view if it doesn’t already exist or does not have the proper formatting. This option is required for timestamp replication when a view is specified.

If the Case-Insensitive View Searches option (located under the Document Selection tab) is selected, then the view columns will be set to sort case-insensitive.

**Note:** When you use a permanent view to support keyed searches, such as those used by replication activities, and an existing view needs to be altered (as is always true for timestamp replication), that view will be deleted and a new view will be created.

- Agent to Execute: This field enables you to choose an agent name from a resultant list. The agent specified will further refine the documents selected. The agent is executed on the result set produced by the command (or selection) statement in either a direct transfer or command activity.
Normally you would use a formula agent with a SELECT statement that selects the documents you want to appear in the result set. The product documentation states that you can use a LotusScript agent, but it's not clear how the LotusScript agent selects which documents are in the result set. With either agent type, the agent can also apply a full-text selection.

Note: This option is only valid if the Notes connection document will be used in a direct transfer or command activity.

- Special Fields gives you the option to add document header information to the field list returned by a Select (otherwise, it just contains the fields that appear on the form):
  - Load Document Universal ID (UNID): Appends a virtual field named UNID to the result set. This field will fetch the Universal Note ID for each document. This is needed for Virtual Fields activity using UNID as key (see 8.1, “Virtual fields activity” on page 152). The UNID is returned as a binary value.
  - Load Document Notes ID (NOTEID): Appends a virtual field named NOTEID to the result set. This field will fetch the Note ID for each document.
  - Load Parent Universal ID (REF): Appends a virtual field named REF to the result set. This field will fetch each document's parent Universal Note ID, which is used by Notes to maintain response hierarchies.
  - Load Last Modified Timestamp (@Modified): Appends a field called @Modified to the result set, which contains the Notes implicit document timestamp. Select this option on the relevant Notes connection and specify it as a timestamp in the replication activity to use Notes implicit timestamps in replication. This prevents having to use additional Notes timestamp fields.
  - Extract File Attachments (FILE): Appends a virtual field named FILE to the result set. FILE contains file names of all attachments in a document and extracts files to disk. The files are extracted to the directory specified in the File Path for File Attachments field (located under the General Options tab). If the File Path field is blank, they are extracted to the “current directory” that LEI is running in. Extracting to the current directory is not recommended, since it may overwrite other files with the same name. When transferring file attachments from one Notes database to another, do not use this option; the Copy File Attachments option (see below) is more efficient for this purpose because it does not write to files.

- Fetch Selections: Specifies how fetching is to be performed. Choose one of the following options.
  - Fetch Document Items: Gets the field names from the form designated as source metadata.
  - Fetch View Column Data: Gets data from the view designated at View to Use for Selection option. The fields are automatically named column1, column2, and so on. The destination metadata must contain fields of this name, or use map by position or user defined mapping to prevent field mapping failures. Selecting this option automatically checks the Include Documents of Other Forms option (located under the Document Selection tab).

- Other Options:
  - Translate Multivalue Types to Text: Transfers the contents of multivalue types of Text List, Number List, and Datetime List as text. The values will be concatenated together into a semicolon-delimited string. This is useful when the destination database cannot accept a multivalue’s default binary type.
- Copy File Attachments (Notes to Notes only): Copies any file attachments associated with transferred documents; regardless of whether the fields are mapped or not. If this option is not selected, transferred file attachments will appear as inoperable icons in the target rich text. To use this option from an activity with manual field mapping, add a field called LCXFILE to both field lists (this is not necessary for automatic field mapping by name or position).

- Copy Special Composite Fields (Notes to Notes only): Copies any composite support fields associated with transferred documents. This includes the fields, links, and fonts to return full fidelity of composed data. To use this option from an activity with manual field mapping, add a field called LCXCOMP to both field lists (this is not necessary for automatic field mapping by name or position).

- Maximum Length for Text Data: Signals LEI to use this length as the maximum for text type fields. This option is often used when a metadata is being created in a relational database and text lengths and types are important. The default text length is slightly under 64KB. If the Truncate Data When Necessary option is set in the LEI activity document, LEI truncates the text field to the maximum length indicated here before transfer. If the Truncate Data When Necessary option is not checked, any record with one or more fields greater than this limit is not transferred and LEI logs a Data overflow error.

**Note:** Notes documents have a limit of 32KB of “summary” data (which normally means, non-rich text data). Therefore, unless you map a Notes rich text field to a text column in the other connector, you will not normally exceed the default limit.

<table>
<thead>
<tr>
<th>General Options</th>
<th>Document Selection</th>
<th>Field Selection</th>
<th>Data Transformation</th>
<th>Data Creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>View to Use for Selection:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agent to Execute:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Special Fields Option:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fetch Selections:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Options:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 6-13 Notes connection document, Field Selection tab](image)

### 6.3.5 Data Transformation section

When you select the Data Transformation tab, the options seen in Figure 6-14 appear. The Data Transformation tab provides options for executing formulas to transform data.

- **Formula to Execute During Select:** Enter the Notes formula that will execute just after the data is fetched. The data is altered as part of the fetch/select operation. This applies when the Notes connector is used to connect to the source database.

- **Formula to Execute Prior to Insert:** Enter the Notes formula that will execute just before the selected data is inserted into the target. The data is changed as part of the insert operation.
operation. This applies when the Notes connector is used to connect to the target
database.

- Formula to Execute Prior to Update: Enter the Notes formula that will execute before the
data in the designated target is updated. The transformation will take place as part of the
update operation. This applies when the Notes connector is used to connect to the target
database.

**Note:** Transformation formulas must contain a SELECT statement. Normally you would
use `select @all;`, for instance:

```
select @all; FIELD NewField := "New Value";
```

However, you can use SELECT with a condition to choose which documents will be
affected by the FIELD statements.

### 6.3.6 Data Creation section

When you select the Data Creation tab, the options shown in Figure 6-15 appear. The Data
Creation tab provides options for managing data creation with the Notes database during
processing of any activities associated with the Notes connection document.

- **Notification Options:**
  - **Send New Documents as Mail:** Sends inserted documents as mail rather than saving
to a Notes database. The transferred records must contain a field named SendTo
containing a valid address. For sources that support SQL, this field can be selected as
part of the SQL SELECT clause in the statement of the activity document. For
example:

  ```
  select column1, column2 from table where SendTo="name@email.com"
  ```

  - **Embed Form in Mailed Documents:** Embeds the form in documents sent as a result of
  using the Send New Documents as Mail option. Use this option when the recipients of
  the mail will not have the relevant form in their mail database.

  - **Commit Changes Immediately:** This option commits each change to the Notes
database immediately. The default behavior is to commit all changes at database
  close. This option can slow down performance. This option only applies to Batch
  operations.

  - **Refresh Views as Database is Closed:** Reflects the database changes in the
  databases views at the end of the transfer or replication process when the destination
database is being closed. This option causes view refresh to occur as part of the
  transfer rather than to occur when the user next opens the view.

  Using “Refresh Views” does save time for the next user to open the view, particularly if
large numbers of records have been created or updated in Notes. However, databases
with large numbers of documents and many views, may tie up the server for too long
doing this. Also, by building indexes for views that might not be in use, it can make the Notes database file size larger than it needs to be. An alternative is to write a LotusScript agent that uses GetView to refresh only selected views, call the agent from a Scripted Activity, and make that a dependent activity to the activity that updates the records.

– Purge Deletion Stubs as Database is Closed: Use this option to clear all deletion stubs from the Notes database when disconnecting.

**Note:** We strongly recommend that you use this option only with the Refresh Views as Database is Closed option, since once the stubs are deleted, they will not be properly removed from the views. Selecting the Refresh Views as Database is Closed option will remove them before purging. This option cannot be used on a Notes database with the Notes replication setting to Delete documents not modified in the last N days. Doing so will skip the purging and generate an error indicating the conflict.

To use this option successfully, the Notes id specified must have sufficient access to the database to enable modification of Notes replication settings.

▲ Encryption: Specifies how encryption should occur.

– Encrypt Enabled Fields: Encrypts all enabled fields in the form of the target document.

– Private Encryption Key(s): To use private encryption keys (the default is to use the public key), provide one or more private encryption key names, separated by commas or semicolons in this field. These keys must be available on the LEI server.

▲ Deletion: Delete Database upon Connection Deletes the database specified in the activity upon connection. This is used together with Creation option below to delete and then recreate the database prior to population.

▲ Creation: Create Database (if it doesn't exist), creates a destination Notes database to receive the data if a database does not exist. For database creation, you can optionally enter a template server and template file path to provide the design of the Notes database.

This option should not be selected when the target Domino server is part of a Domino cluster because the database may be created in an inconsistent state. When the target is a Domino cluster, create the database using a Notes client prior to running the LEI activity.

– Database Template Server: The Domino server containing the template file to be used to create the destination database.

– Database Template File Name: The file path of the Notes.ntf template file to be used to create the destination database.

**Note:** Repeatedly deleting a Notes database and creating a new one at the same path has certain drawbacks:

▲ Databases are identified by a replica ID and database ID. The new database will have new IDs.

▲ Since the replica ID is used to identify replica databases on another Domino server, the new database will not replicate with copies on other servers.

▲ The same applies to users’ local replicas.

▲ Notes doclinks and Domino URL links also use the replica ID, and will no longer work with the new database.
6.3.7 Other section

Refer to 6.1.3, “Other section” on page 78.

6.4 File connection document

LEI on the iSeries server supports File connection documents. This type of connection document allows you to connect to a subdirectory on the iSeries integrated file system (IFS). File connection documents are sometimes used with direct transfer or archive activities to move files from one file system to another.

**Note:** The File connection documents can not be used with virtual documents or virtual agent activities.

An example of where File connection documents can be used is if you want to run an activity when a file is created on a file server. In this case you can use a polling activity with the file connection document. Whenever a new file is placed in a particular location in a file system the polling activity triggers a dependant activity.

To create a File connection document, open the LEI Administrator database (decsadm.nsf) on the LEI server. Click on the Add Connection button and click File. See Figure 6-16.
In the File connection document there are three sections that can be filled in. These sections are:

- **Connectivity**: Establishes the basic connectivity for the File connection document.
- **Subdirectory**: Contains the selection field definitions.
- **Other**: Allows you to categorize your connection documents in the LEI Administrator database, as well as add descriptive comments about the connection.

These sections and the fields in the sections are described in detail below.

### 6.4.1 Connectivity section

This section contains information on how to connect to the parent directory of the sub-directory you want to connect to.

- **Name**: This is any name you want to give to your File connection document.
- **Directory Path**: Enter the directory path containing the subdirectory to be used as metadata. This path must indicate the parent directory of the subdirectory containing the needed file(s), not the directory in which the file(s) reside. The specified path must be accessible from the server where the LEI Administrator is installed.

For iSeries you can enter a forward slash (/) which will allow you to access the root directory or a period (.) which allows you to access the current directory. The current directory for LEI will typically be the data directory of the Domino server.

- **File Content Options**: This allows you specify if the contents of the files that will be accessed are Binary (that is, BLOB) or text.
- **Sort Order**: Specifies one of the following three file sorting methods:
  - **Sort Filenames as Binary**: This option does not consider special properties of text. This option gives the best performance.
  - **Sort Filenames with Case Sensitivity**: ‘A’ is different ‘a’.
  - **Sort Filenames without Case Sensitivity**: ‘A’ is not different then ‘a’.
6.4.2 Subdirectory Selection section

This section contains information on how to connect to the subdirectory that contains the files.

- **Name**: Specifies the name of the subdirectory name to search in. Once specified, all associated file attributes are displayed.

- **File Attributes**: Lists the file attributes and corresponding data types of the subdirectories listed.

6.4.3 Other section

Refer to 6.1.3, “Other section” on page 78.

6.5 Text connection document

The Text connection document allows data transfer between text files and Lotus Connectors. You define formats for the input (source) and output (destination) data in a file, called the ZID file. The ZID file is referenced by file name from the Text connection form, or you can specify the actual ZID file contents entirely within the Text connection form. You can also select various options for processing the source and destination data and can select internationalization of days of the week and months of the year.

The Text connection document does not support any browsing. It does not support selection criteria, keyed operations, or conditions. As a result, it cannot be used with the archive activity or with the Advanced RealTime activities (virtual fields, virtual documents and virtual agents).
The Text connection document supports a limited select operation with minimal properties and no keys. This allows it to be used in combination with the order metaconnector as the source of a one-way non-timestamp replication activity. In any other configuration, it does not support the replication activity.

The command statement has no effect; all records are retrieved for relevant activities, including direct transfer and polling.

To create a Text connection document, open the LEI Administration Database (decsadm.nsf) on the LEI server. Click on the Add Connection button and then click Text. See Figure 6-19.

In the Text connection document there are three sections that can be filled in. These sections are:

- **Connectivity**: Enables you to specify the full path of the source or destination text file and whether the connection is a source or a target.
- **Text Specifications**: Describes either the input or output data, depending on whether you are defining a source or destination form.
- **Other**: Allows you to categorize your connection documents in the LEI Administrator database, as well as add descriptive comments about the connection.

These sections and the fields in the sections are described in detail below.

### 6.5.1 Connectivity section

The Connectivity section (Figure 6-20) defines the file name and path of the source or target file. The fields in the Connectivity section include:

- **Name**: Specifies a unique name that identifies this connection. The maximum number of characters allowed is 255.
- **File Name**: Specifies the path of the source or destination text. When using long filenames, include the filename as is (with any embedded spaces). Do not use quotation marks in the filename since they will be considered part of the filename.
- **Connection Type**: Specifies whether the connection is a source or target.
6.5.2 Text specifications section

The Text Specifications section (Figure 6-21) describes either the input or output data, depending on whether you are defining a source or destination form.

- **File Type**: Specifies the type of file you are describing in this form.
  - Text: Specifies that the file type is text. Text files can be composed of either fixed-length or variable records.
  - Binary: Specifies that the file type is binary.

  By default, the Lotus Connector for Text processes files in text mode. Text mode translates each carriage return/line feed combination to mean a new record. In certain circumstances, when records are not delimited by carriage return/line feed combinations, you might want the Text Connection to process the data exactly as is with no text mode translations, and would select binary as the file type.

- **File Text Character Set**: If you want to override the default character set of the underlying Connector, enter the overriding character set in this field. For example, enter CP392 if you wish to override the default character set with the CP392 character set. The default character set is the character set that is native to the system on which you are working.

- **Record Delimiter**: If the data records are variable in length, enter the delimiter you are using. The default is \n, indicating a new line or carriage return.

- **Fixed Record Size**: Fixed-length records are determined by a user-defined decimal value that specifies the actual record length. Variable length records are determined by a user-specified delimiter in character or hexadecimal notation, for example, a carriage return/line feed. This field is mutually exclusive to Record Delimiter.

Field Specifications tab (Figure 6-22): A Text connection offers several options that specify how the data should be processed. For example, you can request that the Connection check for a minimum input record size, strip quotation marks from input text or add quotation marks to output text, and skip a specific amount of data before beginning processing. The fields are described below:

- Specify field options or, for a ZID file, specify @filename.zid: Either reference the ZID file by name (using the syntax: @filepath\filename.zid) or specify the actual field options. If you specify a filepath, use a path that’s meaningful to the LEI server, not a local file on
your workstation. The field definition format is described in the document, “Field Options and Text Information Definition Syntax” in the Domino Designer Help database.

- An ASCII table is provided to assist you in creating definition strings. The ASCII table lists the decimal, hexadecimal, and character equivalents for the ASCII character set. Supported escape sequence characters are indicated in red text. Character options for 0-255 are available.

<table>
<thead>
<tr>
<th>Field Specifications</th>
<th>Text Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Specify field options or for 2ID, field, specify @filename 2id)</td>
<td></td>
</tr>
</tbody>
</table>

**ASCII Table**

*Figure 6-22  Text connection document, Field Specification tab*

**Text Options - Source tab** (Figure 6-23): This tab allows you to specify options if the connection is the source in an activity. The fields are described below:

- **Input Options**: Check for Input Values Within Quotes, checking this option places quotes around field values in the output document. Select this option if you want the Lotus Connector for Text to remove quotation marks that surround input text. If you do not select this option, the Lotus Connector for Text puts text into the document fields exactly as supplied.

- **Minimum Record Size**: Specifies the minimum record size. Use this option when you need to set a minimum input record length. The default value is 1. A value of 0 indicates that no minimum record length exists. Specifying a value greater than 0 forces the Lotus Connector for Text to bypass input records that are smaller than the specified minimum. This feature is very useful for bypassing junk records or blank lines that might be present in an input text file.

- **Maximum Record Size**: Specifies the maximum record size in byes. If fields beyond a certain record size should be ignored, enter the largest valid record size here. The default value is 128K.

**Note:** If transferring information to or from Notes, the total size of “summary” fields (that is, non-rich text) is 32KB. Notes may store more than one byte per character.

- **Skip Records**: Specifies the number of records the Lotus Connector for Text should skip before it begins processing. This is useful for ignoring header information or extraneous white space at the beginning of a data file. The default is 0.

- **Skip Bytes**: Specifies the number of bytes the Lotus Connector for Text should skip before it begins processing. This is useful for ignoring header information or extraneous whitecap at the beginning of a data file. The default is 0.
Chapter 6. Connection documents

Figure 6-23  Text connection document, Text Options - Source tab

Text Options - Target tab (Figure 6-24): This tab allows you to specify options if the connection is the target in an activity. The fields are described below:

- **Target Options:**
  - **Output Values Within Quotes:** Selecting this option outputs your data with surrounding quotes. If you do not select this option, the Lotus Connector for Text outputs the data exactly as it exists.
  - **Append Data to Output File:** Selecting this option appends the data to an existing file. If you do not select this option, the Lotus Connector for Text processes output text as a new file and overwrites any existing data in the file. If you elect to append the data to an existing file, the Lotus Connector for Text writes the string specified by the record delimiter to the output file before any document processing begins. If the file does not exist, the Lotus Connector for Text creates one.

- **Fill Character/String:** Specifies the character that you want to occupy unused or whitecap areas in output records. If you do not enter a value, the Lotus Connector for Text defaults to ASCII blanks.

Figure 6-24  Text connection document, Text Options - Target tab

Text Options - Processing tab (Figure 6-25): This tab allows you to specify options for the processing of your text file. The fields are described below:

- **ZID Field Options:** Echo ZID Field Options, Enabling this option captures the ZID state values in a log file.
- **Maximum Records Processed, Stop after n records processed:** Specifies the maximum number of records that can be processed by a single activity using this connection.
- **Maximum Length of Simple Text Data in bytes:** Specifies that any record larger than this value will be handled as a composite format binary field.
- **Output File Header:** The text value entered here will appear as the first entry in the output file.
- **Output File Trailer:** The text value entered here will appear as the last entry in the output file.
Figure 6-25  Text connection document, Text Options - Processing tab

Text Options - International tab (Figure 6-26): This option helps LEI translate the language used in the file to be processed. The fields are described below:

- Days of week: Enables you to enter the days of the week in the expected language of the incoming text file. For example, for a Spanish text file enter Domingo, Lunes, Martes, and so on.
- Months of year: Enables you to enter the months of the year in the expected language of the incoming text file. For example, for a Spanish text file enter Enero, Febrero, Marzo, and so on.
- A.M. String: Enables you to specify the A.M. equivalent term in the expected language of the incoming text file.
- P.M. String: Enables you to specify the P.M. equivalent term in the expected language of the incoming text file.

Figure 6-26  Text connection document, Text Options - International tab

Text Options - Connection tab (Figure 6-27): The field in this tab is described below:

- ZID Script Replacement/Substitution Strings, format is "current string = replacement string": Enables you to specify a substitute string for the current string. You can change the definition of fields in your text file by changing the values of the ZID area of the connection. This lets you change the value assigned to the field to something else.
6.5.3 Other section

Refer to 6.1.3, “Other section” on page 78.

6.6 Metaconnectors

A metaconnector is a special kind of LEI connector that provides preprocessing operations on
connector data prior to transfer within a defined activity form. A metaconnector is usually an
intermediary between a source connection document and an activity. To take advantage of
the metaconnector’s preprocessing, an activity document refers to the metaconnector
connection document instead of the original connection document. For example, in a situation
where you use the order metaconnector to sort data from DB2 before doing a replication
activity with a Notes database, the replication activity document should use the order
metaconnector connection document as the source connection, not the DB2 connection
document. An illustration of how data flows when using metaconnectors is shown in
Figure 6-28.

Data flow when using metaconnectors

LEI supports the following metaconnectors:

- Collapse/Expand
- Meter
- Order
- Trace

For instance, create a DB2 connection called “EmployeeTable”.

Then create a Order metaconnection document that refers to “EmployeeTable” as its
“Connection to use”. Specify EMPNO as the sort key. Name this document “Employee Lookups
By EmpNo”. Now, when you create an activity, you have a choice. When you select which
connection to use, you can choose “EmployeeTable” and get the data in no particular order (most efficient if doing a Direct Transfer), or you can select “Employee Lookups By EmpNo” to get the data sorted by EMPNO field (which you might need for a Replication activity).

It’s also possible to “stack” metaconnectors. Suppose you were having a problem with your replication activity and you needed to see what SQL statements were being sent to the DB2 server. You can create another metaconnection document, in this case a “Trace”. Name the new connection “Trace Sorted Employee Records”. In the “Connection to use” field, enter “Employee Lookups By EmpNo”. Then you can edit your replication activity, and temporarily use “Trace Sorted Employee Records” as your connection, instead of “Employee Lookups By EmpNo”. The SQL commands now go through two levels of preprocessing — first to sort, then to log the commands and their results.

### 6.6.1 Collapse/Expand Metaconnector

The Collapse/Expand Metaconnector provides the capability to take multiple records from a data source table and collapse them to a single form field, or perform the reverse operation which is to expand the data into multiple records. This is particularly useful for converting multi-valued Notes fields into multiple rows in a relational table, and vice-versa. Since the Notes connector is the only one supporting multiple-value fields, this metaconnector is only ever used when Notes is the source or target.

The collapse operation happens during the Fetch; expand occurs during Insert or Update. Therefore, Collapse/Expand would normally be used to filter records between the activity and the RDB, not between the activity and Notes. This lets the activity work with the combined records. The typical situation is shown in Figure 6-29. Multiple relational rows with a common key field are combined into a single row, including fields that contain multiple values. The rows in the relational source have to be sorted by the grouping keys (in this case ID), so that the records that need to be combined are adjacent in the result set. This sorting can be done either with an Order metaconnection, as shown in the diagram, or by working with a data source that’s already sorted. In some cases, the activity arranges for the data to be sorted, as discussed in 7.8.4, “Create a Collapse/Expand Metaconnection for the RDB” on page 136. In that case, the Order metaconnection on the RDB is usually not required.

![Figure 6-29 Typical use of Collapse/Expand metaconnector](http://www.ibm.com/support/docview.wss?uid=swg21102955)

There is a technote (#1102955) that describes common misconceptions regarding the Collapse/Expand Metaconnector. This technote can be found at:

To create the Collapse/Expand Metaconnector connection document, open the LEI Administrator database (decsadm.nsf) on the LEI server. Click on the Add Connection button and then click Collapse/Expand Metaconnection. See Figure 6-30.

![Figure 6-30 Creating Collapse/Expand Metaconnection document](image)

See Figure 6-31 for an example of the Collapse/Expand MetaConnector connection document. The fields in the Collapse/Expand Metaconnector connection document are described below:

- **Name**: Specifies a unique name that identifies this connection. The maximum number of characters allowed is 255.
- **Connection to Use**: Specifies the name of the underlying connection that is providing the data. This will often be an Order Metaconnector.
- **Grouping Keys**: Specifies the field or list of fields that define the grouping key. One internal record is constructed from all external records with the same key value. To ensure that all records with this key value are collapsed into a single multi value record, make sure that the external result set is ordered using the Order Metaconnector by the grouping key fields.

  **Note**: Grouping occurs only on exactly equal keys, it is not possible to ignore case even if combining the Collapse/Expand Metaconnector with the Order Metaconnector.

- **Additional Write Keys**: Specifies additional key field or fields which define the unique key within a collapsed record. For example, if four orders by a single customer are collapsed into one document, with the customer number as the key, and one order is changed, then the order number may be required to ensure the key’s uniqueness across individual external records for this customer on expansion.

  **Note**: Additional Write Keys are only needed if there are fields in the relational table that don’t map to Notes fields. If it doesn’t matter whether the activity updates records into the “same” rows as it read them from, then you need not enter a value here.

- **Other Options, Disable Trimming of Text Trailing Spaces**: Disables the trimming of trailing spaces in text data. In normal operation, whenever text data is fetched from a connection, trailing spaces are removed from the text. The property NoTrimText has been added to the direct transfer and replication activities. When using this option in a direct transfer or replication activity that uses a Collapse/Expand Metaconnector, you must also select this option here in the metaconnection document.
6.6.2 Meter MetaConnector

The Metering Metaconnector provides a way to collect statistical usage data. A Meter Metaconnection can identify and quantify data access.

To create the Meter Metaconnector connection document, open the LEI Administrator database (decsadm.nsf) on the LEI server. Click on the Add Connection button and then click Meter Metaconnection. See Figure 6-32.

See Figure 6-33 for an example of the Meter Metaconnector connection document. The fields in the Meter Metaconnector document are described below:

- **Name**: Specifies a unique name that identifies this connection. The maximum number of characters allowed is 255.
- **Connection to Use**: Specifies the name of the underlying connection that is providing the data.
- **Subtotal Frequency**: Specifies how often to record subtotal results computed in number of records. Subtotals can be printed during the process. The number specified is how many records to process to get subtotal information. For example, if you process 1000 records...
and the subtotal frequency is set to 300, you would receive 3 subtotal results and a final tally.

► Key Fieldname: Use this to generate results grouped by a particular data field value. This is an optional field containing the value to group results by.

► Meter Filepath: Specifies the path to the meter log file.

**Note:** The meter log file is written to a directory accessible from the LEI server. To transfer this file into a database, follow the metering activity with a direct transfer using the Lotus Connector for Files to access the meter log file and any other Connector as the destination.

► Record Level: Selects the items to track at the record level. The options are:
  – Track Bytes: reports the number of bytes per record.
  – Track Keys: reports the key field value(s) for each record.
  – Track Timestamp: reports the current time for each record.

► Total Level: Selects the items to track at the subtotals and totals level. The options are:
  – Track Bytes: reports the number of bytes per record.
  – Track Records: reports the number of records transferred.
  – Track Keys: reports the key field value(s) for each record.
  – Track Timestamp: reports the current time for each record.

![Figure 6-33 Meter Metaconnector connection document](image)

### 6.6.3 Order MetaConnector

The Order Metaconnector is useful when ordering data sets from different server sources. For example, a DB2 UDB table on an iSeries server and a Notes database on a Domino server may use different sorting methods when ordering data. This can result in a data set comparison problem when using a LEI replication activity, which requires data sets to be ordered in parallel for data set comparisons. You can use the Order Metaconnector to sort the DB2 data the same way that Notes sorts data, ensuring the order pattern will be in parallel for accurate data set comparisons during the Replication Activity. This is particularly an issue when replicating DB2 data with other databases.

For more information on this issue see technote 110127, LEI Addressing Sort Order Issues When Replicating DB2/400 Data to a Notes Database accessible at:

To create the Order Metaconnector connection document, open the LEI Administrator database (decsadm.nsf) on the LEI server. Click the Add Connection button and then click Order Metaconnector. See Figure 6-34.

See Figure 6-35 for an example of a Order Metaconnector connection document. The fields in the Order Metaconnector document are described below:

- Name: Specifies a unique name that identifies this connection. The maximum number of characters allowed is 255.
- Connection to Use: Specifies the name of the underlying connection which is providing the data.
- Metadata Name: Specifies the name of the selected table. When you select the connection to use, the metadata name from that connection appears in this field.
- Owner: Specifies the owner name of the selected metadata.
- Text Trimming: Allows you to disable trimming of text trailing spaces.
- Sort Character Set (optional): Specifies the optional character set to force text data into. By default, the underlying connection's character set is used. To provide a character set, use the corresponding constant suffix. Character sets are listed in the Lotus Connector LotusScript Extensions Guide. For example, for character set Code Page 932, represented by the LEI constant LCSTREAMFMT_CP932, enter CP932 here. This selection can affect the sort order of the data; if Binary sorting is used, then text fields are sorted according to the binary character codes in the character set you select here.
- Sorting Fields *if not provided in Activity: Contains a list of fields in which to sort data. For activities where ordering is specified, such as replication, these selections are overridden and therefore ignored in the connector form. For other activities, this is a place to provide the ordering (which was not available before).
- Sorting Options: Specifies whether to sort in ascending (default) or descending order. LEI replication depends on an ascending sorting to operate properly. Do not use the Sort Descending option with a replication activity.
- Order: Specifies how text is to be sorted. The choices are:
  - Sort Text as Binary
  - Sort Text with Case Sensitivity
  - Sort Text without Case Sensitivity
6.6.4 Trace MetaConnector

The Trace Metaconnector allows you to trace events associated with a specified sub connection. You may specify options including where to capture data and whether to include a timestamp with each trace log entry. The Trace Metaconnector is a tool for you and the Lotus support staff to use when troubleshooting.

To create the Trace Metaconnector connection document, open the LEI Administrator database (decsadm.nsf) on the LEI server. Click on the Add Connection button and then click Trace Metaconnection. See Figure 6-36.

Figure 6-36 Creating a Trace Metaconnector connection document

Figure 6-37 shows an example of Order Metaconnector document. The fields in the Order Metaconnector document are described below:

- Name: Specifies a unique name that identifies this connection. The maximum number of characters allowed is 255.
Connection to Use: Names the underlying connection to be traced.

Trace is displayed on screen: Specifies whether or not to write trace data to the LEI console window.

Trace is written to a file: Allows you to capture trace data to a text file. All output captured to a log file is appended to the end of the file. If the output file does not exist, it will be created. If the log file cannot be opened, a message indicating this will be sent to the activity log. The log file can grow quite large. You can delete the log file when the trace data is no longer needed.

Log Filename: User-specified log file name that allows you to specify a log file name. This option can provide more control over the trace output destination.

**Note:** The default log file name is trace.log. If you specify some other name (using the Log Filename option), such as "c:/foo/mytrace.log" on Win32, then the initial output, up to the point where the file name parameter is read, will be sent to "trace.log", and trace output after that point will be logged to the "c:/foo/mytrace.log" file.

Trace is written to the activity log: Determines whether or not output will be written into the activity log.

Record time stamp with each logged entry: Records a timestamp with each entry that is captured in the log file.

A common mistake people make when using a Trace metaconnector, is just to create the metaconnection document and fail to associate it with an activity. Let's say that you're trying to debug a Replication activity that uses the “Connect to DB2 Address File” connection. Simply creating the Trace metaconnection will not have any effect. You must also change the Replication activity to use the Trace metaconnection instead of the “Connect to DB2 Address File” connection.
Batch activities

Batch activities run on the LEI server, usually on a schedule, with no associated user workstation. They do their work, then quit. By contrast, Advanced Realtime activities, described in the next chapter, run continuously while they are active, and handle user requests for information or action via the Lotus Notes client or a Web browser interface.

This chapter discusses each of the different kinds of batch activity, and gives examples that you can follow along using the Additional materials you can download from the Redbooks Web site. To run the examples, you will need:

- Downloaded sample Domino databases. Refer to Appendix C, “Additional material” on page 251 for instructions on downloading and installing the sample files.
- A relational database system that the LEI server has access to, and in which you are authorized to create tables.
- A Domino server with LEI installed. Do not use a remote administered server. You can administer batch activities remotely, but we don’t in these examples.

**Note:** To minimize the wait for these test activities to run, edit the server configuration document in the LEI Administrator Configuration view. Set the Poll Interval to something small, say 10 seconds. Save and close the document, and shutdown and restart the LEI server to make the change take effect. On a production server you would want a longer poll interval, for efficiency.

You should be able to do these examples with any relational database that LEI supports. For details of setting up your LEI server with client software or ODBC connections to access the database, refer to the database vendor’s documentation. If using iSeries or AS400 platforms, the IBM Redbook, *Implementing IBM Lotus Enterprise Integrator 6 on the IBM @server iSeries Server*, SG24-6941, contains DB2 setup guidelines.

This chapter describes the batch activities supported in LEI 6 and provides detailed examples of command, direct transfer, scripted and replication activities.

These examples and the ones in the next chapter are meant to be done in order — most examples use data that you created in previous examples.
A “Command” activity type lets you enter one command in the native SQL of whatever kind of database you’re connecting to. For a Notes database, enter a macro formula to execute against the documents in the Notes database.

When you execute a Command activity, LEI establishes a connection, executes the command, and closes the connection, which automatically commits any transactions.

**7.2 Example 1: Setting up a Command Activity**

For all these examples we used DB2, but you can use whatever database is available to you. It’s probably better not to use MS Access; it will work, but since Access doesn’t use owner names, there are more differences between it and the screen shots of the examples. Proceed as follows:

1. Open the LEI Administrator database (decsadm.nsf)
2. Disable the user assistant (see page 8).

**7.2.1 Create a connection document to the RDB**

1. Create a connection document to your relational database using the action Add Connection....

   Fill in the connection document fields as listed below (as shown in Figure 7-1):

2. Name: enter Sample DB.
3. Server, Database, User Name, Password, and so on: enter your connection information.
4. Other fields in Connectivity section: accept default values.
5. Selection Type: Table
6. Owner: Click the selection button. If you get a list of owner names, your connection has been defined correctly. Otherwise, an error message will tell you what went wrong. If you have a problem, refer to Chapter 5, “Diagnostic tools and techniques” on page 67.
7. Optionally, choose an owner/schema where you may create new tables.
8. Optionally, enter a category and comment in the Other section.
9. Click Save and Close to save the new connection document.

Notice we haven’t chosen a table. We have to if we’re going to use a metaconnector on this connection, but otherwise it’s just a convenience; when we select to use this connector in an activity, this provides a default value for the table name used in the activity.
7.2.2 Create the Command Activity

The command we’ll execute will create a table in the relational database. We’re using DB2 for our example, so the command is written in DB2 SQL. If you using a different database, your SQL syntax may be different. Follow the steps to create a document like that shown in Figure 7-2.

**Note:** The command language for a Notes connection is the Notes formula language. The formula is applied to each document selected by the Notes connection document (much like a formula agent set to run on selected documents).
1. In the Name field, enter Create AGENTS Table.

2. Click the Connection selection button and select the connection document Sample DB.

3. In the Command Statement field, enter the SQL command you would like to execute. For this example, we used:

   ```sql
   CREATE TABLE tablename (
       ID INTEGER UNIQUE NOT NULL ,
       FULLNAME VARCHAR (60) NOT NULL ,
       EXPERTISE VARCHAR (400) ,
       PICTURE BLOB (50 K) NOT LOGGED NOT COMPACT ,
       ASSIGNMENT VARCHAR (100)
   )
   ```

   You should adjust the table name and SQL syntax for your target system. You want to create a table with the attributes shown in Table 7-1.

   **Note:** If you don't want to take the trouble to figure out the SQL command for your database platform, you can skip this example — the next example can create metadata automatically if it needs to, or you can create it using the tools provided by the database vendor.

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Datatype</th>
<th>Size</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>n/a</td>
<td>non-null, unique</td>
</tr>
<tr>
<td>FULLNAME</td>
<td>Text</td>
<td>60</td>
<td>non-null</td>
</tr>
<tr>
<td>EXPERTISE</td>
<td>Text</td>
<td>400 (255 will do)</td>
<td></td>
</tr>
<tr>
<td>PICTURE</td>
<td>Binary/BLOB</td>
<td>50KB (if applicable)</td>
<td></td>
</tr>
<tr>
<td>ASSIGNMENT</td>
<td>Text</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Activity Execution Options
Moving further down the form, there is a section of options you can set, shown in Figure 7-3. The default values will do for this example, but you should be familiar with these settings, which appear on all the batch activity forms.

![Activity Execution Options](image)

- **Designated Server:** In an LEI Clustered environment, you may want to select which server you intend this activity to run on. If you leave this field blank, it will run on the first of them that happens to notice the activity waiting to run.
- **Dependent Activity(s):** You can use the button here to select other batch activities that should be executed after this one completes. If you select more than one, they will run in parallel (if server load constraints allow).
- **OS Priority:** If the task will take some time, select how much attention you want it to get from the CPU, relative to other processes running at the same time. For a short task like this one, this setting is unimportant.
- **Enable Logging:** Entries are written to the activity log giving statistics about the job — how many documents transferred, how many deleted, and so on. For a Command activity there are actually no statistics to report, because LEI isn’t handling the documents itself to count them. The field is included for consistency with other forms.
- **Buffer Logging:** Buffering the log messages is more efficient, but you run the risk of losing log entries if the process crashes. Leave this checked unless you are experiencing crashes and need to be sure you have all the log entries to help diagnose the problem.
- **Send Log Report and Status:** Controls whether e-mail is sent when the activity runs, when it ends with an error, or not at all. If you choose something other than Never, additional fields will appear to let you enter the e-mail recipient and select plain text or rich text.

Scheduling Options
The Scheduling tab on the Command activity is shown in Figure 7-4. The default settings will do for the purposes of this example, but these options appear on all batch activity forms, and it helps to know what they do.
Run Activity Once and Disable. This option is handy for setting up big jobs that you only need to run one time, during non-working hours. So that you don’t have to be working during non-working hours, set it up to run at a given time, enable the schedule, then go home and check the result in the morning.

Schedule: Select whether the activity should run at a prearranged time and interval. If you don’t enable the schedule, it will only execute when you manually request it, or when another activity finishes, that lists this one as a dependent activity.

The next several fields are self-explanatory.

Retry Option: Selects what happens if the activity fails with an error. If you check this field, the form prompts for additional information: how long to wait between retries, and how many times to try before giving up.

7.2.3 Execute the Command Activity

1. Save and Close the Command Activity form.
2. Go to any of the activity views and highlight the activity Create Agents Table.
3. Click the Start Activity button.
4. While waiting for the LEI server to start the activity, an hourglass icon is displayed beside it in the view. Press F9 periodically to update the view display. A radio tower displays while the activity is running (this activity might not run for long enough for you to see this).
5. If the icon changes to a suitcase, the activity completed successfully. If it displays a red X, there was an error. To find out what the error was, click the Current Activity Execution Log button.
6. Use the tools supplied by your database vendor to confirm that the table got created and has the right field types.

7.2.4 Create the AGENTEQUIP table

Using the same steps as above, create a second Command activity to create the AGENTEQUIP table. In this case, the command you should execute is:

```sql
CREATE TABLE yourschema.AGENTEQUIP (
  ID INTEGER NOT NULL ,
  DESCRIPTION CHARACTER (60) NOT NULL ,
  DATEISSUED DATE NOT NULL ,
```
Chapter 7. Batch activities

7.3 Direct Transfer Activity

A Direct Transfer activity copies data from one database to another. The data to be transferred is identified by a statement, for example a SQL query or Notes selection formula. Direct Transfer activities can be combined with other activities; for example, you might use a Polling activity to detect the presence of new data, then use Direct Transfer to import the data.

Direct Transfer is the fastest way to move data from one place to another if you know that you want to always add the new data, instead of updating existing data. It does allow an option to replace all the existing data in the target (in relational terms, it “truncates” the destination table). To make sure database B contained an up to date copy of the data in database A, you could overwrite all the data in B with a nightly Direct Transfer.

Overwriting the previous data is reasonably efficient for relational databases and text file exports, but this plan has drawbacks when applied to Notes documents. Notes documents contain extra information — document IDs and other header information — that are assigned when the document is created or edited. When you delete one you’re not deleting just the data in the fields, you’re also throwing away this header. This disables several functions that are important to Notes and Web browser users.

▶ Doclinks: To call a document to someone’s attention, Notes users and applications use Notes mail to send a link to the document. If the document is deleted and a new one created, even if the new one contains identical field data, the doclink doesn’t work, because it relies on the document Universal ID (UNID), a unique identifier assigned when the document is created.

▶ URL links: The URL of a Notes document also contains the UNID of the document. Therefore, users of your Web apps cannot create a “Favorites” link to the document.

▶ Replication: Replication with other Domino servers and Notes clients will still work if you delete and replace all the documents, but it will be much, much slower. The replication algorithm used by Notes transmits only those documents that have changed since the last replication. Furthermore, Notes tracks the last modified time of each field in the document, and the replication algorithm only transmits those fields that are different. If you delete and recreate all the documents, you get a double whammy. Since every document is new, you have to send every document. In addition, every old document that you deleted left behind a “deletion stub,” which Notes uses to remember which documents have been deleted. All these deletion stubs must also be replicated to other servers so that they will know to delete their copies of these documents.

Table 7-2 Definition of AGENTEQUIP table

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Datatype</th>
<th>Size</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>n/a</td>
<td>non-null</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Text</td>
<td>60</td>
<td>non-null</td>
</tr>
<tr>
<td>DATEISSUED</td>
<td>Date</td>
<td>n/a</td>
<td>non-null</td>
</tr>
<tr>
<td>STATUS</td>
<td>Text</td>
<td>20</td>
<td>non-null</td>
</tr>
</tbody>
</table>

As before, you should supply your own schema name and adjust the SQL as required for your target system, to create fields as described in Table 7-2.
View indexing: For efficiency, Notes caches the field values that appear in views, storing them in a “view index.” When a user opens a view, Notes doesn’t scan the entire database to come up with the view to display. Instead, it scans for documents that were added, deleted, or updated since the view was last used, makes the necessary updates to the view index, then displays rows from the index. If you modify every document, then the next user to open the view has a long wait, because every document in the database has to be examined to see if it goes into the view.

There are ways around this last problem; the Notes connection document contains an option to recalculate all the view indexes once the task is complete. This has its own drawbacks, however. Refer to 6.3.1, “Connectivity section” on page 80.

For these reasons, it makes sense to use a Replication activity to do regular updates to Notes information, instead of replacing Notes documents en masse.

## 7.4 Example 2: Using Direct Transfer

The next example uses a Direct Transfer activity to copy information from a Notes database into two relational tables. An example Notes document is shown in Figure 7-5. This document comes from a database used by a spy agency to keep track of its agents. Several of the fields correspond to the AGENTS table created in the Command activity example (see table definition on page 108). The Notes fields are:

- **AgentID** (number): corresponds to the AGENTS.ID.
- **AgentName** (text): corresponds to AGENTS.FULLNAME.
- **Specialties** (keyword, multi-valued): corresponds to AGENTS.EXPERTISE. Since RDB fields are not multivalued, this will be stored as a delimited string in the RDB.
- **Mission** (text): corresponds to the AGENTS.ASSIGNMENT.
- **Photo** (rich text lite, allowing pictures and attachments): corresponds to AGENTS.PICTURE, which is a BLOB type.

**Note:** Rich text stored in a relational database binary field is stored in a proprietary format not usable except by Notes. This is true even if all it contains is a picture or file attachment.

Rich text copied to a RDB text field can be used by other programs, but it will only contain the text from the Notes field, not any formatting or graphics.
Chapter 7. Batch activities

113

Figure 7-5   Sample Notes data to be transferred to RDB

In addition to the fields listed above, this Notes form contains three multivalued fields, EquipName, EquipDate, and EquipStatus, whose values you can see in the columns of the table at the bottom of the form. Each of the “Equip” fields must contain the same number of values (in the document shown, they each contain seven values).

These fields correspond to the ones in the AGENTEQUIP table, described in Table 7-2 on page 111. Unlike the Specialties field above, we’re not satisfied to store these multivalues as delimited strings. Instead, we’d like one row for each value, to create a many-to-one relationship with the AGENTS table via the key field ID.

Transferring data to this secondary table will require a second Direct Transfer activity.

7.4.1 Create a connection document to Notes

To use a batch activity to move data into or out of a Notes document, you must have a Notes connection document to the Notes database (.nsf file) that you want to use.

1. Download and set up the two Notes sample databases, as described in Appendix C, “Additional material” on page 251.

2. From the LEI Administrator main screen, click Add Connection... -> Notes, and fill in the fields in the Connectivity section as listed below and shown in Figure 7-6:
   - Name: Enter a unique name for the connection document; for this example use Notes Agent Records.
   - Domino Server: Fill in the name of the Domino server that hosts the database.
   - Notes Database: Type or use the button to select the filepath of the Notes database you want to use. For this example, type demo\LeiSampleData.nsf.
3. In the General Options section (see Figure 6-11 on page 82), enter the following:
   - Binding Options: Do Not Enforce Form Design. Binding options are used when updating Notes documents. Since we’re only copying from Notes, it doesn’t matter what we choose here.
   - Other fields: use default values.

4. In the Document Selection section, as shown in Figure 7-7:
   - Click the button beside “Form to Use” and select the Agent Record form.
   - Fields to Use From Form: don’t select any fields — we’ll use all of them.
   - Other fields: use default values.

5. In the Field Selection section, Other Options, check Translate Multivalue Types to Text. This takes care of changing the multivalue field Specialties to a delimited string for storage in the AGENTS.EXPERTISE field.

6. In the Data Creation and Data Transformation sections, use the default values. See 6.3.6, “Data Creation section” on page 87 and 6.3.5, “Data Transformation section” on page 86 for an explanation of these fields.
7. In the Other section, you may want to enter a category and comments for your own reference.

8. Click Save and Close.

### 7.4.2 Create the main Direct Transfer activity

We now have a connections to the Notes database and form, and a connection to the relational database (created in 7.2.1, “Create a connection document to the RDB” on page 106). We can now create Direct Transfer activities to use these connections to copy from Notes to the RDB.

The first activity we’ll create will copy data to the AGENTS table.

1. From the LEI Administrator main screen, select Add Activity -> Direct Transfer.

![Figure 7-8 Creating a Direct Transfer activity.](image)

2. Fill in the fields of the activity form as follows, as shown in Figure 7-9:
   - Name: Enter a unique name for the activity; for this example, type Copy Notes data to AGENTS table.
   - Click the button next to the Source prompt. Select Notes Agent Records as the source connection document (as shown in Figure 7-10).
   - Click the button next to the Target prompt. Select Sample DB as the target connection. This is the same document we used for the Command activity. There’s no reason you can’t use a single connection document for multiple activities, provided the options in the connection document are compatible with all the activities.
   - Since the Sample DB connection document doesn’t specify metadata, you are prompted to select it, as shown in Figure 7-11. Since you did specify that the metadata type was Table, and you also selected an owner, the choices list only tables belonging to that owner. When you make a selection, the fully-qualified table name is filled in, as shown in Figure 7-9.
**Note:** To use different metadata than the choices available in the selection boxes, (for instance if the target metadata does not yet exist) type the fully qualified table, view or procedure name into the field. However, you may only use the type of entity specified in the connection document — in this case, “Table.”

**Source**
- Connection: Notes Agent Records
- Form: Agent Record

**Target**
- Connection: Sample DB
- Table: AGUIRRE AGENTS

**Mapping:**
- Source Field(s): AgentID, AgentName, Mission, Photo, Specialties
- Target Column(s): ID, FULLNAME, ASSIGNMENT, PICTURE, EXPERTISE

**Select Statement:** SELECT @All

**Figure 7-9** First sample Direct Transfer activity form
3. Specify the field mappings — which fields in the Notes form correspond to which fields in the AGENTS table. If you know the field names you can just type them in, one per line, but it's generally easier to click the Mapping selection button and use the field mapping dialog shown in Figure 7-12.
To use this dialog, click in the Source Fields to select the fields you want to copy from Notes to the RDB. Then click in the Target Fields to select matching fields in the corresponding order to the source fields. In this example, select ID first, then FULLNAME, then ASSIGNMENT, and so on. It’s not necessary to map all the fields on either side. However, for the activity to work you must map the same number of fields in source and target.

Instead of mapping fields manually, you can check the “Automatic” checkbox and select to map by name, or by position in the field list. For this to work, you have to be mapping all the fields. To map by position, the corresponding fields have to be in the same position in both connections. To map by name, they have to have the same names. Neither one is true in this case, so we have to specify the field mapping ourselves.

AgentID is the key we’ll use to uniquely identify a record. However, Direct Transfer doesn’t care about keys. It doesn’t search for records or care whether they’re sorted; it just selects from one table and inserts to the other. So you don’t need to say which are your key fields for a direct transfer.

4. Enter the Select statement to select records from the source. When copying from an SQL data source, you would use an SQL statement such as `SELECT * FROM AGUIRARD.AGENTS`.

In this example, the source connection is Notes, so the selection statement is written in the Notes macro language. Enter:

```
SELECT @All
```

5. Data Transfer Options
   a. Precision Options
      Data Truncation Options: Select Allow Precision Loss Only (the default). We’re selecting that if a text or binary value is too long for the field you try to copy it to, the activity will log an error, but loss of numeric precision, or of a time from a date/time value, is not a problem.
   b. Source Data Options: accept default values.
   c. Target Data Options
Existing Data Options: check “Overwrite Existing Data”. This will make LEI “truncate” the target table (delete all rows) before starting to copy data. This is necessary if you want to run the activity multiple times, or it will duplicate existing data, causing an error if the table contains a unique key.

It's at this point that you have the opportunity to make LEI create the destination table automatically by checking Create Target Metadata. This works much better if the source connection is another relational database, instead of Notes. Since Notes doesn't have any equivalent of character field lengths or the non-null flag, the target table would be created with default options which might not make sense for your data. For this example, do not check Create Target Metadata.

You may notice when you check “Overwrite Existing Data”, the checkbox “Try Update Before Insert” becomes hidden. This option modifies the usual behavior of the Direct Transfer so that it first tries to update existing records with a keyed Update. If you enable this option, you will be asked to supply the key fields. “Try Update Before Insert” is not quite the same as a Primary Key replication (7.7, “Replication activity” on page 130), because it will not delete entries from the target if they have been removed from the source, but it's a good choice when you know source data are never deleted and when the destination table contains fields not in the source.

**Note:** “Try Update Before Insert” will probably have poor performance if the target connection is Notes and the Notes database contains many documents. Keyed updates in Notes are slow because of the Notes architecture.

d. Performance Options

Number of Records to Transfer Concurrently: 10. This will result in a fetch from Notes and an insert to the RDB of up to 10 records at a time. Depending on the connector, this multi-row transfer may be substantially more efficient than the default one at a time.

6. Activity Execution Options: use defaults. See “Activity Execution Options” on page 109 for an explanation of these options.

7. Scheduling: use defaults. See “Scheduling Options” on page 109 for an explanation of these options.

8. Click Save and Close to save the new activity, and return to the main LEI Administrator screen.

At this point, if you wish, you can start the activity and use the tools supplied by your database vendor to see the results. If it doesn’t seem to be working, refer to 5.4, “Activity debugging techniques” on page 71 for steps to figure out what’s wrong.

### 7.4.3 Create a Notes connection document by copying

As discussed in 7.4, “Example 2: Using Direct Transfer” on page 112, the Notes form has multivalued data that should be copied to a secondary table that's related to the main AGENTS table by a key, ID. The ideal design of this table is shown in Table 7-2.

Doing this requires a second Direct Transfer activity.

In the current example, however, we are not going to create the table exactly as shown here; instead, we'll let a Direct Transfer activity create the table automatically. This uses some default options for storing these datatypes. If the source were another relational database, things like the non-null option and the sizes of text fields would copy over to the new table. However, in a Notes form definition there is no equivalent of a text field length or a non-null
option, so the table fields will be created with some default options appropriate to the datatype.

As also discussed previously, one Direct Transfer goes to a single table. To write data to this second table requires a second Direct Transfer activity. This one will be special in that one Notes document no longer corresponds to a single row in the relational table. For instance, storing the equipment data for the sample Notes document shown in Figure 7-5 on page 113, will require seven rows.

Later sections will discuss how to accomplish this translation. But first, we have to have the data to translate. We already have a connection document to the Notes database and form. But this connection has the option Translate Multivalue Types to Text enabled. An activity using this connection will not be able to tell the difference between a multivalue field and a text field that happens to contain semicolons. So we have to have a second Notes connection document with this option disabled.

Since we already have a connection document that’s very close to what we want, the quickest way to create the new one we need is by copy and paste.

1. Go to the Connections view of the LEI Administrator and highlight the Notes connection document Notes Agent Records.
2. Use Edit -> Copy and Edit -> Paste (or Ctrl+C and Ctrl+V) to create a new copy of the document.
3. Since multiple connections with the same name aren’t allowed, you will be prompted whether you want to rename the new connection. Answer Yes (If you answer No the paste will not be allowed).
4. Enter a new name, Notes Agent Equipment.
5. Highlight the new connection document and select Actions -> Edit Document.
6. Go to the Document Selection tab and check the fields we need for this activity: AgentID, EquipName, EquipDate and EquipStatus, as shown in Figure 7-13.

This step is not needed to make the direct transfer work, but it is more efficient to only fetch the fields you need, particularly if some of the fields you don’t need might be large (rich text fields, especially).
### 7.4.4 Copy the RDB Connection

For the next phase of this operation, you will need a connection document for the relational database, similar to Sample DB but with metadata selected. As in the last section, we will copy and modify a connection document to make this happen.

**Note:** Changing the metadata or any other parameters in a connection that's already in use by an activity, is very likely to break the activity.

1. Open the LEI Administrator to the Connectors view and highlight the Sample DB connection.
2. Use **Edit -> Copy** and **Edit -> Paste** (or Ctrl+C and Ctrl+V) to create a new copy of the document.
3. As in the previous section, you are prompted whether to enter a unique name for the new connection; answer **Yes** and type the new name, Sample DB AGENTEQUIP.
4. Use **Actions -> Edit Document** to edit the new connection document.
5. Click the selection button next to the **Table** prompt, and select the AGENTEQUIP table.

### 7.4.5 Collapse/Expand Metaconnection

We needed the Sample DB AGENTEQUIP connection document, to create a metaconnection document. In this case, we want to make multivalue fields coming from Notes get translated into multiple rows in the AGENTEQUIP table. The metaconnector sits between the activity and one of the connections. As described in 6.6.1, “Collapse/Expand Metaconnector” on page 98,
Collapsible/Expand collapses on read and expands on write; therefore, in general, it sits between the activity and a relational connection.

Figure 7-14 shows the metaconnection document we need. Fill in the fields as follows:

1. Name: Choose a unique name for the connection. The name must be unique among all connections and metaconnections.

2. Connection to Use: Use the button to select Sample DB AGENTEQUIP connection, which we created in the previous section.

3. Metadata Name: filled in automatically with the table name from the connection document.

4. Grouping Keys: Select “ID”. These are the names of the key fields for the many-to-one relationship between the table listed here and the Notes document. You would select one or more keys from the “Connection to Use”; those keys must uniquely identify a Notes document. In this example, the ID key does that (corresponding to the Notes field AgentID).

5. Additional write keys: Leave blank. These are useful during two-way replication or in virtual fields activities, if you need to uniquely identify the row from which the multivalued data came so that when a value changes, you can update that same row. Refer to 6.6.1, “Collapse/Expand Metaconnector” on page 98 for more information.

Figure 7-14   Collapse/Expand metaconnector

An important note about this example: often, the Collapse/Expand requires that the data in its connection are sorted. When it scans the entries in the table — for instance, while doing a Replication — the data with the same keys have to be grouped together when they are fetched. However, the data in the Sample DB AGENTEQUIP connection are not sorted. We can get away with that here, because in our Direct Transfer, we’re only writing to this connection, never reading from it to create a merged row. The Replication example (page 132) shows the more typical case.

7.4.6 Create the secondary Direct Transfer activity

You now have all the connections you need to transfer information from Notes, through a Direct Transfer activity, to a collapse/expand metaconnection, to the AGENTEQUIP table. In this section, we create the Activity document that makes this happen. The new Direct Transfer activity is shown in Figure 7-15 on page 124. Fill in the fields as follows:

1. Name: Enter a unique name for the activity; for this example, enter “Copy Notes data to AGENTEQUIP Table”.

---

<table>
<thead>
<tr>
<th>Connectivity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>CollExp Agent Equipment</td>
</tr>
<tr>
<td>Connection to Use:</td>
<td>Sample DB AGENTEQUIP</td>
</tr>
<tr>
<td>Metadata Name:</td>
<td>AGUARD.AGENTEQUIP</td>
</tr>
<tr>
<td>Grouping Keys:</td>
<td>ID</td>
</tr>
<tr>
<td>Additional Write Keys:</td>
<td></td>
</tr>
<tr>
<td>Other Options:</td>
<td>Disable Trimming of Text Trailing Spaces</td>
</tr>
</tbody>
</table>

Figure 7-14   Collapse/Expand metaconnector
2. Source: Use the selection button to choose the Notes Agent Equipment connection document (created in 7.4.3, “Create a Notes connection document by copying” on page 119). The Table value is supplied automatically.

3. Target: Use the selection button to choose the Coll/Exp Agent Equipment connection document. The Table value is supplied automatically.

4. Use the Mapping button, like in Figure 7-12 on page 118, to map the following Notes fields to their corresponding relational fields:

<table>
<thead>
<tr>
<th>Notes Fieldname</th>
<th>RDB Fieldname</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentID</td>
<td>ID</td>
</tr>
<tr>
<td>EquipDate</td>
<td>DATEISSUED</td>
</tr>
<tr>
<td>EquipName</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>EquipStatus</td>
<td>STATUS</td>
</tr>
</tbody>
</table>

5. Select Statement: Select which source records to transfer to the target. Since the source is Notes, this will be a macro formula. To see why we cannot SELECT @All here, as we did in the first Direct Transfer, look at the Notes document for Agent 17. He has no equipment; if we tried to create an AGENTEQUIP record for this agent, it would have null values in three columns that don’t accept nulls, causing the activity to error out.

To prevent this, we want this activity to only process documents for agents that actually have some equipment. Because of the logic on the Notes form, we can know that it’s only possible to create documents with an equal number of entries in the three Equip fields. Therefore, our selection formula can just test whether one of them contains a value:

```
SELECT @Elements(EquipName) > 0
```

6. Under Direct Transfer Options / Target Data Options, check the box to Overwrite Existing Data.

7. All else: use default values.

8. Save and Close.
7.4.7 Establishing a chain of activities

Since we now have two Direct Transfer activity documents that operate on the same Notes documents, it makes sense to always run them together — otherwise, the data may be inconsistent between the two. To make this happen, you can enter “dependent activities” in a field of the activity document. Whenever a batch activity finishes execution, if there are any dependent activities listed in its activity document, those activities are executed immediately afterwards.

In our example, we’ll set up the Copy Notes data to AGENTS table to run first; every time it runs, the Copy Notes data to AGENTEQUIP Table activity will follow.

1. Locate the Copy Notes data to AGENTS table activity in a view of the LEI administrator.
2. Open it, and use the Edit Document action to put it into edit mode.
3. Go to the Activity Execution Options tab, and find the Dependent Activity(s) field.
4. Click the selection button and choose Copy Notes data to AGENTEQUIP Table. The result is shown in Figure 7-16.
5. **Save and Close.**

At this point, if you run this activity, the other one would automatically run afterwards; to test this, you could edit one of the sample Notes documents, add equipment line items to one of the entries, start the activity and see whether the equipment ends up in the AGENTEQUIP table.

This capability is useful when you want to run multiple activities on a schedule, but one activity depends on another to complete before it can begin. You can schedule the first activity in the chain, and have the rest run as dependents to that one.

### 7.5 Scripted activity

Scripted activities let you execute LotusScript (LSX) commands by running an agent on the Domino server. Using the LSX, you can program any operations that require functionality not available in the activities provided by LEI.

To learn to use the LC LSX, see Chapter 9, “Getting Started with the LC LSX” on page 191. In this section we discuss setting up an activity to run this code.

You can write LC LSX code in any LotusScript function — it doesn’t have to be a scripted activity. For instance, in a Domino Web applications, you could make LC LSX calls from the Webqueryopen agent to load relational information into fields of the document.

However, scripted activity code is special because it runs in the context of an LEI activity. This gives it the ability to create activity log entries, and to load connection information by supplying the name of an LEI connection document (instead of separately specifying the database type, name, userid, password, and so on).

To create a scripted activity agent, you must have the Domino Designer client installed. Once the agent exists, you don’t need Domino Designer — you can use the LEI Administrator to make it run as an activity.

Scripted activities are not limited to the LC LSX. You can run any type of Domino agent this way. There are a couple of reasons you may want to run non-EI related agents this way.

- LEI runs scheduled activities exactly on schedule or as close to it as possible. The Domino agent manager, on the other hand, will defer tasks if its agent executives are already busy running other agents. If there’s something that *must* be done between 12:30 AM and 12:40 AM, an LEI Scripted Activity is a better choice than a scheduled agent.

- LEI can bypass some agent access restrictions, allowing the agent to access data on other servers that don’t consider the running server a “trusted” server.

If you have scripted activity agents from LEI 3.x, they will need to be updated to continue to function in LEI 6. The Migration tool takes care of some of this for you; refer to 4.1.1, “Migrating scripted documents” on page 58 for more information.
Some special considerations when creating a Scripted Activity agent in Domino Designer:

- The agent must not run on “Selected documents” or “All Documents in view”, since there are no selected documents and no view from which it’s run. An incorrect setting of the Trigger and Target properties of the agent will result in the error “Error: Lotus Notes: Invalid index (0x80000D01)” in the activity log when the scripted activity is run.
- If you use the LC LSX at all, you must create a LCSession object to specify the session name for logging purposes.
- When creating an LCConnection object, the constructor takes an LEI Connection document name instead of a connector name, as described in 9.2.2, “LCConnection Class” on page 193.
- If your agent exceeds the “Maximum duration” in the Activity Execution Options, LEI will send it a shutdown command and give it a couple of minutes grace period to finish up. If it doesn’t finish, LEI will forcibly end it. Unfortunately, there is no way in the current release for the agent to detect the shutdown signal (future versions may include this). Therefore, if your agent might exceed the timeout in the activity document, it should include a self-imposed timeout. When starting up, use the LotusScript Now function to find out what time it is and store that in a variable, then periodically subtract the start time from Now to see how long the agent has been running. This gives you a chance to do an orderly shutdown, closing files and making sure your changes are committed, rather than LEI cutting you off abruptly with things still in a mess.
- LEI will also send a shutdown command to the agent if the administrator manually stops the activity. There’s no way to predict this like there is with a timeout, so you should program defensively. When you need to keep data consistent in different tables, use a single LCConnector object; set the CommitFrequency property to 0, and use the LCConnection.Action method to manually commit a set of related transactions simultaneously.

### 7.6 Example: Creating a scripted activity

This example shows how we make LEI run a Domino agent on schedule as a Scripted activity. Since this activity accesses a Notes database directly (not through the Notes Connector) and no other data source is involved, you don’t need any connection documents to create the Scripted Activity document in the LEI Administrator.

However, the agent we’re going to activate in this activity uses the Sample DB connection document described on page 106. If you don’t already have this connection document, please do so now.

In this section, we don’t discuss the actual agent code, just how to set it up to run. For an explanation of the code in this example, see 9.8.1, “LEI Programming 1: Transfer data from profile documents” on page 223.

1. Customize the agent. Use Domino Designer to open the database demo\LeiSampleData.nsf, and go to the Agents view, as shown in Figure 7-17.
2. Double-click to open the agent. There are certain parameters in the first screen that you may want to customize, as shown in Figure 7-18.

- If you didn’t name your relational database connection “Sample DB”, enter the name here on the CONNECTION_NAME line.
- The agent will create a new table called “KEYWORDS”; if you prefer a different name, enter it in the KEYWORD_TABLE line. You may use a name of the form ownertablename.
Implementing IBM Lotus Enterprise Integrator 6

The agent will create an index named “Keywords_Ind”; if you prefer a different name, enter it on the KEYWORD_INDEX line.

If you like the values the way they are, at least make some trivial change — add a space to one of the blank lines, for instance — so that you can save it with an ID that your server will recognize.

3. Save your changes to the agent, and close the agent editing window.

4. Switch back to the Notes client. From the LEI Administrator main screen, use the action button Add Activity -> Scripted.

5. Fill in the new activity document as shown in Figure 7-19.
   - Name: type a unique name to identify the activity. For this example, enter Agent-Export Key Lists.
   - Agent Server: enter your server name here.
   - Agent Database: Type or use the selection button to select demo\LeiSampleData.nsf (or whatever path you used for the sample database).
   - Agent Name: Type or use the selection button to select Update Keyword Lists.
   - Designated Server: select your server name from the list.
   - All other fields: use the default values. For explanations of the fields, refer to “Activity Execution Options” on page 109 and “Scheduling Options” on page 109.

   ![Scripted Activity](image)

   **Figure 7-19** Scripted activity document


7. Find the new activity in any Activity view, and use Start Activity to start it running.
The result of this activity, once it completes execution, will be a table named KEYWORDS (or the name you specified in the agent) containing the keyword values used for the AGENTS.EXPERTISE and AGENTEQUIP.DESCRIPTION fields. The table contains two columns; TYPE, giving the keyword type ("SPEC" or "EQUIP") and VALUE, with a single keyword value.

![Sample Contents - KEYWORDS](image)

**Figure 7-20  Results of Update Keyword Lists agent**

**Note:** Scripted activities, unlike other types of activity, create multiple log entries; one for the activity itself, and one for each named session created by the agent. In 6.0.2 and later, you can view all the logs on a single screen using the Current Activity Execution Log button. For earlier versions, you have to open the log view and look for log entries with the session name specified in the agent.

**Example 7-1  Partial log entry from Update Keyword Lists agent**

<table>
<thead>
<tr>
<th>Activity: any name you choose</th>
<th>Processed by server: your server name here</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>08-05 06:37:17 PM - 08-05 06:37:21 PM</td>
</tr>
<tr>
<td>08-05-2003 06:37:17 PM</td>
<td>Activity started</td>
</tr>
<tr>
<td>08-05-2003 06:37:18 PM</td>
<td>Connected: LCONNECTION&lt; Name=&quot;Sample DB&quot;, IsConnected=(1), IsPooled=(0), Database=&quot;DB2V71&quot;, Userid=&quot;qepump&quot;, Password=NULL, Metadata=&quot;KEYWORDS&quot;, Index=&quot;&quot;, MapByName=(1), Writeback=(0), FieldNames=(binary), OrderNames=(binary), Condition=&quot;&quot;, StampField=&quot;&quot;, BaseStamp=&quot;12:00:00 AM&quot;, MaxStamp=&quot;12:00:00 AM&quot;, TextFormat=(65535), CharSet=&quot;NATIVE&quot;, Procedure=&quot;&quot;, Owner=&quot;&quot;, AlternateMetadata=(0), CallContext=(0), RecordLimit=(0), Scrollable=(0), Position=(0), CommitFrequency=(0), RollbackOnError=(0), CreateMaxLogged=(0), NoJournal=(0), TraceSQL=(0)</td>
</tr>
<tr>
<td>08-05-2003 06:37:18 PM</td>
<td>Removed all records from metadata 'KEYWORDS', Connector 'Sample DB', Method -Action [Truncate]-</td>
</tr>
<tr>
<td>08-05-2003 06:37:18 PM</td>
<td>Cleared old keyword list.</td>
</tr>
<tr>
<td>08-05-2003 06:37:18 PM</td>
<td>inserting: Type=&quot;SPEC&quot;, Value=&quot;Being Inconspicuous&quot;</td>
</tr>
<tr>
<td>08-05-2003 06:37:18 PM</td>
<td>inserting: Type=&quot;SPEC&quot;, Value=&quot;Causing Confusion&quot;</td>
</tr>
</tbody>
</table>
The log entry created by this agent is not typical — for purposes of demonstration, it contains a lot of text from the agent calling LCSession.LogText method.

7.7 Replication activity

A replication activity synchronizes data between two data sources, for example, Domino and DB2 UDB databases, or Oracle and Sybase. It can also be used to merge data from one data source to another.

You can use the replication activity in conjunction with a polling activity to refine when the replication activity occurs. For example, you could poll a data source for a specific event, such as a data insertion, and on detection of that event start the replication activity that replicates the new data into another data source.

There are two types of replication available from the LEI replication activity:

- Primary key replication is unidirectional, from “source” to “target”.
- Timestamp replication is bidirectional, but requires a timestamp column.

7.7.1 Primary key replication

LEI primary key replication replicates data based on a unique key common to both connections that can be composed of one or more fields in the metadata. The function of the primary key is to determine if matching records exist in both data sources and if an update to a record, the insertion of a new record, or the deletion of a record is required.

How does primary key replication work?

When replication occurs, records are replicated according to the following default rules:

- If a record with matching key values exists in both connections, and the data field values are identical, no action is taken.
- If a record with matching key values exists in both connections, and the data field values are different, the target record is updated.
- If a source record contains key values that are not present in the target, a new record is inserted into the target.
- If a target record contains key values that are not present in the source, that record is deleted from the target.

Note: In primary key replication, the replication activity never changes records in the source connection.
Primary key replication is often useful in conjunction with a Virtual Fields activity (Figure 8-1, “Activity Document — VF Agent Records” on page 160). You can use the Replication activity to create “stub” documents in the Notes database. Note, however, that if the Virtual Fields activity intercepts the “Create” action, it will intercept the replication event’s creation of a new Notes document and attempt to create a corresponding relational record, which would contain a duplicate key. For details refer to 8.1.4, “Keeping Virtual Field key documents up to date” on page 154.

### 7.7.2 Timestamp replication

As described in the previous section, primary key replication never changes records in the source connection. If you need to be able to make changes in both connections and keep the data synchronized, use timestamp replication. In timestamp replication, you still designate a source and target, but changes can flow either direction. The source/target distinction is used in resolving conflicts where the same record has been modified in both data sources — the source wins.

For timestamp replication to work, you must have a field in both the source and target connections that will be used as the timestamp. In Domino, the field needs to be of type Date, with the ability to store a date and time; you may instead use the “last modified” value from the Notes document header, but you must check the option in the Notes connection document to make this value available (refer to “Special Fields” on page 85).

**Important:** The RDB column used as a timestamp in timestamp replication must be a single column containing both date and time; for instance, for DB2 it must be of type TIMESTAMP, not DATE or TIME.

### How does timestamp replication work?

The LEI replication activity document maintains internal timestamp fields based on the last replication from each respective connection. These fields are SrcTimeStamp and DestTimeStamp. LEI stores separate values because two different servers may have some difference between their clocks.

You can see these fields by viewing the document properties of the activity document. To do so, right click while active in the activity document or choose File -> Document properties from the Action bar. Use the Fields tab and scroll down to locate these field names as shown in Figure 7-21.

![Figure 7-21 Timestamp Replication Activity Document Property](image)

When replication occurs, records are replicated according to the following default rules:
If a source record contains a timestamp value that is later than the SrcTimeStamp value, and a target record does not exist with the same key fields, a new record is inserted into the target.

If a target record contains a timestamp value that is later than the DestTimeStamp value, and a source record does not exist with the same key fields, a new record is inserted into the source.

If a source record contains a timestamp value that is later than the SrcTimeStamp value, and a target record exists with the same key fields, the target record is updated.

If a target record contains a timestamp value that is later than the DestTimeStamp value, and a source record exists with the same key fields, the source record is updated.

Records are never deleted from either connection. If another process deletes a record from one data source, it will be restored from the other data source when replication next runs.

**Note:** If a source and target record share the same key fields and contain timestamp values that are later than the SrcTimeStamp and the DestTimeStamp respectively, then the target record is updated. In this case, the replication activity does not use the timestamp values to determine the replication action.

### 7.8 Example: Primary key replication, RDB to Domino

In 7.4, “Example 2: Using Direct Transfer” on page 112, we saw how to set up an activity that copies data from one source to another (Notes to RDB in that case), overwriting the data in the target. In 7.3, “Direct Transfer Activity” on page 111, we discussed the drawbacks of overwriting data in a Notes database. Replication is the preferred method.

In this example, we’ll take advantage of the relational tables created for previous activities. We’ll use a special form and agent to make a change to the data in the RDB, then replicate that change back into the Notes data. This requires the use of both Collapse/Expand and Order metaconnectors.

As with the Direct Transfer, we have two relational tables that correspond to the same Notes form. The main table, AGENTS, contains a single row for each Notes document, while the AGENTEQUIP table may contain a variable number of rows with the same ID field value, corresponding to the multivalued fields on the Notes form with that ID. As with the Direct Transfer, this will require a Collapse/Expand metaconnection.

In addition, however, with a Replication activity, sorting is very important. This activity fetches all records from both sources and scans down each list looking for those with matching keys. Not only must both the source and target be sorted, they must be sorted in exactly the same order. Order metaconnectors sometimes come in handy for this.

Figure 7-22 shows the relationships of the different connection and activity documents we will use in this exercise.
7.8.1 Create Notes connection to Agent records with view selection

Like the Direct Transfer, we need a connection document to our Notes form. We can't reuse one of the existing ones because Replication activities need to access the Notes database in a different way — by means of a view rather than by a simple selection.

For the Direct Transfer, it made no difference what order the records were read in. For Replication, however, sorting is very important. That's why we don't just use the connection documents we already have. We need connections that are just a little bit different from the existing ones, so we'll copy and modify them.

The most efficient sorting for a Notes application is through the use of a view index. In the Notes connection document, you can select a view to filter and sort the entries. That's the change we need to make.

Note: A replication activity can’t use a Notes database as source or target without having a view sorted by the replication keys. If necessary, the activity will create the view on the fly and delete it when done. This is inefficient, so it's better to allow the replication activity to have a dedicated view that need not be re-created each time.

1. Go to the Connections view in your LEI Administrator, locate the Notes Agent Records connection document created earlier. Select this record in the view, copy and paste.
2. When prompted, answer “Yes” to rename the pasted record, and name it Notes Agent Records by ID.
3. Edit Notes Agent Records by ID document. In the Document Selection tab, Fields to Use From Form, select all the fields, as shown in Figure 7-23. It's not actually necessary to check all the fields, since the replication activity will override this selection to fetch your “mapped” fields, but you must at least check the key fields, in this case AgentID. These are needed for the Order metaconnector we'll create next.
4. In the Field Selection tab, type (ByID) in the View to Use for Selection field, and check Allow View Creation/Overwrite, as shown in Figure 7-24. The (ByID) view does not yet exist, but the Replication activity will create it.

5. The Translate Multivalue Types to Text option should still be checked, as in the original connection. It's needed for the Specialties/EXPERTISE field. In this case the data will be moving the other way, from the RDB to Notes, so text values will change to a multivalue.


7.8.2 Create Notes Connection to Agent Equipment with view selection

1. Highlight the new Notes Agent Equipment connection document in the view, copy and paste. Name the new copy Notes Agent Equipment by ID.

2. Edit the new document. This connection will be used to read the multivalued fields EquipName, EquipStatus, and EquipDate, which translate to multiple relational rows. To do this mapping, the connection will have to not convert multivalued fields to text. The connection we've copied already has the right fields selected, and has the multivalue option set the way we want.

3. View to use for selection: As shown in Figure 7-24, enter (ByID). In this case, it's not necessary to check Allow View Creation/Overwrite because the main document replication will have created it, though you can if you wish.
Chapter 7. Batch activities

7.8.3 Create Order metaconnections for Notes connections

**Attention:** This is a step that shouldn’t be necessary, since the Replication activity is supposed to be able to read sorted data out of the Notes view specified in the connection document. However, apparently this feature is not working correctly, so for current versions, the Order metaconnection is needed to sort the Notes data.

We now require Order metaconnection documents to sort the Notes data by the AgentID field, so that the replication activity will get them in order.

1. To begin, from the LEI Administrator main screen, select **Add Connection -> Order Metaconnection**.

2. Fill in the fields as shown in Figure 7-26:
   - Name: Enter a unique name for the connector; for this example, type **Order Notes Agent Records by ID**.
   - Connection to Use: use the selection button to select **Notes Agent Records by ID**.
   - Metadata name: automatically filled in with the form name from the Notes connector. While you could edit this field, it's not generally a good idea, because the Sorting Fields selection gives you choices from the default metadata on the connection document.
   - Sorting fields: select or type **Agent ID**.
   - Other fields: use default values.
3. **Save and Close.**

4. From the LEI Administrator main screen, again select *Add Connection -> Order Metaconnection*.

5. Create another document like the one shown above, but in this case select Notes Agent Equipment by ID as the connection to use.

### 7.8.4 Create a Collapse/Expand Metaconnection for the RDB

As explained in 6.6.1, “Collapse/Expand Metaconnector” on page 98, the Collapse/Expand filter collapses multiple rows with a matching key into a single multivalue during read, and expands multivalues into multiple rows during write. In most cases, it would be used between a relational connection and an activity, not between a Notes connection and an activity.

In 7.4.5, “Collapse/Expand Metaconnection” on page 121, we created a Collapse/Expand metaconnection document named *Coll/Exp Agent Equipment*, which we can reuse for this activity.

There’s an important point to note about this. The Collapse/Expand metaconnector requires that the data in its connection is sorted by the Grouping keys specified on the form. However, the *Coll/Exp Agent Equipment* refers to *Sample DB AGENTEQUIP*, which is not sorted. How is that going to work?

The answer is that when a connection is used in a replication activity, the replication activity takes advantage of the native sorting capability of the database server — in other words, it knows how to use the “Order By” clause in SQL. When the activity sends a Select command...
to the Collapse/Expand filter, Collapse/Expand passes it on to the database, and the data come back from the database already sorted.

You would need to use an Order metaconnector to sort the data if reading data from the RDB during an activity that did not automatically sort its data. For instance, a Direct Transfer from the RDB to Notes would require an Order metaconnection be used to pre-sort the data before the Collapse/Expand got to see it. The example Direct Transfer activity for which we created this Collapse/Expand doesn’t require sorting because it never reads the relational data — only inserts records.

### 7.8.5 Create the main Replication activity

The first of our two replication activities will replicate the fields associated with the AGENTS table. This will take care of creating new Notes documents if a new row is inserted, and deleting Notes documents when a row is deleted.

The other activity will update the data in the multivalue Notes “equipment” fields based on the data in the AGENTEQUIP table. This has to run after the main replication is complete, so that any records to which it should add information will already exist. It’s valid to have a secret agent without any equipment, but it’s illegal to have an agent document with equipment listed and no agent name.

To create the main replication activity:

1. From the LEI Administrator main screen, use **Add Activity -> Replication**.

![Figure 7-27 Activity — Replicate RDB Agents to Notes](image)

2. Fill in the fields as shown in Figure 7-27.
– Name: a unique name for the activity document; for this example use Replicate RDB Agents to Notes.

– Source Connection: use the selection button to select Sample DB.

– Source Table: Since Sample DB doesn’t refer to a table, only an owner name, you will be asked to select a table. Choose the AGENTS table.

– Target Connection: Select Order Notes Agent Records by ID.

– Target Table: Since this connection does define metadata, you will not be prompted. Use the automatically assigned value, Agent Record.

– Mapping: Click the mapping button and select key fields and other fields, as shown in Figure 7-28. This is similar to the Direct Transfer field mapping dialog in Figure 7-12 on page 118, except that in addition you must match the key fields in one connection to the key fields in the other.

3. Fill in fields in the tabs in the bottom part of the form, as shown in Figure 7-29. Since this is the first example of a replication activity, we’ll cover briefly what each setting does. For more detailed information, consult the LEI Activities and User Guide.

– Enable timestamp replication: Leave unchecked. This is a primary key replication, since there is no timestamp value we can use in the RDB.

– Log Key Values On Error: use the default value (enabled). If any error occurs, this will write the unique key identifying the error into the log.

– Save Conflict Data to Metadata: use the default value (disabled). If you check this field, you can select either the source or target connection, and enter a table name in that
connection (a form name for Notes) where any data that’s being deleted or overwritten will be copied to save the old value.

**Note:** The word “conflict” when talking about a Replication activity, means any difference in the data between the two sources — a normal event. This is different from the use of this term in Notes/Domino replication, where it refers to a situation of the same document having been modified in two different replicas, an exceptional condition calling for special handling.

- Enable Conflict Logging: Enable this feature. This makes a note in the activity log any time a difference is found in the data between the two sources. It’s useful to turn this on during initial testing of your activity so that you can tell whether there is excessive activity — for instance, whether a lot of records are being inserted and deleted, a sign of inconsistent sorting between the two connections.
- Logging Options: leave as default. You can change the appearance of the log entry for a conflict, or end the activity when a conflict is found. This lets you use a conflict activity as a way of comparing the data in two connections without actually modifying any. You can look at the log later and see which keys were not matched or had different data.
- Limit to: limits the number of log entries.
- Create Target Metadata: like the Direct Transfer activity, Replication activities can create their target table if it doesn’t already exist. Use the default value, disabled.
- Data Comparison and Trimming: this field lets you select how nearly the same values must be, to be considered identical. A common problem when moving data between different databases is that they might represent the data with different degrees of precision. For instance, one database might represent times to 1/100th second precision, while another might store times only to the nearest second. Similarly, a field stored in decimal format in one database might be stored in floating point elsewhere. Certain values, such as 0.1, can be represented precisely in decimal but not in floating point. Reduced precision comparison lets you ignore the tiny difference.

The remaining tabs, Activity Execution Options and Scheduling, are described starting on page 109.
Implementing IBM Lotus Enterprise Integrator 6

7.8.6 Test the main replication activity

Backup your data
Don't run any activity for the first time on your live data, of course. With a little test database like this one you can afford to mess around, because you can always download another copy. But for real applications it's very important to have a test system with tables defined identically to your production system. Replication activities in particular, because of their sorting requirements, are easy to set them up incorrectly. You may end up accidentally deleting Notes documents that contain other information that you wanted to keep.

In this case, we would suggest creating a temporary copy of the redbook sample Notes database (File -> Database -> New Copy) so that if the documents in the original get messed up, you can copy and paste to restore them.

Do the initial test run
The relational data and the Notes data should be identical at this point, since we did a Direct Transfer to copy over all the Notes data in an earlier example (7.4, “Example 2: Using Direct Transfer” on page 112). So if you run the Replicate RDB Agents to Notes activity, it shouldn't make any changes.
To try that out, go to the LEI Administrator main screen, highlight the Replicate RDB Agents to Notes activity, and click Start Activity.

Wait for the activity to finish running; press F9 repeatedly to update the view until the icon changes to a suitcase or red X.

Troubleshooting your replication activity

If your activity has a red X icon, there was an error that prevented it from completing. It might still have modified some data.

If the view shows a suitcase icon, the activity completed without an error, but that does not necessarily mean that it did what you intended; just as you may get in your car and drive somewhere without an accident, but that doesn’t mean you’re not lost.

Use the Current Activity Execution Log to see the result of the replication activity execution. Since nothing was changed, there should be only a few log entries, for example:

08-12-2003 12:41:55 PM Activity started
08-08-2003 12:17:48 PM Created Notes view '(ByID)', Connector 'Notes Agent Records by ID', Method -Select-
08-12-2003 12:42:01 PM Activity finished

If this is not what happens, review your connections and activity settings to make sure everything was done as shown. A message such as this one:

08-08-2003 12:17:48 PM *Replication Conflict* Action: Insert into Connector B; Record key values: AgentID = 17

means that the activity found a record with ID = 17 in the relational data but didn’t find a matching document in Notes, so it created a new Notes document. There are two reasons this can happen:

1. The Notes document actually does not exist, or
2. The records in the two connections are not sorted the same way, so 17 didn’t appear at the expected position in the result set to match the position of the 17 in the other list.

To see whether the latter is the case, look for an entry such as this:

08-08-2003 04:01:56 PM *Replication Conflict* Action: Remove from Connector B; Record key values: ID = 17

If there is an insert and a remove for the same key value, you can be sure that there’s a sorting problem. If it’s just that the data were not identical, then if you run the activity again you should not get an “insert” message.

Here are some other messages and their meanings:

08-08-2003 09:53:12 AM *Replication Conflict* Action: Update at Connector B; Mismatch in field '...'; Record key values: ...

This is not an error; it’s just telling you that it found a record that exists in both databases, with some fields that are not identical. So, it updated the record in the target connection.

This is a problem only if you keep getting it when you know there have been no changes to the data since the last replication. If that happens, it usually means that there’s a comparison precision issue, input translation, or blank trimming difference between the connectors. For instance, your Notes database can store times to 1/100th second precision. If the RDB stores times only precise to seconds, then 10:15:08.93 AM in Notes will never be equal to 10:15:08 AM in the RDB. You can compensate for this using the Reduced-Precision Comparison option in the activity. Likewise, if one connection has a
transformation on all updates to a field that translates the data to uppercase, you would need to enable Case-Insensitive String Comparison so that the activity will recognize the strings as the same value.

Error: Creation of a special view is required for ordered or timestamp selections, but replacement of view ' (ByID)' has not been enabled, Connector 'Notes Agent Records by ID', Method -Select-

You didn't check the Allow View Creation/Overwrite box in the Notes connection document. When the activity runs, it wants to create a temporary view to be able to select and sort the Notes records. If it can't, the activity won't run.

Error: Data overflow in field '...', Connector '...', Method -Insert-
Error: Type mismatch: Connector A field '...' is type ...; Connector B field '...' is type ...

These messages generally indicate some error in the field mappings.

Created temporary Notes view ' (LEIIndexView0)' ** Supply a view name to create a more efficient permanent view, Connector 'Notes Agent Equipment', Method -Select-

This doesn't prevent the activity from working — it's just advice for improving efficiency. Every time this activity runs it creates a temporary Notes view, then deletes the view when it's done. Provide a view name in the Notes connection document — it can be a view that doesn't already exist — and it will create or update the view the first time it runs and leave it there when it's done. Thereafter, it can use the view without having to create it each time. This can be a significant time savings, particularly in a Notes database with a large number of documents.

7.8.7 Update the relational data

Once you have the activity running successfully and not making changes when you run it on already identical data, you can make some changes to the source data and see whether they show up in the destination.

If you have a tool for editing the relational data, you can use that. Or, write command activities as we did in 7.2.2, “Create the Command Activity” on page 107. Or, use the SQL action button from the sample database. This button opens a form (shown in Figure 7-30) where you can select one of the connection documents in the LEI Administrator, and enter an SQL command to execute in that connection. This uses the command language of the base connector; for a Notes database you would enter a macro formula rather than SQL. In this example, I used the Sample DB connection to DB2, and the SQL commands we want to execute are:

► update yourschema.agents set FULLNAME = 'Susan Hilton' where (ID = 99)
► insert into yourschema.agents values (57, 'Bond Jennifer Bond', 'Snooping', NULL, 'Yurt surveillance')
Chapter 7. Batch activities

7.8.8 Retest the main Replication activity

Now that the relational data have been updated, you can run the activity again and see the results (we’ve left out the timestamps):

Activity started
  *Replication Conflict*  Action: Insert into Connector B;  Record key values:  ID = 57
  *Replication Conflict*  Action: Update at Connector B;  Mismatch in field 'FULLNAME';  Record key values:  ID = 99
Activity finished

As we were hoping, the new record 57 has been added to Notes, and the FULLNAME field in record 99 has been updated. Figure 7-31 shows the results in the Notes view.
7.8.9 Create a secondary Replication activity

The next step is to use the connectors we created earlier to make a replication between the AGENTEQUIP table (as the source) and the Notes documents (as the target).

1. From the LEI Administrator main screen, select **Add Activity -> Replication**.
2. Fill in the fields of the activity document as shown in Figure 7-32.
Chapter 7. Batch activities

Name: Enter a unique name for the activity; for this exercise use Replicate RDB Equipment to Notes.

Source Connection: select Coll/Exp Agent Equipment.

Source Table: filled in automatically with the AGENTEQUIP table name.

Target Connection: Select Order Notes Agent Equipment by ID.

Target Table: filled in automatically with the Notes form name Agent Record.

Target Restrictions: enable Skip Insertions and Skip Deletions.

The main replication activity makes the decision on whether there should be a record in Notes. We would not want to have this activity delete an agent record in Notes just because the agent has not been issued any equipment (as is the case for Agent 17, for instance). That's why we Skip Deletions in this activity.

The case for Skip Insertions is not as clear cut. If there's a line item in the AGENTEQUIP table with an ID that doesn't match an ID in Notes, the activity will try to create a record to store that equipment information (but we won't let it). If there's no Notes record, then there's also no record with that ID in the AGENTS table, because this activity will always run right after the main replication activity. That's either a data error in the RDB, or it represents an agent record that was added in the short interval between the first and the second replication activity. In either case, it would be wrong to create a Notes record that listed just an agent ID and equipment, and not the other information about them. If there is a new AGENTS, we'll catch it on the next replication.

Mapping: type the field names as shown above, or use the mapping dialog (Figure 7-33).

![Figure 7-33 Field mapping for Replicate RDB Equipment to Notes](image)
3. In the sub-tabs of the Replication Options, select options as shown in Figure 7-34.
   - Enable Conflict Logging, for the purposes of testing your activity. This lets you see every update it makes to the Notes database, including records it would have deleted or inserted had we not disabled those operations in the Target Restrictions.
   - Reduced-Precision Comparison: you may want to enable this option; it gives a better chance of the Replication activity realizing that fields have the same value.
   - Save and Exit.

### 7.8.10 Test the Secondary replication activity

The first time you run the replication activity, you should in theory see no changes since the equipment information hasn't been changed since it was last copied from Notes via Direct Transfer. In fact, we found that the replication activity doesn't seem to be able to detect whether multivalue date or number fields are equal.

#### Initial run with identical data

Highlight the Replicate RDB Equipment to Notes activity document in a Notes view, and use the Start Activity action to make it run. Once the icon in the view shows the run is complete, use the Current Activity Execution Log action to view the log document.
The log entry from the first run looks like this (leaving out the timestamps):

Activity started
*Replication Conflict* Action: Skip Remove from Connector B;
   Record key values: AgentID = 17
*Replication Conflict* Action: Update at Connector B;
   Mismatch in field 'DATEISSUED';
   Record key values: AgentID = 43
*Replication Conflict* Action: Update at Connector B;
   Mismatch in field 'DATEISSUED';
   Record key values: AgentID = 44
*Replication Conflict* Action: Skip Remove from Connector B;
   Record key values: AgentID = 57
*Replication Conflict* Action: Update at Connector B;
   Mismatch in field 'DATEISSUED';
   Record key values: AgentID = 86
*Replication Conflict* Action: Update at Connector B;
   Record key values: AgentID = 99
Activity finished

The “Mismatch in field ‘DATEISSUED’” entries represent the multivalue date comparison bug described above. The destination documents are being modified even though the values are already the same. Multivalued text fields don't have this problem.

This is still a successful replication, but it's a problem that it's making unnecessary modifications to the Notes document. This will make unread marks useless in the application, and will hurt performance for replication and view indexing. If you're using a timestamp replication (not the case in this example) there may also be excessive replication/save conflicts because documents are more likely to be modified while people are editing them.

The “Skip Remove” log entries tell us that there are two Notes documents — AgentID = 17 and 57 — which the replication activity would have deleted had we not checked Skip Deletions. These agents have no rows in the AGENTEQUIP table, which is a normal situation.

Modify Equipment data in the RDB
As we did with the main replication activity, we can again modify the relational data to see the result of replication. Use a COMmand activity or the SQL form (described in 7.8.7, “Update the relational data” on page 142) to make the following changes to the data.

- insert into yourschema.agentequip values (99, 'Tear gas pen', '07-15-2002', 'OK');
- insert into yourschema.agentequip values (17, 'Communicator Watch', '07-19-2002', 'OK')

Note: Your database's SQL syntax or date formatting may vary.

Re-run the activity
Run the activity again, and view the results in the sample database. You should see that equipment has been added to for agents 17 and 99, as shown in Figure 7-35.
7.9 Other data moving activities

LEI has other data moving activities that are valuable tools for your data moving needs. This section briefly discusses these activities. For more information about these activities, refer to the LEI Activities and User Guide.

7.9.1 Archive

The Archive activity moves data from one database to another. It is useful to place infrequently accessed data to a removable storage device to free space on a system server, protect data when migrating from one database to another, and create space on a full disk. The Archive activity deletes the original records from the source database.

**Note:** If archiving data from a single table or Notes form to multiple relational tables (as would be true with the database in the above examples), use Archive activity only for one of the destinations, and Direct Transfer for the rest. Chain them using the Dependent Activity field in the Activity Execution Options so that the Archive executes last. That way, your original data are not deleted until they have all been copied.
7.9.2 Java

A Java activity allows LEI to launch a Java application. The Java application is not restricted to LEI-specific applications.

A Java activity is useful when you need more elaborate control of source and destination during data transfers or when you expect to perform different kinds of data massaging based on certain application-specific conditions.

When developing a Java activity or activity document, consider the following:

- The application must exist on the same system as the LEI server that is designated to run that application.
- A Java JVM must also exist on that system.

7.9.3 Polling

The Polling activity polls a database to see if a specified condition exists, and, if so, executes an activity. Polling activity is useful when you need to execute some action when another action or event occurs or is encountered. The Polling activity can run more than one activity at a time (for example, a replication activity and a direct transfer activity).

7.9.4 Admin Backup

The Admin Backup activity backs up the LEI Administrator database (decsadm.nsf) and, optionally, the LEI Script Vault database (leivlt6.nsf). Admin Backup activity is useful when you want to create a safe backup of the LEI Administrator and associated documents.

7.9.5 Admin Purge Log

The Admin Purge Log activity purges the LEI log database of documents older than a user-specified number of days. Admin Purge Log is useful to control the size of your LEI log database.
Advanced RealTime activities

This chapter describes the Advanced RealTime activities, including the differences between virtual documents and virtual fields. It provides examples of when to use virtual documents versus virtual fields and when to use a combination of both activities. This chapter also discusses virtual attachments and virtual agents and provides an example of how to use them.

This chapter does not discuss the Advanced RealTime function of integrated credentials. For more information about integrated credentials, refer to 2.2.2, “Integrated credentials” on page 28.

**Note:** Before running the examples in this chapter, you should download and install the sample Notes database LeiSampleData.nsf from the Redbooks Web site, as described in Appendix C, “Additional material” on page 251.
8.1 Virtual fields activity

The virtual fields activity (previously known as the LEI real time activity or the DECS activity in Domino) lets you merge relational data into documents stored in Notes. When a user opens the Notes document, certain “virtual” fields that you designate are loaded from a relational database, based on keys read from the Notes document. Also, if you choose to allow it, when the user saves the document, the virtual field values showing on their screen are written back to the relational database. This lets users edit the RDB contents from their Notes client. Any validations imposed by the RDB (such as a non-null requirement for a field) act as validations in the Notes form also — the user can’t save the document unless the virtual fields pass validation.

Virtual fields activities can also monitor creation and deletion of Notes documents, inserting and removing relational rows as Notes documents are created or deleted.

From the point of view of the Notes user, the use of Virtual Fields is essentially transparent — it seems that they are just editing information in a Notes database. There is no need to set up database client software on the user's workstation, or configure anything. All the setup and maintenance of the activity is done centrally, on the Domino server.

Virtual Fields can also be used in conjunction with Domino Web applications running on that server.

Users can create local replicas of databases containing virtualized fields; their local replicas will contain the values of the virtual fields as of the time that document was replicated to them. Users’ local replicas will not access the RDB directly — if they edit the values locally, their changes will be written to the RDB immediately when they replicate their database with the server. The same applies to replicas of the database on other Domino servers.

8.1.1 Usage notes

Frequently, the Notes documents for a Virtual Fields activity are “stub” documents, containing only the field with the relational key value — all other fields are virtualized. It’s up to you to decide which fields are stored only in the Notes database, which are stored only in the relational table, and which are stored in both places.

A form may be associated with virtual fields activities. For instance, a “Customer” document may store in Notes only a CustomerID field, which is used to bring in the customer's address, main phone number, Web site URL, and so on, including the employee ID number of their main sales rep within your company. This SalesRepID may in turn be used as a key for a second Virtual Fields activity, which loads the sales rep's name and phone number.

In such a case, you must use the monitor order of the Virtual Fields activities to make sure that the activities run in order; if the second Virtual Fields activity runs first, the key value it needs will not have been loaded yet.

In the same way, Virtual Fields may also be used in combination with Virtual Documents activities (8.6, “Virtual documents activity” on page 172). Use the monitor order to make the Virtual Documents activity run first.

Virtual fields functionality for Notes/Domino 6 includes the following features not available in the previous versions of the real time activity function:

- Support for computed subforms
- Options for new line delimiters
- Integrated credentials
- Support for procedure return parameters following insert and update operations (DB2 only)

Output from write operations allows results of write operations to be returned to fields in the Domino document being monitored following insert and update operations. One activity can monitor all forms in a database, making it possible to perform a generic lookup of a given field.

A common use of this capability is to retrieve the value of a unique ID field generated by the database server when a document is created using Notes or Domino.

For step-by-step virtual fields examples, refer to 8.3, “Example 1: Virtual Fields activity” on page 158, and the sections that follow it.

### 8.1.2 Setting up the virtual fields activity

The following elements are required to setup the virtual fields activity:

- A connection document to the external database. See Chapter 6, “Connection documents” on page 73 for more details.

  Virtual Documents activities operate directly on the Notes databases, not through the Notes Connector. Therefore, no Notes connection document is needed.

- One or more virtual fields activity document where you specify which Domino form to monitor and map the Domino fields to the fields of the external table(s). You need one Virtual Fields activity document for each relational table or view you want to connect to.

- With multiple virtual fields activity documents, you need to specify the “monitor order”, that is, the order in which they will execute, making sure that each activity runs after its key fields are loaded into the document.

  If you use more than one virtual fields activity for a single Domino form, you may specify the order in which the virtual fields activities are executed. The monitor order also enables you to use multiple activities which connect to different tables and use the value found by the first virtual fields activity as a key for subsequent virtual fields activities.

Once all the required elements are in place, the virtual fields activity must be running for users to access the external data. In addition, you must have key documents, which are “real” Domino documents containing the key fields. Fields not actually stored in Notes are not available for use in Domino views; because of this, you may choose to “retain” copies of some virtual fields in the Notes document.

**Note:** “Retained” fields displayed in views are not automatically kept up to date with the relational data. You must open the document to see the latest information. If you need views kept up to date, see 9.8.2, “LC LSX Example 2: Update Virtual Fields in Notes views” on page 226.

In most cases, the relational table already exists, so you will need to create key documents in Notes to correspond to the existing rows. These documents are not automatically created. While the activity is not running, use the Initialize Keys action from the activity form to populate the Domino database with one key document for each external system record. Only the key fields and any that the activity specifies to “retain” will be populated.

**Important:** “Initialize Keys” does a mass copy of key information into Notes, without checking whether the key documents already exist. You should not use this action with a Notes database that already contains key documents.
To incorporate new data that is added to the external data from outside Domino after the keys are already initialized, refer to 8.1.4, “Keeping Virtual Field key documents up to date” on page 154.

Occasionally, when you set up the activity, the data already exists in Notes and you want to create rows in the relational table. You can do this with a Direct Transfer activity, as described in 7.3, “Direct Transfer Activity” on page 111. But here's a way that you may find easier: on the Virtual Fields activity form, set the Missing External Records option to “Create Record”. Start the activity. Then, run a simple agent such as the following to open each record:

```
Example 8-1 Agent to initialize Virtual Fields relational table
REM "Run on all documents in database or view.";
SELECT Form = "yourform";
```

By reading each document to evaluate the selection formula, this agent will make the Virtual Fields activity try to load fields from the relational database. When it finds there is no corresponding relational row, the activity will create one with a copy of the Notes fields.

### 8.1.3 Event options for virtual fields activity

There are four events that occur in the Domino form that can be monitored by a virtual fields activity:

- **Create**: When a user saves a new Notes document, insert a corresponding row in the relational database.
- **Open**: When a user opens an existing Notes document — either on their screen, or via an agent — read the relational database to supply values for any virtualized fields.
- **Update**: When a user saves changes to an existing Notes document, use a keyed Update to write the modified values into the relational database.
- **Delete**: If the Notes document is deleted, also delete the corresponding row in the relational table.

If the Notes database were being used for reference only, or if these fields were being supplied only for the user's information, not for editing (as with the sales rep name and phone number in the example on page 152), you might want to monitor only the Open event, to make sure the user can't make any changes to the original data.

You can add a Domino formula or stored procedure to further control these events. Refer to the LEI Activities and User Guide for more information.

### 8.1.4 Keeping Virtual Field key documents up to date

A challenge often faced by users of Virtual Fields is how to add new key documents when records are added to the table by processes running outside of Notes.

We do not recommend deleting all the key documents at regular intervals and recreating them manually using the Initialize Keys action in the activity form. Your application will have much better performance if you avoid deleting and recreating documents or modifying them unnecessarily.

Creation of new key documents is best done with a Replication activity (7.7, “Replication activity” on page 130). The replication activity would have the Update and Delete functions disabled (Insert only, in other words), and it would map only the key and “retained” fields.
The tricky part is how to prevent the Virtual Fields activity from interfering with the Replication activity. For instance, Replication detects a new record and inserts a new key document into Notes. The Virtual Fields activity notices a Notes document being created by the replication activity and attempts to insert a new record in the relational database. But the record is already there, which causes either a duplicate record, or an error message in the activity log.

There are a few different approaches you can take to prevent this.

**Program the Virtual Fields activity to not handle the Create event**

If you don't enable the Create event in the Virtual Field document, it will not notice the Replication activity creating new key documents, so all will be well.

The drawback to this is that it prevents Notes users from creating records through Notes. If you don’t plan to allow this, then this technique is best for you.

**Turn off Virtual Fields activity during replication**

If you do need users to be able to create new records into the relational table, then your Virtual Fields activity will have to handle the Create event. In that case, consider how long a delay is acceptable before the key documents get created. If it’s sufficient for your application that the key documents are created once a day, during off hours, then you can program the Virtual Fields activity to shut down between, say, 11:30 PM and 12:30 AM, and schedule the replication to happen during that interval.

The main drawback to this approach is the delay between the creation of relational records and their availability to Notes users. Also, however, it will only work if there’s a time of day when you’re quite sure no Notes users will be working in the database. If someone creates or edits a record while the Virtual Fields activity is turned off, their changes will be lost (or at least, will not be copied to the relational table).

**Make Virtual Fields activity able to identify a replicated record**

This technique is more complex than the approaches described above, but it lets you create new key documents at frequent intervals, while the Virtual Fields activity is running, without causing unnecessary inserts in the relational database.

When you create a Replication activity between a relational database and Notes, you have to select a Notes connection record. The Notes connection form contains fields where you can enter “Data Transformations” (6.3.5, “Data Transformation section” on page 86). In the Notes connection you use for your replication, enter an “Insert” transformation such as the following:

```plaintext
Select @All;
FIELD _LEI_REPLICATION := "1"
```

The presence of the _LEI_REPLICATION field (or a field with some other name you choose) lets you tell a record added by the replication activity, apart from one added by the Notes user.

Next, write a “Filter formula” in the Virtual Fields activity to avoid processing replicated records. The Filter formula field is on the Options / General Options tab of the form:

```plaintext
@NoteID != "NT00000000" | _LEI_REPLICATION != "1"
```

Which means, only process this document if either it’s not being created now, or it’s being created other than by the replication activity.

**Note:** @IsNewDoc doesn’t seem to work in this context, but the @NoteID test does the job just as well.
Using Replication to update “retained” fields
Techniques similar to the above can also be used to manage the updating of “retained” fields in Notes when those change in the relational database. As shown in 8.4.3, “Considerations in storing copies of fields” on page 167, when you use Virtual Fields, the information in a Notes view can be out of synch with the information displayed when you open the document. This happens when the relational data doesn’t match the copies of that data stored in Notes.

If you don’t require live updates during the day, or users cannot edit the Notes documents, then an off-hours replication or a Virtual Fields activity that triggers only on Open, may be better for you.

If you need frequent updates and want to use the “Filter formula” technique described in the previous section, you must do something more to let the Virtual Fields activity tell the difference between an update done by the Replication activity, and an update caused by a user editing a Notes document. You would change your filter formula to the following:

```
_LEI_REPLICATION != "1"
```

This doesn’t test for a newly created document. Then you must change the Notes form to clear the _LEI_REPLICATION field when the user edits the document. This is most easily done by adding a hidden Computed field called _LEI_REPLICATION, and setting its formula to @Unavailable. Note, however, if you use Enforce Form Design in the Notes connection document, this may result in your _LEI_REPLICATION field being cleared during replication.

Alternately, you might use a LotusScript agent, as shown in 9.8.2, “LC LSX Example 2: Update Virtual Fields in Notes views” on page 226, to process updates without needing an extra activity.

8.2 Virtual Attachments

Virtual attachments allow you to use Domino to add file attachments and OLE objects to the external system database through a Domino application. If you use this feature, file attachments in forms you specify will be stored in the external system rather than in your Domino application.

Virtual Attachment is not a separate activity; it’s an option that you can activate as part of a Virtual Documents or Virtual Fields activity.

If you have attachments in Domino documents monitored by a Virtual Documents activity, you cannot save the document unless you're virtualizing attachments. Since the entire Notes document is stored in the RDB, you can't store anything in the document unless there's a place for it in the RDB.

For Virtual Fields, if you enable Virtual Attachments, then all attachments on that form will be virtualized, even if you put them in a rich text field that’s not mapped to a relational field. If you do not enable Virtual Attachments, then all attachments will be stored in the .nsf file, even if you put them in a rich text field that is mapped to a relational field.

The virtual attachments feature uses a separate relational table to store the attachment contents. This table must be in the same database and library as other tables used in the activity. It’s possible to use the same attachments table for different forms and different .nsf files.

The list of required and optional virtual attachment columns is shown in Table 8-1.
Table 8-1  Virtual attachments columns for Virtual Fields activities

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Required or Optional</th>
<th>Data Type and Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIDBID</td>
<td>Required</td>
<td>CHARACTER 16 FIXED</td>
</tr>
<tr>
<td>EIFIELD</td>
<td>Required</td>
<td>INTEGER</td>
</tr>
<tr>
<td>EIFILESIZE</td>
<td>Required</td>
<td>INTEGER</td>
</tr>
<tr>
<td>EIUNID</td>
<td>Optional</td>
<td>CHARACTER 32 FIXED</td>
</tr>
<tr>
<td>EIFILENAME</td>
<td>Optional</td>
<td>CHAR 256 VARYING</td>
</tr>
<tr>
<td>EICONTENTS</td>
<td>Required</td>
<td>BINARY with max size = enough to contain your largest attachment</td>
</tr>
<tr>
<td>EIFILE *</td>
<td>Optional</td>
<td>BINARY(1K)</td>
</tr>
<tr>
<td>foreign key columns *</td>
<td>Optional</td>
<td>same as data columns</td>
</tr>
</tbody>
</table>

* = used in LEI 6.0.3/6.5 only

Refer to the LEI Activities and Users Guide for more details about these columns.

Note: The virtual attachments table must be on the same external system and the same database and library as the external system data being accessed by the activity. The virtual attachments table cannot reside in a different external system or database than the one being used by the virtual fields activity. This is because the same connection document for both virtual fields and virtual attachments is used to access the external system.

For a step-by-step example of using virtual attachments, see 8.3, “Example 1: Virtual Fields activity” on page 158. It may also be helpful to refer to 8.6, “Virtual documents activity” on page 172.

8.2.1 How file attachments work in Notes

To understand what virtual attachments do, it helps to know how attachments are stored in Notes when you’re not using LEI.

There are two ways to add an attachment to a Notes document. If it's done from the Notes client or using LotusScript code, the attachment icon appears in a rich text field. If from a Domino Web application, the attachment icon may appear in a rich text field, but more often it’s added as a “V2-style” attachment, displayed below a horizontal line at the end of the Notes document.

The attachment information is stored in three parts:

- A “stored object” record in the Notes database, which exists separate from the document. This contains the contents of the document in binary form. It may be compressed using LZ1 or other encoding scheme.
- $FILE item in the Notes document. There will be one $FILE item for each attachment stored in Notes. OLE Objects may have multiple $FILE objects. The $FILE contains information that’s needed to make use of the information in the stored object record, including the filename, original file size, ID of the stored object, and compression format. If the $FILE item is lost, there is no way to access the contents of the stored object.
Icon in the Rich Text field (optional). The rich text “CD” record for a file attachment contains only the filename and icon graphic. If you lose the rich text contents, you can still access the attachment, since it will then appear as a V2-style attachment icon.

You can map a rich text field to a binary field in a Virtual Documents activity (as we will in 8.3, “Example 1: Virtual Fields activity” on page 158). However, the binary data of the rich text includes only the icon and name of any attachments. If you also want to virtualize the attachment contents, you must activate the Virtual Attachment functionality, which is new in LEI 6.

Virtual Attachments are not a separate LEI activity; they are an option of the Virtual Fields and Virtual Documents activities. The attachments are not stored in the same table as the rest of the virtualized fields; you must create a separate table for them. LEI provides a utility to create this table automatically from the Activity form. The table must be in the same database as the table containing the virtual field data.

For some applications, you might want to virtualize only the attachments and keep all the fields in Notes. This lets you use your relational database as a mass data store for large attachments, so that your Notes database file stays relatively small.

Virtual attachments do replicate to other servers and to local replicas. Unless you set the replication options to truncate large documents, any replica of the database will contain a copy of each attachment.

**Important:** The $FILE item is not virtualized by a Virtual Fields activity. This item must be available to access the document contents, and it is stored only in the “key” document. Therefore, if you delete the “key” document in Notes, you permanently lose access to your attachment — if the activity is running and you have enabled the Delete event, it will be deleted, but even if it’s not deleted, the information needed to access it is gone.

Of course, the same situation is true of documents stored entirely in Notes.

With a Virtual Documents activity, the $FILE item is stored in the RDB, as part of the EINOTEPROPS field. As with Virtual Fields, if this information is deleted, there is no easy way to recover the file attachment.

### 8.2.2 Virtual Attachment “Resync” for Virtual Documents

Later versions of LEI, beginning with 6.0.3/6.5, contain a utility to retrieve file attachments whose key documents have been lost. To use this utility, you need to have created your attachment table with extra fields to contain an extra copy of the $FILE item and the data keys, so that LEI can match the attachment record with a new Notes document or Virtual Document. Refer to the 6.5 release notes and whitepapers on the Lotus Support Web site for details.

### 8.3 Example 1: Virtual Fields activity

For our example, we’ll use the same relational tables used in the previous examples (AGENTS and AGENTEQUIP), and a Notes form that’s nearly the same as the form used in the previous examples, but with a different name. In this example, we “virtualize” the fields on that form.

There’s one important difference between the new form and the original Agent Records form: the new form uses hide formulas to make it impossible to edit the AgentID field after a new
document has been saved for the first time. Since the Virtual Documents activity does a keyed update of the relational data when the user saves changes to the record, the user editing the key fields will either make the update fail, or make it update the wrong record.

As we’ll see when we review the activity options, there’s an option to control how LEI behaves when the user tries to save a document with changes to its key fields. The default behavior will not permit this, so it’s not necessary to also design the form to make editing them impossible, as we have done here. We did it for ease of use; why let the user make edits that we know will be rejected anyway?

8.3.1 Populate the data in the relational tables

This example assumes that you already have the relational tables populated with data, as was done in 7.4, “Example 2: Using Direct Transfer” on page 112.

In most cases, your relational database will already contain the data you want to virtualize, so this step should not be necessary.

8.3.2 Create a Connection Document

Normally you would have to create a connection to your relational database. For this example, we can reuse the “Sample DB” connection document you created for the Direct Transfer activity.

8.3.3 Create the Activity Document

From the LEI Administrator main screen, use the action Add Activity -> Virtual Fields (RealTime Notes).

Fill in the fields as described below. Since this is the first Virtual Fields activity we’ve done, all fields are described briefly below even when we’re using the default value.

**Fill in the main section of the Activity Document**

In the section of the document above the Options tabs, enter field values as shown in Figure 8-1 and described below.
Virtual Fields Activity

Identification

Name: **VF Agent Records**

Domino Application

- **Domino Server**: BoboBy/Magic
- **Database**: Redbook Source Data
- **Name**: demoLEISampleData.nsf
- **Form**: Agent Record VF
- **Enable Subforms**: Use Computed Subforms

Lotus Connection

- **Data Source**: Sample DB
- **Table**: AGENTS

Mapping

- **Key(s)**: AgentID (Integer)
- **Field(s)**: AgentName (Text), Mission (Text), Photo (Binary), Specialties (Binary)

Events

- **Document Events to Monitor**: Create, Open, Update, Delete

**Figure 8-1 Activity Document — VF Agent Records**

1. **Name**: Enter a unique name identifying the activity; in this case, VF Agent Records.
2. **Domino Application**: use the selection button to select the sample database, demoLEISampleData.nsf.
3. **You will be prompted to select a form**: select Agent Record VF.
4. **Use Computed Subforms**: do not enable, since there are no computed subforms on the form. If you check this box, you can enter a list of subform names, similar to the Notes connection document (page 79).
5. **Lotus Connection**: Use the selection button to select the Sample DB connection. You will be prompted to select a table; select the AGENTS table. If you instead selected the Sample DB AGENTS connection, you would not be prompted to select a table, since that connection document already refers to a specific table.
6. **Click the Mapping button and map the fields as shown above and in Figure 8-2.**
7. Events: Select Create, Open, Update, and Delete. In cases such as this one, where there's a single Notes document corresponding to a single row in a relational table, you will generally want the Virtual Fields activity to intercept all events. The meanings of the selections are described in 8.1.3, “Event options for virtual fields activity” on page 154.

**Fill in the Options tabs of the Activity document**

Select options for the Virtual Documents activity as described below.
**General Options**

<table>
<thead>
<tr>
<th>Options</th>
<th>Event Options</th>
<th>General Options</th>
<th>Multi Value Data</th>
<th>Integrated Credentials</th>
<th>Virtual Attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor Order</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max Connections</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Form Override</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter Formula</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Integrity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trim Trailing Spaces</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Line Delimiter for Connection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 8-3  General Options for VF Agent Records*

- **Monitor Order:** use the default value, 1. This activity will execute first each time an event occurs with the Agent Record VF form. Later we'll add another Virtual Fields activity for the same form, with a higher number.

- **Max Connections:** use the default value, 1. Refer to 2.3.1, “LEI considerations” on page 32 for an explanation of this setting.

- **Form Override:** Do not check. This option makes the activity apply to all forms in the Notes application, not just the selected form.

- **Filter Formula:** Leave blank. If you wish, you can enter a Notes macro formula here to restrict which Notes documents are handled by the Virtual Fields activity. Do not use SELECTI just write a formula that evaluates to a true/false value, for example, `Status = "New" : "Active"`

- **Data Integrity:** use the default “Allow precision loss”. You would normally decide which option is appropriate based on the nature of your data. Keep in mind that the data movement here is two-way — the activity reads data from the RDB when you open the document, and writes it back to the RDB when you save. Therefore, if there's precision loss happening during the read, that degree of precision will be removed from the RDB during the update!

- **Trim Trailing Spaces:** If the RDB contains fixed-length text fields, it's usually nicer to remove the trailing spaces on all fields that don't need to be compared to the corresponding RDB fields for exact equality. Use the default choice, Trim, for this example.

- **New Line Delimiter for Connection:** this option is used when Updating or Inserting into the RDB. If the Notes field contains a newline character, how should this be represented in the RDB? For this example use the default, Linefeed and Carriage Return.

- **Data Storage:** By default, the Notes fields that you mapped to RDB fields will not be stored in Notes (except for the key fields, which LEI must have to look up the others). This option lets you store copies of the mapped fields AgentName, Mission, Photo and Specialties, in the Notes document. For the moment, use the default “Remove” choice, and we'll see the effect of this decision later.
Multi-value Options

This option lets you combine multiple rows of relational data into multi-valued fields in Notes. In 7.4.5, “Collapse/Expand Metaconnection” on page 121, we did this with a separate metaconnection document. This option lets you do the same thing without an extra document. See 8.5.2, “Create a secondary Virtual Fields Activity” on page 168 for an example of this option in use.

Integrated Credentials Options

Refer to 2.2.2, “Integrated credentials” on page 28.

Virtual Attachment Options

Enable “Virtualize Attachments”. Click the Create Virtual Attachment Table button. This will automatically create the separate table that will be used to store your attachments. Fill in the dialog as shown in Figure 8-7 and described below.

Name: Enter the table name for the table to be created, in this case yourschema.AGENTATTACH.
Will the table contain attachments from more than one Notes database? Select “No.” You can have a single attachment table and use it with multiple Notes databases; LEI uses the EIDBID column to tell which is which.

**Note:** Version 6.0.3 and later create extra columns in the attachment table to allow for recovery in case the key document is deleted. The “old style” of attachment table will still be usable, but you’ll get a warning in the log and will be unable to use the new “Resync Attachments” function.

Max attachment size (DB2 only). This portion of the dialog has a different default value and appearance in Version 6.0.3 and later, as shown in Figure 8-8. Even in earlier versions, do not leave this field blank; for future compatibility, we recommend entering a reasonable size limit here rather than using the default “maximum” size. Changes to the DB2 connector in Version 6.0.3 have the system allocate the maximum amount of memory for every fetch, even though the actual contents may be much smaller.

![Max attachment size field](image)

**Figure 8-8** DB2 Max Attachment Size field for Version 6.0.3/6.5 and later.

Login Information: Uncheck this checkbox if you need to enter alternate credentials than those given in the connection document. Since this tool creates a table, it will need to connect to the database using an ID with enough authority to do that.

Click OK.

**Save Activity and Initialize Keys**

- Press Ctrl+S or use the File -> Save menu to save your changes.
- Use the Initialize Keys button from the activity form. This will display the informational dialog shown in Figure 8-9.

![Informational message from Initialize Keys action](image)

**Figure 8-9** Informational message from Initialize Keys action

Notice that the dialog says only that the AgentID field will be created in the key documents. With the Data Storage option selected above, only AgentID will be stored in Notes; all the other fields that we mapped to the AGENTS table are virtual.

If the form contains other fields that were not mapped, they will be stored in Notes, however. For instance, because this activity doesn’t map EquipDate, EquipStatus and so on, they will be stored in the Notes document. We haven’t done anything (yet) to create a relationship between this form and the AGENTEQUIP table.

- In this case, we do want to create the key documents

- Use the Save and Close action to save your changes and return to the LEI Administrator main screen.
Highlight the VF Agent Records activity in a view and use the Start Activity action.

Wait, pressing F9 periodically, until the radio tower icon shows in the view. This indicates that the activity is running.

Test the Virtual Fields Activity

Open the sample database in your Notes client and go to the view Virtual Fields \ by Agent ID. The result is shown in Figure 8-10 (with the document preview pane open to show both a view and a document).

![Figure 8-10 Results of first Virtual Fields activity](image)

The view contains columns for the name and mission, but although we can see these values when we open a document, only the AgentID field is displayed in the view. Notes views will only display fields that are stored in the Notes document. Since we chose the Data Storage option to store only key fields in Notes, only the key field AgentID is available for use in views.

Also notice that the Equipment Table is not filled in. We’ll fetch that information with a separate Virtual Fields activity in a later example. But first, let’s fix this view.

At this point, you may want to edit some documents, or use the New VF Agent action to create a new document and see that the data appears in your RDB. Even new documents you create, will not display the virtual field values in the view.
8.4 Example 2: Adjust Virtual Fields to Store Additional Fields

In this exercise, we change the activity created in 8.3, “Example 1: Virtual Fields activity” on page 158, so that certain fields are available for use in views.

8.4.1 Stop Activity and update Activity document

- Return to the LEI Administrator main screen, highlight the VF Agent Records activity, and use the Stop Activity action to shut it down.
- Press F9 periodically until the suitcase icon is displayed in the view, indicating the activity has stopped.
- Use menu Actions -> Edit Document.
- Go to the Options / General Options tab, and change the Data Storage option to “Leave selected real-time fields in documents.”
- This selection makes a “Saved Fields” option appear. Use the selection button to select the AgentName and Mission field, as shown in Figure 8-11. These two fields will remain in the Notes document when it's saved. Notice that the dialog lists only mapped fields that are not keys, since these are the only ones that are not stored in the Notes document otherwise.

![Figure 8-11 Selecting fields to store in Notes](image)

- Save and Close.

8.4.2 Update documents to contain the new fields

- From the LEI Administrator main screen, highlight the VF Agent Records activity, and use the Start Activity action. As before, wait for the activity to start.
- Return to the sample database and go to the Virtual Fields \ by Agent ID view. You may be disappointed at first to find that there is no change. That's because copies of the virtual fields are stored in the Notes document only when the document is saved or created.
- Edit and save a document. Now, you should be able to see the name and mission in the view.
- To update every document with the new fields, use the “Refresh documents” action in the Virtual Fields \ by Agent ID view; this action uses the formula @Command([ToolsRefreshAllDocs]), which edits and saves each document in the view. The view should now be up to date.
8.4.3 Considerations in storing copies of fields

When you store copies of fields in the Notes document, there's a tradeoff involved: you get to use the fields in views, but anytime you have two copies of the same data, you have to worry about keeping them in synch. For instance, suppose I update the agent's mission in the relational database. Although an up to date value is loaded when I open the document on screen, the view continues to display the old value until the Notes document is edited and saved (or “refreshed” as in the previous section). Figure 8-12 shows the situation. The ASSIGNMENT field for Agent 44 has been updated in the AGENTS table, but that change is visible in Notes only in the document window, not in the view.

To correct this situation, you could use the Refresh Documents action periodically, or you could write a scheduled agent to regularly “touch” each document. However, it's inefficient to update every document when perhaps only a few rows of relational data, or none, have changed. Especially in views with a lot of documents, this can cause long delays when views are opened and excessive replication times.

It's more efficient to use a Replication activity to synchronize the data. However, the Replication activity won't work while the Virtual Fields activity is running; so, you would need to use the scheduling options on the two activities to turn off Virtual Fields briefly while you run the Replication.

Alternatively, you could use a LotusScript agent to scan for differences and correct them, as shown in 9.8.2, “LC LSX Example 2: Update Virtual Fields in Notes views” on page 226. This type of code can run while the activity is active.

The advantage of using a Replication activity is that it also creates new key documents when data rows are added, which this agent does not do.

8.5 Example 3: Virtual Fields activity with multi-value fields

The next example shows how to create a second Virtual Fields activity on top of the exiting activity. This second activity, like the replication activity in 7.8.9, “Create a secondary Replication activity” on page 144, converts between multivalue data in the Notes document,
Implementing IBM Lotus Enterprise Integrator 6

and multiple rows in a secondary table. However, this time it is not necessary to create a Collapse/Expand metaconnection to do this — the functionality is available directly from the Virtual Fields activity form.

Restriction: A secondary Virtual Fields activity does not support the Update event; trying to update the record results in the message “Error: Field count mismatch” in the activity log. This is true whether or not the activity uses multi-valued data.

For applications that do not use multi-values, a workaround is to use a relational view as the source data, rather than a table, so that all updates can be done by a single Virtual Fields activity.

Alternately, you could use a Replication activity instead of Virtual Fields.

8.5.1 Create a connection document to the AGENTEQUIP table

For this exercise, we will reuse the connection Sample DB AGENTEQUIP created for a previous example (7.4.4, “Copy the RDB Connection” on page 121).

8.5.2 Create a secondary Virtual Fields Activity

From the LEI Administrator main screen, use Add Activity -> Virtual Fields (RealTime Notes). Fill in the resulting form as shown in Figure 8-13 and described below.

Figure 8-13 Activity VF Agent Equipment

1. Name: Enter a name that uniquely identifies the activity; in this case, VF Agent Equipment.
2. Use the Domino Application selection button to select demo\LeiSampleData.nsf.
3. You will be prompted to select a form; select Agent Record VF.

4. Use the Lotus Connection selection button to select the Sample DB AGENTEQUIP connection. The table name AGENTEQUIP will be filled in automatically.

5. Use the Mapping button to map Notes fields to relational fields as shown.

6. Document Events to Monitor: Select Create, Open, and Delete. As noted earlier, since this is the second Virtual Fields activity on the same form, Update is not supported.

**Activity “Options” Section**

There are several options and their values explained here.

**General Options**

<table>
<thead>
<tr>
<th>Options</th>
<th>Event Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Options</td>
<td>Multi-Value Data</td>
</tr>
<tr>
<td>Monitor Order</td>
<td>2</td>
</tr>
<tr>
<td>Max Connections</td>
<td>1</td>
</tr>
<tr>
<td>Form Overwrite</td>
<td>Monitor any form</td>
</tr>
<tr>
<td>Filter Formula</td>
<td></td>
</tr>
<tr>
<td>Data Integrity</td>
<td>Prevent both precision and data loss</td>
</tr>
<tr>
<td>Trim Trailing Spaces</td>
<td>Trim spaces on all non-key fields</td>
</tr>
<tr>
<td>New Line Delimiter for Connection</td>
<td>Linefeed and Carriage Return</td>
</tr>
<tr>
<td>Data Storage</td>
<td>Remove all real-time fields from documents</td>
</tr>
</tbody>
</table>

*Figure 8-14 Options - General for VF Agent Equipment activity*

- Monitor Order: Enter 2, so that this activity will run after VF Agent Records (which has a Monitor Order of 1). Virtual Fields activities handle one document at a time as the document is requested from the server, so when you open a document that uses the Agent Record VF form, first the VF Agent Records will supply the name, mission, and skills, then this activity will add the equipment.

  In many cases, it might not matter which activity executes first. The monitor order becomes especially important when one activity fetches a value that another activity requires as a key or selection criterion.

  In this example, the monitor order is important because only the activity that executes first can monitor the Update event.

- Other fields: use default values.

**Multi-Value Data Options**

This tab lets you use the functionality of a collapse-expand metaconnector (page 98) without having to create a separate document. The key fields you select in the mapping dialog will be used as the “grouping keys”. The other values you would supply for the metaconnection are entered in this section.

Fill in the fields as shown in Figure 8-15 and described below.
Use multi-value data fields: enable this option. This will convert multi-value data fields in the Notes document into multiple rows in the relational table and vice-versa.

- Subkey fields: use this field when there is additional data in the relational table that you’re not mapping to Notes fields, and you need to make sure that updates to the relational data go into the same rows that they were read from. In this case, we’re not going to do any updates, and anyway we’re mapping all the relational fields, so use the default value, <none>

- Sorting: enable Sort multi-value data. This lets you sort the multivalue rows by a selected relational field.

- Sorting fields: use the selection button to select the DATEISSUED field (the selection dialog is shown above). The rows read from the relational table will be sorted by date.

- Text order: Not important since the sorting field is not text. Use the default, Binary.

- Direction: use the default, Ascending, to sort the rows in ascending date order.

**Integrated Credentials**

Use the default values.

**Virtual Attachments**

Do not enable virtual attachments, since the VF Agent Records activity is already handling that.

**Event Options Section**

Use default values except as noted below.

**Open Options**

Since there’s a possibility that an agent has no equipment, it might happen that when we open an agent document, there are no equipment records that match that agent ID. We don’t consider this an error situation.
Chapter 8. Advanced RealTime activities

8.5.3 Test the new activity

- From the LEI Administrator main screen, locate the VF Agent Records virtual fields activity and make sure that it’s running. If not, use Start Activity action.

- Locate the new VF Agent Equipment activity. Highlight this document and use the Start Activity action.

  **Note:** It doesn’t really matter what order you start these activities, and if you want you can also select and start them both at the same time (use spacebar or click in the left view margin to mark a document selected, with a checkmark symbol). All activities that operate on a particular form should be running before anyone tries to access any of the documents that use that form.

- Wait until both activities show the radio tower icon, indicating that they’re active. Use F9 to refresh the view occasionally.

- Open the sample database and go to the view Virtual Fields \ by Agent ID.

- Open documents in this view and observe that you can now see their list of equipment, sorted by date. This form doesn’t let you edit the equipment list in an existing document; since the activity doesn’t handle the Update event, any changes that you did make would be overwritten the next time the document was opened.

- Use the **New VF Agent** action to create a new agent record. Fill in the editable fields, then click the Add Item button to assign the agent some equipment. Save the new document.

You might notice as you play around with this, that you get an error if you create an agent with no equipment. This could happen if the equipment table is set to disallow null values in some of its columns. Since each of the Equip fields would actually contain the value “” (empty string) in that case, LEI will try to insert one row into AGENTEQUIP with only the agent ID filled in.

There are a few ways you could deal with this: first, you might decide that you can have null values in the table, and change the table design to allow it. Or, you might add a validation to the Notes form to prevent saving an agent with no equipment. Or, you could create two separate Virtual Fields activities, one for the Create event and one for the Open and Delete events. In the activity to handle Create, enter a filter formula (under General Options) to prevent the activity handling these cases (for example, SELECT EquipName != ‘’).
8.6 Virtual documents activity

The virtual documents activity lets Domino users open, create, update, or delete external relational data directly and transparently through a Lotus Notes/Domino application. Even though the data resides in an external database, from the end users’ point of view, it is as if they are working solely in a Domino database; they may not be aware of the presence of an external database.

The user's workstation does not need to have relational database connectivity software to use Virtual Documents (not even ODBC). The database client must be installed on the Domino/LEI server. Network access to the external data source is handled by the Domino server, which contains LEI connectivity software for each supported database type.

For step-by-step virtual documents examples, refer to 8.9, “Example 4: Using a virtual documents activity” on page 182.

8.6.1 How does the virtual documents activity work?

A Virtual Documents activity takes all the information that would normally be stored in Notes for a particular document, and stores it in relational tables instead. That means not only the fields that “map” to a particular form fields, but also all the system information such as last modification time, list of fonts used in the document, and so on.

The following elements are required to setup a virtual documents activity:

- A connection document to the external database. See Chapter 6, “Connection documents” on page 73 for more details.
- A Virtual Documents activity document which specifies the Domino form that will be monitored by the activity and the mapping between the Domino form fields and the external system table columns. You also need to make sure that the external system contains a table with four special columns that map to Domino generated control fields. Refer to 8.6.3, “External system metadata for virtual documents activities” on page 174 for more details on these four special columns.

Once all the required elements are in place, the virtual documents activity must be running in order for you to work with the external data.

With the virtual documents activity running, you can create a new document in Domino and save it. A new record will be created in the external system table containing the data you entered along with the extra information needed by Domino to treat this record as if it were a “normal” Domino document. Then, when you open the document later in Domino, it will be displayed correctly (including any database links, document links, attachments, and so on). Views containing view selection formulas will correctly display your virtual documents as required by the selection formula.

You can also open, update, and delete documents as long as the virtual documents activity is running. If the activity is no longer running, and you attempt to open a document from a view, you will be prompted with an error message. Refreshing the view while a virtual documents activity is not running causes all of the non-native document information to disappear from the view.

If the external data is updated by a non-Domino application, the changes will not be immediately effective in Domino until the Domino view is refreshed. Use function key F9 to refresh or Shift+F9 to rebuild the view if the data is altered in this fashion.
The same is true for newly created external data. If another external system client creates records, the newly created records are automatically virtualized if you enable the Virtualize External System Data option in the General Options section of the virtual documents activity document and will appear in the view when it is refreshed.

Deletions and insertions should be performed with the Domino client through the virtual documents activity instead of through non-Domino clients and applications. This helps ensure the integrity of other Domino processes, such as replication, as it concerns virtual documents. For more information on this topic, refer to 1.3.4, “Current limitations with using LEI 6” on page 10.

### 8.6.2 Differences between a regular Notes document and a virtual document

The behavior of normal and virtual documents is very similar, but there are some differences that you should be aware of.

**Fields that are not mapped are not stored**

Notes’ .nsf file is a free-form database format that lets you create any field you want. Virtual Documents, on the other hand, have to store their data into a fixed set of columns. If there is no column to store a field into, the field cannot be stored. Any value the user entered into the field is lost.

**$Fields are stored in system properties area (EINOTEPROPS column)**

An exception to the above rule are system fields such as $UpdatedBy, $Readers and $Revisions. These fields are stored in their exact binary form in the EINOTEPROPS column of the relational table.

Since they are stored in binary form, the values of these fields are not available to outside processes. However, Notes knows how to read them.

**Tip:** All fields whose names begin with “$” will be stored in EINOTEPROPS, even if they are not “regular” system fields. This gives you a way to add information to the virtual document that’s not associated with a column — just use a “$” at the beginning of your fieldname.

Note, however, that in some database systems you must specify a size limit for the EINOTEPROPS area. This limit must include enough space for the system information plus any information you add.

The “Create External Key Table” button on the Virtual Document activity form contains a calculator for figuring out how much space you’ll need for the system fields — add your extra information to that.

**Multivalue fields are not directly supported**

There’s some difference in the way multivalue fields work between 6.0.1, 6.0.2, and 6.0.3/6.5. In none of these versions do they work exactly like Notes multivalue fields. In all these versions, when you write them into a text column, they are stored as delimited strings. When you access the Notes document later, they are not split into multiple values and converted back to their original datatype, however. In 6.0.2 and earlier, they remain single strings with
In 6.0.3/6.5, they are returned to Notes as rich text values. This change is a bug, not the intended functionality.

This is a problem especially with Reader and Author fields. There's a great deal of difference between the access allowed by a Readers field containing the two-element list “Accounting” and “[Admin]”, for instance, and the single string value “Accounting; [Admin]”.

Until this is fixed, you can get around the problem by giving your multivalue fields names that begin with “$”, as described in the previous section. This will force LEI to store them in the EINOTEPROPS field in their exact binary form, and they will not get messed up by the multivalue processing.

Do not map these “$” fields to a relational column. If external applications need to refer to the values, you can create a separate Computed field with a copy of the values, and map that field to a data column. Updating the copied value in the table will of course not affect the value of the original “$” field in Notes, however.

The other way to deal with a multivalue field, is to store it in a separate table with a many-to-one relationship to the original data table. A Virtual Fields activity is used to read and update the related data. This is the preferred technique because that's how relational databases normally store multivalue information.

8.9, “Example 4: Using a virtual documents activity” on page 182 contains an example of both techniques.

### 8.6.3 External system metadata for virtual documents activities

Unlike the virtual fields activity, a virtual documents activity does not require key documents. However, it does require a table in the external system with four special columns that map to Domino-generated control fields. There are two ways to accomplish this:

- **Integrated key table**
  
  Add the four control data columns (described below) to the external system data table directly. This is the table that will be mapped to the Domino form in the virtual documents activity. For maximum performance, always index the data table key columns (see 8.6.5, “Indexing” on page 177).

- **External key table**
  
  Create a separate table, called an external key table, on the external system that contains only the four control data columns plus a foreign key field(s) to map one-on-one to the actual data table. You can use the table creation utility button in the virtual documents activity document to automatically create this separate table (external key table). 8.9, “Example 4: Using a virtual documents activity” on page 182 provides step-by-step instructions on how to do this.

**Note:** The foreign key columns in the external key table should not be set to enforce unique values, even if they are unique in the data table. Duplicate values may appear in the key table in the normal course of operation.

If you use the external key method, specify the name of the external key table in the virtual documents activity form and the key(s) used to map the key table to the actual data table. These keys can be made up of a combination of fields. The advantage to this method is that you do not need to alter an existing data table.

When using an external key table to store the Domino control fields, the key table must reside in the same database and library, and have the same user access rights as the data table.
For a discussion about the performance of adding the control data columns to the data table directly versus creating the external key table, refer to 2.3, “Performance considerations” on page 32.

With either method, the additional four control data columns are automatically mapped to corresponding internal Domino fields when the virtual documents activity is created. You do not need to map them in the mapping section of the virtual documents activity document. They do not appear when you enable manual mapping. There is nothing you have to do to ensure their being mapped other than to be certain that the names and data types are correct.

The four additional columns required are described in Table 8-2.

Table 8-2  Virtual Documents control data columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EINOTEID</td>
<td>INTEGER (32-bit)</td>
<td>Defaults to zero. This field must not be NULL. This field contains the note id assigned by the Domino server when the document is created. A value of zero here means that the row was created by an external process (not Notes) and has not yet been “virtualized” into a Notes document.</td>
</tr>
<tr>
<td>EIUNID</td>
<td>CHARACTER 32 fixed</td>
<td>Contains the unique hexadecimal id assigned to the document by the Domino server.</td>
</tr>
<tr>
<td>EIMODIFIED</td>
<td>Date/time or DB2 TIMESTAMP</td>
<td>A timestamp that tracks the last time this document was modified. This time comes from the Domino server if the modification happened in Notes/Domino; it can also be assigned by outside applications.</td>
</tr>
<tr>
<td>EINOTEPROPS</td>
<td>BLOB or DB2 VARCHAR for bit data</td>
<td>Contains the header properties of the Notes document and any Notes system fields not stored in the above columns, including Form field, $Revisions, and so on.</td>
</tr>
</tbody>
</table>

Important: These four column names must not be altered! The external system column names, whether added to the existing data table or created in an external key table, must match these names exactly and must be of the appropriate data types.

For more information about these four Domino control data columns, refer to the LEI Activities and User Guide (leidoc.nsf). This guide is available in the /help directory of the Domino server’s data directory where your LEI server is installed. You can also download it from the Lotus Developer Domain Documentation Library at the Web address of:

http://www.lotus.com/ldd/notesua.nsf

Considerations on creating an external key table

When creating virtual documents with the external key table option, the following considerations apply:

- The Virtual Documents activity form contains a control, “Create External Key Table”, that will create this table for you. You will have to answer some questions about your application (depending on what relational database you’re using).
- There is no support currently to automatically add the four required columns (EINOTEID, EIUNID, EIMODIFIED, and EINOTEPROPS) to an existing table from the virtual documents activity form. If you would prefer to add the fields to an existing table, you could...
create the external table first, then use your relational database configuration tool to copy fields to the table you want them to appear in. Notice which columns are indexed in the external key table and retain those indexes when you copy the fields.

Alternatively, you can create a new table with the new columns and then use an LEI direct transfer activity to copy the data from the old table to the new table.

- You must enable writeback to modify a the foreign key that relates the external key table to your data. Modification of non-key fields does not require writeback. Alternatively, you can set up the activity to delete the record and then recreate it with a new key value.
- The data table and the external key table must reside on the same external system data source (same connector/server/database/userid/password, different table name).
- If rows are deleted from the data table by another application and the corresponding entry in the ID table is not deleted, you will receive “unable to retrieve external record” errors when you attempt to access that virtual document. Refer to 8.6.6, “Deletion involving virtual documents” on page 177 for more information.
- If there are data rows with duplicate keys, only one ID table entry will be created. Only the first data row will be retrieved on open, and both records will be modified on delete and update.
- For maximum performance, always index the data table key columns (see 8.6.5, “Indexing” on page 177).
- Regardless of whether the corresponding key columns in the data table may be constrained to not allow duplicate values, do not impose that setting on the key columns in the external key table. In the external key table, key columns should be set to allow duplicate values.

8.6.4 Virtualizing and revirtualizing

When a virtual documents activity is first launched, and periodically thereafter if you program the activity to do so, LEI scans the documents in the data and key tables, looking for data rows for which Notes system information has not yet been generated. In an internal key table, a row is considered to need “virtualizing” if EINOTEID = 0. With an external key table, virtualization occurs if there is no key table row matching the primary keys.

If the .nsf file gets damaged and you have to restore it from backup, it may be necessary to “revirtualize” the virtual documents to get them to show up in the views again. Revirtualizing should not be considered a routine troubleshooting step. It's more of an emergency measure to recover your virtual documents in cases where they're not showing up in Notes and you've tried everything else.

To flag a document for revirtualization, you can either set the EINOTEID value to zero, or if you're using an external key table, delete the rows containing the primary keys from the rows whose documents need to be regenerated.

Note: When using a virtual documents activity that also uses virtual attachments, you must only virtualize each row once. If you reset the EINOTEID back to zero and there are virtual attachments present, they will become inaccessible through Notes. There will be no way to retrieve them from the external system database.

Version 6.0.3/6.5 and later lifts this restriction; virtual attachments (and other system fields such as $Fonts) are preserved provided you revirtualize into the same .nsf file as previously. However, if using an external key table you must not delete rows, as this erases information needed to locate the attachments; instead, just set EINOTEID = 0 in those rows.
8.6.5 Indexing

Indexing is an important requirement for proper virtual documents activity operation, as well as for proper operation of virtual attachments when they are used with virtual documents activities.

For virtual documents, we strongly recommend that you index the external system data columns listed in Table 8-3, for maximum performance.

The “Create External Key Table” tool will automatically create these indexes.

Table 8-3  Data columns to index

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Required or Optional</th>
<th>When to Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>EINOTEID</td>
<td>Required</td>
<td>Always</td>
</tr>
<tr>
<td>EIMODIFIED</td>
<td>Required</td>
<td>Always</td>
</tr>
<tr>
<td>EIUNID</td>
<td>Optional</td>
<td>When supporting Web applications through Domino HTTP server.</td>
</tr>
<tr>
<td>Foreign key</td>
<td>Required</td>
<td>Always when using external key tables.</td>
</tr>
</tbody>
</table>

8.6.6 Deletion involving virtual documents

In a regular Notes application, when a document is deleted, a marker called a deletion stub remains behind. This special document is needed during replication with other Domino servers and local client replicas, so that the server can tell the difference between a document that's been deleted and one that needs to be copied from another replica. After an amount of time set by the database administrator\(^1\), the deletion stub goes away.

When a virtual document is deleted, a deletion stub is also created. But since no document information is stored in the Notes database, it's a virtual deletion stub.

- If an external key table is used, the row in the data table is deleted, and the row in the key table is not actually deleted, but is flagged as deleted by changing the EINOTEID field. The EIMODIFIED field is also updated to the current time.
- With internal keys, there is only one table. The row is flagged as deleted by changing the EINOTEID field, and EIMODIFIED is updated, but the data are not otherwise changed.

If other applications read the data table, and you are using internal keys, the deletion stub can be inconvenient. When other applications read data from the table, they find the “deleted” items are still present.

There are two ways to deal with this.

1. You can use a relational view, or change the WHERE criteria of other applications that read the table, to exclude records flagged as deleted. A deleted document is flagged by setting the high-order bit of the 32-bit EINOTEID number. So the condition to select non-deleted documents is either $\text{EINOTEID} > 0$ or $\text{EINOTEID} < 2147483648$, depending whether your database system treats this bit as a sign flag.

2. When using internal keys, the Virtual Documents activity form contains an option to “maintain deletion stubs,” which is enabled by default. If you disable this option, then when a document is deleted from Notes, the table row is immediately deleted also. You should only use this option if you’re certain that the Notes application will not be replicated to other Domino servers or to Notes client workstations, or else deletions will not be

\(^1\) The “purge interval,” 90 days by default.
replicated out, and deleted documents may reappear. In fact, we suggest you use the replication settings dialog to disable replication of the Notes application if you use this option.

Virtual documents fully supports Domino soft deletions. A soft deleted virtual document will behave like a native soft deleted document in Domino. However, there is no difference in the relational data for a soft deleted document, so outside applications can’t tell that the document is deleted until you “empty the trash.” At that point, LEI will handle the deletion as described above.

**Tip:** Outside applications can mark virtual documents as deleted by setting the EINOTEID and EIMODIFIED fields as described in this section. This guarantees that the document will also be deleted from all replicas of the Notes application. If the row is simply deleted from the table, it will not be erased from all Notes replicas.

### 8.6.7 Domino replication involving virtual documents

If you have a virtual documents activity set up for a Domino database and you use native Domino replication either locally to a workstation or to another Domino server, the replica will contain only native Domino documents, with field values identical to those in the relational table.

When using Domino replication involving virtual documents, consider the following scenario as described in Figure 8-17. Domino server 2 has a Domino database with virtual documents activity running against a relational table. Domino replicas are created on Domino server 1 and on local workstations.

Users can create new documents on the Domino replicas (that is, on Domino server 1 or local replicas). In the normal course of events, these will replicate to Domino Server 2, and the Virtual Documents activity will intercept the attempt to create a new Notes document, storing the document fields in the relational table instead.

However, suppose that at the time of replication, Domino server 2 is functioning but the LEI server is down or the Virtual Documents activity is not running. The new documents will be created in the .nsf as regular Notes documents, and will never be inserted into the relational table.

The same sort of thing can happen without replication, if users create documents directly on Server 2 while the LEI activity is not running.
A similar scenario to consider is described in Figure 8-18. Domino server 2 has a Domino database with virtual documents activity that uses an external key table. Domino replicas are created on Domino server 1 and on local workstations.

If external business applications (such as RPG or Cobol programs) delete rows from the relational table, the deletion will not be reflected in the Domino replicas. The reason is that the key of the deleted records still exist in the external key table. The user will get an error message when they try to access the record, but it will still appear in views.
The easiest way to deal with this issue is to prevent it from occurring by avoiding any document editing on the LEI server unless the activity is running. However, if by some chance it does occur, you can “virtualize” any existing documents by using LotusScript agent code. For an example, see 8.9, "Example 4: Using a virtual documents activity" on page 182.

### 8.6.8 Event options for virtual documents activity

There are four events that occur in the Domino form that are always monitored by a virtual documents activity:

- Create
- Open
- Update
- Delete

You can add a Domino formula or stored procedure to further control these events. Refer to the LEI Activities and User Guide for more information.

### 8.7 Differences between virtual documents and virtual fields

Both virtual documents and virtual fields activities enable transparent use of external data from Domino. The primary distinction between the two is that virtual documents do not require key documents (previously referred to as stub documents) which need to have the keys initialized. Virtual fields do require such documents and the keys initialization. Furthermore, virtual documents fully support views and view refreshes.

In a virtual documents activity, all fields on the Domino form must be mapped in order for any changes to the field to be saved. Since none of the document resides in the Domino database (.nsf file), all field data is retrieved from the external system database. Any fields that are not mapped to external database fields will not be stored.

In contrast, using a virtual fields activity, any field that is not mapped to external system data is saved in the Domino document. This is because in the virtual fields activity, the key documents exist in the .nsf file.

Virtual documents activities always monitor all available form events (create, open, update, and delete). In virtual fields activities, you must specify which events to monitor.

### 8.7.1 Advantages of virtual documents activity

Following are the advantages of using a virtual documents activity:

- No component of the actual Domino document is stored natively in the Domino application. However, all the data is available to a Domino user and is indistinguishable from data actually stored in Domino (mostly).
- You are not required to maintain key documents in Domino applications, making it more scalable and easier to maintain.
- Virtual documents fully supports views and view refreshes. By contrast, virtual field data will not appear in a view unless the user elects to leave selected non-key virtual fields monitored fields in the key document, these could potentially contain stale data even after refreshing the view.
- Virtual document activities support background virtualization of records added to the external system. This process automatically adds new records that have been added to
the external system to views. It also virtualizes all records in the existing data table that have not been virtualized.

8.7.2 Limitations of virtual documents activity

Following are the limitations of using virtual documents activity:

- One Domino form can only be mapped to one virtual documents activity.
- One Domino form can only be mapped to one external data table. (However, you can combine Virtual Documents with Virtual Fields to add mappings to other tables).
- One external key table can only be associated with one virtual documents activity.
- Fields that you don't map to a back-end field, are not stored anywhere when you save the document.

**Note:** LEI will not prompt you with any errors when you create or activate multiple virtual documents activities that map multiple Domino forms to one external data table. However, you will risk getting unpredictable results when running the activities.

8.7.3 Using virtual fields with virtual documents

The ability to use virtual fields with virtual documents gives the best of both worlds. It eliminates the need of having to maintain key documents with virtual fields. It also addresses the limitations of virtual documents that only allows users to map to one external data table.

When using virtual fields with virtual documents, do not to enable updates of the join keys in the virtual fields activity. In the Event Options / Update tab, Key Field Updates should be set to Block.

Since all fields in the document are virtual, you must use the monitor order of the activity to assure that the Virtual Documents activity handles the document before the Virtual Fields activity — otherwise, the Virtual Fields activity will not have the key field it needs to locate its data.

Like Virtual Fields used on normal documents, Virtual Fields used in conjunction with a Virtual Document cannot be used in views. In this case, however, “retaining” copies of the virtual fields in the document will not help you because there is no place to store the copies. Note the discussion of “$” fields in 8.6.2, “Differences between a regular Notes document and a virtual document” on page 173 for one way to manage this; however, this is not generally advised as it can make your EINOTEPROPS field very large and harm performance.


8.8 Virtual agents

A virtual agents activity lets you run an external stored procedure as an agent that you trigger from a Domino application. The virtual agents activity creates the Domino agent and adds it to the Actions pull-down menu. It works as if you are running a normal Domino agent from the Actions menu.

The virtual agents activity can execute stored procedures with or without parameters. If the stored procedure requires parameters, you must select Domino documents from the view to
run the agent against. These documents must contain fields that are named exactly as the
parameters in the external system stored procedure and must also have matching data types.

Stored procedures that do not accept input parameters do not run against any specific
document, are created as agents whose target is None (that is, does not run on any
documents), and the stored procedure is executed once.

Procedures that do accept parameters are created as agents that Run on selected
documents and require that at least one document be selected. The stored procedure is
executed once for each selected document, using parameter values read from that document.

For a step-by-step example of using virtual agents, refer to the redbook, Implementing IBM
Lotus Enterprise Integrator 6 on the IBM @server iSeries Server, SG24-6941.

8.9 Example 4: Using a virtual documents activity

In this example we show using a virtual documents activity against a relational table. Since
most LEI users prefer external key tables, we do that in this example. We also show two
approaches for handling multivalued information.

We’re going to use essentially the same application as we’ve used in our previous examples,
with a few small changes that we’ll discuss. Figure 8-19 shows the relationships between our
different tables.

The Notes form we will use for this exercise is Agent Record VD. This form is similar to the
versions we used for the other examples, but the multivalue field Specialties field has been
renamed $Specialties, for reasons discussed in 8.6.2, “Differences between a regular Notes
document and a virtual document” on page 173. Briefly, using a name beginning with “$” will
force LEI to store a copy of the field’s binary information in the EINOTEPROPS column, so
that it will continue to be multivalued after the document is saved.

The steps to implement this example are described in the following sections.

---

**Figure 8-19  Tables and relationships for Virtual Documents example 1**

The Notes form we will use for this exercise is Agent Record VD. This form is similar to the
versions we used for the other examples, but the multivalue field Specialties field has been
renamed $Specialties, for reasons discussed in 8.6.2, “Differences between a regular Notes
document and a virtual document” on page 173. Briefly, using a name beginning with “$” will
force LEI to store a copy of the field’s binary information in the EINOTEPROPS column, so
that it will continue to be multivalued after the document is saved.

The steps to implement this example are described in the following sections.
8.9.1 Create a connection document to your relational database

For this exercise we will reuse the Sample DB connection created in 7.2.1, “Create a connection document to the RDB” on page 106.

8.9.2 Create the data tables

Either use Command activities, as described in 7.2, “Example 1: Setting up a Command Activity” on page 106, or use the SQL action in the sample Notes database, to create the AGENT_VDOC_DATA and AGENT_VDOC_EQUIP tables shown in Figure 8-19. For example, in DB2, the commands would be as follows:

```sql
CREATE TABLE yourschema.AGENT_VDOC_DATA (  
    ID INTEGER  UNIQUE NOT NULL ,
    FULLNAME VARCHAR (60)  NOT NULL ,
    PICTURE BLOB (50 K )  NOT LOGGED NOT COMPACT ,
    ASSIGNMENT VARCHAR (100)
);

CREATE TABLE yourschema.AGENT_VDOC_EQUIP (  
    ID INTEGER  NOT NULL ,
    DESCRIPTION CHARACTER (60) ,
    DATEISSUED DATE ,
    STATUS CHARACTER (10)
);
```

Remember that the Command activity and the SQL form only let you issue one command at a time.

If you’ve been following the other examples you might notice we haven’t created a column for the Specialties Notes field. This is deliberate, as we’ll see later.

8.9.3 Create the Virtual Documents activity

From the LEI main screen, use Add Activity -> Virtual Documents.

Main Section

Fill in the fields as shown in Figure 8-20 and described below.
Name: enter a unique name to identify the activity, in this case VDoc Agent Records.

Domino Application: use the selection button to locate the sample database.

You will next be prompted to select a form; select Agent Record VD.

Lotus Connection: Use the selection button to choose the Sample DB connection.

You will be prompted to select a table, since the connection document doesn’t specify one. Choose AGENT_VDOC_DATA.

Mapping: use the mapping dialog to create the mappings shown in Table 8-4.

<table>
<thead>
<tr>
<th>Notes field</th>
<th>RDB Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentID</td>
<td>ID</td>
</tr>
<tr>
<td>AgentName</td>
<td>FULLNAME</td>
</tr>
<tr>
<td>Mission</td>
<td>ASSIGNMENT</td>
</tr>
<tr>
<td>Photo</td>
<td>PICTURE</td>
</tr>
</tbody>
</table>

Notice when you do the mapping, in this case you are not asked to choose key fields. Unlike Virtual Fields, there is no document in Notes to relate the table to, so at this point it doesn’t matter what the key fields are.

Options / General

Key Table Option: since our virtual document system fields are stored in a separate table, select Use External Key Table.

Key Fields: select ID from the keyword list.

Click Create External Key Table. The resulting dialog is shown in Table 8-22. Some connectors will ask additional questions to help you determine the maximum size of the system field area (EINOTEPROPS). In case you are using one of these connectors, add at
least 200 bytes to the recommended minimum so there will be enough room for the $Specialties field in addition to the regular system fields and document header information.

![Create Key Table dialog]

The table name you enter in this dialog will be copied to the main form when you click OK, and LEI will create the AGENT_KEYS table.

**Important:** Table created with this dialog in Version 6.0.1 and 6.0.2, may not continue to work correctly with DB2 and ODBC databases when you migrate to 6.0.3/6.5 or later. These versions try to allocate enough space to store the maximum size of a BLOB column for every read. If no maximum is specified they allocate 2 GB. For steps to correct existing tables and improve performance of your virtual documents, refer to September 8, 2003 issue of LDD Today:

http://www-10.lotus.com/ldd/today.nsf/a2535b4ba6b4d13f85256c59006bd67d/62c6432de2aa112785256d9600554c5a?OpenDocument

- Click OK in the dialog.
- Enable “Find rows not entered through Notes” option.
- Select 2 minutes as the refresh frequency. This is probably far too frequent for most production applications, but for testing purposes we’d like to be able to enter new rows and see them appear in Notes fairly quickly.
- Other fields: use default values. These are the same as on the Virtual Fields form (8.3.3, “Create the Activity Document” on page 159).

**Options/Integrated Credentials**

Use default values. This is the same as the Integrated Credentials in Virtual Fields (page 163).

**Options/Virtual Attachments**

Enable Virtual Attachments and use the Create Virtual Attachment Table button to create the attachment table AGENT_VDOC_ATTACH. This is the same process described in “Virtual Attachment Options” on page 163.
8.9.4 Create a new Virtual Fields activity

From the LEI Administrator main screen, select Add Activity -> Virtual Fields (RealTime Notes). Fill in the fields as shown and described below.

**Virtual Fields Activity**

Author: Andre Guilard/Cambridge/IBM

Current Status: Not Enabled
Last Run Time: 07:59:08 PM Yesterday

### Identification

Name: "VF Equipment VDoc Agent"

### Domino Application

- **Domino Server**: Bobbby/Magic
- **Database**: "Redbook Source Data"
- **Name**: "demo/LeiSampleData.nsf"
- **Form**: "Agent Record VDoc"" Agent ID (Float)"
- **Enable Subforms**: Use Computed Subforms

### Lotus Connection

- **Data Source**: "Sample DB"
- **Table**: "AGUIRARDAGENT_VDOC_AGENT_EQUIP"

### Mapping

- **Key(s)**: "ID (m)"
- **Field(s)**:
  - "EquipDate (Binary)"
  - "EquipName (Binary)"
  - "EquipStatus (Binary)"

### Events

- **Document Events to Monitor**: "Create, Open, Update, Delete"

---

**Note:** If you do not virtualize attachments on a virtual document, LEI will not let you save the document with attachments.

If you use a Virtual Documents activity and Virtual Fields on the same Notes form (as we will in this example) the Virtual Attachment functionality may only be used from the Virtual Documents activity.
Name: Enter a name to uniquely identify the activity, in this case VF Equipment VDoc Agent.

Domino Application: use the selection button to select the sample database.

You will be prompted to select a form; select Agent Record VD.

Lotus Connection: Use the selection button to choose the Sample DB connection.

Since the connection doesn’t specify metadata, you will be prompted to select a table. Select the AGENT_VDOC_EQUIP table.

Map fields as shown in Figure 8-24 and in table below.

Table 8-5  Field mappings for VF Equipment VDoc Agent activity

<table>
<thead>
<tr>
<th>Notes field</th>
<th>Relational field</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentID (key)</td>
<td>ID (key)</td>
</tr>
<tr>
<td>EquipDate</td>
<td>DATEISSUED</td>
</tr>
<tr>
<td>EquipName</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>EquipStatus</td>
<td>STATUS</td>
</tr>
</tbody>
</table>

Events to monitor: enable all four events.

General Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Event Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor Order:</td>
<td>2</td>
</tr>
<tr>
<td>Max Connections:</td>
<td>1</td>
</tr>
<tr>
<td>Form Override:</td>
<td>Monitor any form</td>
</tr>
<tr>
<td>Filter Formula:</td>
<td></td>
</tr>
<tr>
<td>Data Integrity:</td>
<td>Prevent both precision and data loss</td>
</tr>
<tr>
<td>Trim Trailing Spaces:</td>
<td>Trim spaces on all non-key fields</td>
</tr>
<tr>
<td>New Line Delimiter for Connection:</td>
<td>Linefeed and Carriage Return</td>
</tr>
<tr>
<td>Data Storage:</td>
<td>Remove all real-time fields from documents</td>
</tr>
</tbody>
</table>

Figure 8-25  General Options for VF Equipment VDoc Agent activity

Monitor Order: Enter 2 here; the Virtual Fields activity must handle the document after the Virtual Documents activity, which has a monitor order of 1.

Multi-Value Data Options

Just as in “Multi-Value Data Options” on page 169, turn on multi-value functionality and set the rows to sort ascending by DATEISSUED.

Integrated Credentials Options

Use the default values. Integrated credentials are explained in 2.2.2, “Integrated credentials” on page 28.
Virtual Attachments Options
The Virtual Documents activity is dealing with attachments, so use the default values here.

Event Options - Open
Set the Missing External Records option to Ignore. We expect there may be some agents with no equipment, so missing equipment records are not necessarily an error.

Event Options - Update
Select Block Key Field Updates. This is the default option, but it's note that this is a necessary setting for Virtual Fields that run on top of a Virtual Documents activity.

Save and Close the new Virtual Fields activity.

8.9.5 Test the new activities

From the LEI main screen, highlight the VDoc Agent Records activity. Use the Start Activity button to make it run.

Wait for the hourglass icon to change to a radio tower, indicating the run has started. Press F9 occasionally while waiting, to refresh the view.

When a Virtual Documents activity starts up for the first time, and periodically thereafter if you have selected the Find rows not entered through Notes option, the activity will scan the relational database for table rows that have not yet been “virtualized”, that is, assigned a Note ID, UNID, and other Notes header and system fields. LEI updates the “EI” fields, and the next time a user opens a Notes view that contains these documents, the newly virtualized documents are visible.

Tip: If you have a large data table, the initial virtualization may take a considerable amount of time and CPU power. It's a good idea to use the scheduling options to set the activity to start for the first time during off peak hours.

Once the Virtual Documents activity is running, locate the VF Equipment VDoc Agent activity and start it running the same way.

With both activities running, you can now open the sample database to the Virtual Docs \ by Agent ID view, which selects only documents that use the Agent Record VD form. Create a couple of agents manually, or copy the agent records stored in regular Notes documents by clicking the Copy Orig. Documents action in the view. This runs a formula agent on “All Documents”. The agent has the Create new documents option selected (in the bottom right corner of the screen when editing the agent) and the code is shown in Example 8-2.

Example 8-2  Agent to convert regular Notes documents to virtual documents

```
SELECT @If(Form != "AgentRecord"; @False; !@IsNumber(0DbLookup("":"NoCache"; "
"VDocsByID"; AgentID; 1))));
FIELD Form := "Agent Record VD";
FIELD $Specialties := Specialties;
FIELD Specialties := @DeleteField;
```

This agent demonstrates how easy it is to write code involving Virtual Agents and Virtual Fields. As long as the activity is running, you can refer to these documents and their fields just
as you would regular Notes documents. If the agent creates a document using the form being monitored by the Virtual Documents activity, that document will be virtualized.

8.9.6 Note IDs of Virtual Documents

With the Virtual Documents activity running, open the Technical Data view of the sample Notes database. This shows all the documents in the database (including Virtual documents) with their Note IDs. The Note ID of a Virtual Document starts with “NT7”, while the ID of a regular Notes document begins with “NT0”. Actually, all the IDs from 00000000 to 3FFFFFFF are used for regular Notes documents, and 40000000 to 7FFFFFFF are used for virtual documents (virtual documents start with the high numbers and count down). As a result, you can write a formula to test the note ID, to tell them apart.

Of course, you should also be able to tell this from testing the Form field. Generally, that’s what you would do. However, as explained in 8.6.7, “Domino replication involving virtual documents” on page 178, in some cases “regular” Notes documents can accidentally get created with the form used by a Virtual Documents activity. If this occurs, you will need to “fix” these documents or the data will not be available in the relational database.

For an example of how to do this, refer to 9.8.3, “LC LSX Example 3: Fix unvirtualized documents” on page 228.
Getting Started with the LC LSX

The LC LSX is the LotusScript programming interface to the Lotus Connectors. The LSX comes with the Domino server and Notes client, so you can use it even if you don’t have LEI. The LSX lets you program any functionality you need that isn’t available through the LEI forms.

Note: The sample code in this chapter can be found in the sample database that you can download with this redbook. Refer to Chapter C, “Additional material” on page 251 for download instructions.
9.1 History of the LC LSX

An “LSX”, or LotusScript Extension, is a program library, generally written in C or C++, which provides additional functions to LotusScript code. Previous versions of LEI employed two LSX libraries:

- “lsxlc” came with the Domino server, allowing access to the functions of the Lotus Connectors for use in conjunction with DECS.
- “lsxlei” came with the LEI product, and was intended for use in conjunction with LEI, scripted activities in particular. This contained the same functions as lsxlc, but in addition, contained support for creating activity log entries and loading connection information from a Connection document in the LEI Administrator instead of specifying all the connection properties in your code.

Beginning with LEI 6, these two libraries have been merged into a single library called “lsxlc”, which contains all the functions previously available in both libraries.

9.1.1 Migrating LEI LSX code from Previous LEI Versions

If you are upgrading from a previous LEI version, and already have some code that uses lsxlei, you will need to change that code to use lsxlc instead. If it is a scripted activity, no other change should be needed; all the functions should work the same. In your scripts, in the Options section, find this statement:

```
Uselsx "*lsxlei"
```

and change it to read:

```
Uselsx "*lsxlc"
```

For any scripts that are in the script vault (leivlt.nsf), the migration utility should take care of this for you. However, if you have scripts in other databases, you will need to edit them manually.

For code other than scripted activities, the migration process is more complex. You will have to look at these and decide how to handle them. The problem is that with LEI Versions 6.0 to 6.0.2, only scripted activities may load connection information from a connection document. For details on creating connections in different situations, refer to 9.2.2, “LCConnection Class” on page 193.

If you have code now that loads connection information from a connection document, you have to either change your script to a scripted activity, or use custom code to read the fields of the connection document. Such code is available in the sample database from the Additional Resources. Look for the DECSdatabase script library.

Note: With LEI 6.0.3/6.5, it should be possible to use either a connector name or a connection document name with the LCConnection constructor, provided your code is running on the Domino server. In other cases, the DECSdatabase class may still be useful.

9.2 The LSX Classes

The classes that in the LC LSX are fully described in the Domino Designer Help, which you should consult for the details of their use, methods and properties. Here we just provide an overview of the classes and their relationships.
9.2.1 LCSession Class

The LCSession object contains global functions and information about the LC environment, including:

- The name of your session, if it is a named session (that is, if it’s a Scripted Activity script).
- Ability to write activity log entries (if in a Scripted Activity).
- Most recent error/status information, if there has been an error. Generally speaking, you will get a LotusScript error exception if any error occurs. You can intercept that with an On Error statement and check the Err and Error system variables to get information about the error. The status properties of LCSession will give you a little more detailed information, so you might use these also in your error handling code.
- Ability to turn Connection Pooling on and off (see 9.5, “Connection pooling” on page 208).

In a scripted activity, you must declare and initialize one LCSession object, to provide the session name. The session name can be any string that you make up — it need not match the name of the Scripted Activity document in the LEI Administrator. This name is used to label the log document in the LEI Log database. In a scripted activity, the syntax for creating the session object is (for instance):

```vba
Dim lcses As New LCSession("sessionName")
```

In non-scripted activity code — anything except an agent run by an LEI scripted activity — the syntax is:

```vba
Dim lcses As New LCSession
```

9.2.2 LCConnection Class

The LCConnection represents your connection to a relational database, Notes database, or other supported data source. It’s your primary object for making things happen. Using this object you can Select, Insert, Update, Fetch, call stored procedures, create and delete metadata and indexes – anything that involves reading or making changes to the database.

The LCConnection contains information about

- the type of connection (ODBC, DB2, Notes, and so on)
- which server/database you’re connecting to
- the userid and password used to access the database
- which table or view or stored procedure you’re using
- other parameters of the connection; list varies depending on the connector type. The Domino Designer Help lists the full set of properties for each connector.

There are two ways to initialize a connection, depending whether this is a scripted activity or other script.

In a scripted activity, you must use a connection document to initialize the connection. For instance, suppose you’ve created a connection a DB2 database table, and named that connection Employee Main. When you want to use that connection in a scripted activity, you could do so as follows:

```vba
Dim employeeCon As New LCConnection("Employee Main") ' in scripted activity
```

This automatically assigns all the properties of the connection object to the database, metadata, and other attributes set by the connection document.
If your code is not running in a scripted activity, then you may not use a connection document name; instead, you would supply the name of the connector you want to use, for example, “db2”, “notes” or “odbc2”. For example, in this case:

```vba
Dim employeeCon As New LCConnection("db2") ' in all other code
```

You would then need to assign the properties of the object to specify which database and metadata, userid and password, and so on.

This is a change from earlier LEI versions, which let you use a connection document name in any type of script by "using" LSXLEI instead of LSXLC. If you have scripts that need this functionality and are not “scripted activity” agents, then you have two choices how to deal with it:

1. Convert the code to a scripted activity.
2. Use custom code to locate and read the connection document, and create the document and load the connection. You can find such code in the sample database, in the DECSDatabase script library.

However you create and initialize the connection, before you can use the LCConnection object for any database operations, you must call the Connect method to establish a live connection to the database.

You may have several simultaneous connections to the same database or even the same metadata.

### 9.2.3 LCField Class

The LCField object represents a single field within a relational table or Notes form. It contains a datatype, value, and flags that control the field's behavior (for instance, whether it allows null values, or whether it's a key field for a lookup).

An LCField can contain single values, or an array of values. Multivalued LCFields are useful for doing multiple-row fetches and updates, which are more efficient for connectors that support multiple-row operations.

It's important to remember that the field name is not part of the LCField object. As we'll see when we discuss the LCFieldlist class, the same LCField object can be known by different names in different contexts, making it possible to establish data flows from one source to another using the LCField object as a conduit.

The methods of LCField allow data to be converted to and from different datatypes and different character sets. An LCField of an appropriate type can also contain binary data, including structured information such as Notes rich text fields and Notes multivalued fields, and unstructured binary information (BLOBS). Refer to the LCStream class for details about binary information.

### 9.2.4 LCFieldlist Class

The LCFieldlist is the basic unit of information that you can read from or write to a database. For a relational database, it represents a single row of data (or a collection of rows, if you're doing multiple-row fetches and updates). A fieldlist therefore has two types of indexes used to...
find information within it — a field index to select the column, and a row index to select the row. Most fieldlists in applications contain only one row — you control this when you write your code.

An LCFieldlist is an ordered collection of LCField objects, including a name for each field. Multiple LCFieldlists can reference the same LCField object by different names (or by the same name). Understanding this is critical to writing efficient code.

Suppose you want to “fetch” information from Table A and “insert” it into Table B, but the tables don’t use the same field names, or the fields aren’t in the same order. Or perhaps, some of the fields you want to write into Table B come from somewhere else, say Table C. Therefore, you can’t simply read Table A into a fieldlist and insert that same fieldlist into Table B. You need a separate fieldlist for each table.

There are two ways you can do that. The obvious way is to create the three fieldlists, put fields into each of them, and write a loop where you fetch a row, then have a series of statements to copy field 1 of fieldlistA into field 3 of fieldlistB, field 2 of fieldlistA into field 5 of fieldlist B, and so on, and finally Insert fieldlistB into Table B.

However, the obvious way is not the best way. Copying values from one fieldlist to another is very inefficient. Instead, it’s better to set things up before you get into the loop, so that field 1 of fieldlistA is the same field as field 3 of fieldlistB. Then, there’s no need to copy data from one fieldlist to another, because the data already exists in both fieldlists. You just need to fetch from your source data, and insert or update to your target data. This makes your code very efficient, especially when you can use it in combination with multiple-row operations to reduce traffic with the database server.

The examples below show how to set up and use fieldlists with shared fields.

### 9.2.5 LCStream Class

The LCStream object is used to represent any type of binary or text information — anything whose length may vary. Internally, an LCStream is used to represent the value of an LCField object of type LCTYPE_TEXT or LCTYPE_BINARY.

Since there is no LotusScript type for storing raw binary information, LCStream is essential for handling BLOBs. However, LotusScript already contains a perfectly good String type, so why would we want to use LCStream objects to deal with strings?

The answer lies in the fact that the LotusScript String type uses Unicode internally, while most relational databases do not use Unicode. In nearly all cases, the string data can be converted to or from Unicode. It’s more a matter of efficiency. Suppose you’re copying ASCII text from database A and storing it as ASCII in database B. It’s a waste of processing power to convert the string from ASCII to Unicode, then back to ASCII, when you could just copy the raw data from A to B without converting it at all. Even if you need to convert the data — let’s say from BIG5 (traditional Chinese) to GB (simplified Chinese) — it’s a waste of CPU cycles to convert the text to Unicode as an intermediate step.

LCStream also lets you manipulate special binary datatypes such as Notes multivalue fields. The Domino Designer Help contains examples of this, so we won’t get into detail here.

LCStream objects may also be used to programmatically perform text conversions. Character sets and code pages are complicated things, and it may sometimes LEI might fail to correctly identify which character set is in use in a particular case. Some of this can be addressed by setting connection parameters and environment variables, as documented in the technotes. However, on occasion nothing will do except to write a program to convert and move the data.
9.2.6 LCDateTime, LCNumeric, LCCurrency Classes

Like LCStream, these objects are used to represent field data. Also like LCStream, there are corresponding LotusScript datatypes for each one. You probably won’t use these classes very much, but there are cases where it comes in handy to have them.

- LCDateTime lets you work with time values accurate to 1/100th of a second. This is useful for preserving the exact value of a timestamp, and also for timing your scripts when working on performance.
- LCNumeric stores decimal numbers up to 88 digits precision. This is useful when working with currencies in countries (or regions) that are subject to hyperinflation. In addition, the value is represented in decimal, allowing exact representation of non-integer values.
- LCCurrency is essentially the same as the LotusScript Currency datatype. It’s provided to support connections with a dedicated currency.

9.3 Basic database operations

This section will not cover each of the relevant methods in detail — we refer you to the LC LSX documentation in Domino Designer for that. But to point you in the right direction, here’s are some little snippets of code showing how to do the basic database operations.

9.3.1 Connect to a database

After you create an LCConnection object of the appropriate connector type, set all the “required properties” for that connector. The Domino Designer Help contains a page for each connector telling what properties are available and which are required. For DB2, for example, you must supply a database name, userID and password.

Once you’ve given all the required values, use the Connect method to establish communication to the database. Until you connect, you cannot use any of the LCConnection methods that require communication to the database server.

For example:

```
Dim con As New LCConnection(“db2”)
con.Database = “SAMPLE”
con.Userid = “EGGMAN”
con.Password = “googoogjoob”
con.Connect
```

If it’s not possible to connect for whatever reason, LotusScript will generate an error condition at this point. See below for directions on error handling.

9.3.2 Handle LC errors

If any method or property causes an error in the database server — for instance, if you try to connect to a nonexistent server, use a table that doesn’t exist or insert a record that contains no value for a required field — the Lotus Connectors will generate a LotusScript error event.

As with all error events, if you’d like to do something other than the default system action of displaying an error message and aborting the script, you can use an On Error statement to intercept the error and execute your own code. Refer to the Domino Designer documentation for a description of On Error.
In addition to the three system global variables, Err, Erl, and Error, which you can use to get information about any error, the Lotus Connectors may provide additional error information which you can gain access to via the LCSession object. You can use error handling code such as the following to display a more complete error message.

```vbscript
Dim lcses As New LCSession ' for scripted activity, give activity name here.
On Error Goto HandleError
...
Exit Sub
HandleError:
If lcses.GetStatus <> 0 Then
    lcses.GetStatusText, 0, “LC ERROR"
Else ' not an LC error
    Messagebox “LotusScript Error “ & Err & “ on line “ & Erl & _
    Error, 0, “LotusScript Error”
End If
Exit Sub
```

Of course, this is just an example — you may choose to do something other than display an error message. For instance, if the error indicates that the table you’re trying to use doesn’t exist, you might have your code create the table and try again. Refer to 9.8.1, “LEI Programming 1: Transfer data from profile documents” on page 223 for an example of this.

In certain cases, you may decide to deactivate certain error conditions altogether.

### 9.3.3 Select and fetch all the records in a table

To select records in a table (or view), use the LCConnection.Select method. The statement takes three arguments: a key fieldlist that controls which documents are selected, an index within that fieldlist, and a destination fieldlist.

To select all records in the table, use the reserved word Nothing in place of the key tablename. For instance, to select all the records in table `schema.tablename`:

```vbscript
con_Metadata = “schema.tablename”
count = con.Select(Nothing, 1, destFieldlist)
```

This assumes the Condition property of the connection has not been set. Refer to 9.3.4, “Select records that meet certain criteria” on page 198 for details on how the Select method controls which documents are selected.

The count value returned by the call may tell you how many matches were found, depending on which connector you use. Mostly, it returns LCCOUNT_UNKNOWN, which doesn’t tell you anything. You would have to try a Fetch to see whether there were any rows found.

The destFieldlist argument should be an empty fieldlist before you call Select. The Select statement will add fields to this fieldlist that describe the columns in the result set. For instance, if you Select from a table that has three columns in it, DOGID, NAME, and BIRTHDATE, then after the Select, destFieldlist will contain three fields with those names. The fieldlist will not contain any data yet, though: you must then fetch rows.

```vbscript
While con.Fetch(destFieldlist)
    ' do something with the data in destFieldlist
Wend
```

Since the destFieldlist you supplied to the Select contains definitions for the fields in the table, it makes sense to use that fieldlist for the Fetch operation. However, you can use a different fieldlist if you wish.
The Fetch method returns a count of records fetched; by default it tries to fetch one record. A return value of 0 means it's reached the end.

In many cases, it's more efficient to define a multi-row destination fieldlist and fetch multiple records simultaneously. Refer to “Progressive5: Multi-row fetch” on page 217 for an example.

9.3.4 Select records that meet certain criteria

To limit your result set to a subset of the records in a table, you can either use the Condition parameter to specify a WHERE clause for the select, or supply a key fieldlist, or both. You can also add a timestamp-based condition as described in 9.3.11, “Using database timestamps” on page 204.

When you call the Select method, the connector code creates an expression that selects the records to be fetched. For databases that use SQL, this will be expressed in SQL format as the WHERE clause of a SELECT statement; for the Notes connection, it will be a macro formula to be used in a SELECT macro statement.

There are three components that go into this expression.

- The key fieldlist argument. The method scans the fieldlist for fields that have the LCFIELDF_KEY flag set. These are “ANDed” together. The fieldlist argument may be Nothing (a special LotusScript constant) or may contain no key fields, in which case all records are selected.
- If the Condition property of the LCConnection object is assigned, then it will be appended to the expression with an AND.
- If the StampField and BaseStamp properties of the LCConnection object are assigned, the WHERE clause will specify that only records with a timestamp later than the supplied time will be included in the result set. For example, if StampField = “Modified” and BaseStamp = Cdat(10/10/2002 12:08:17.45 PM), then the WHERE clause might include “… AND Modified>='10/10/2002 17:08:17.45’ ” (of course, the date/time will be converted to the format required by that database server).

The key fieldlist, if you supply one, may contain some fields that are keys and some that are not. A field is considered a key if its Flags property has a certain bit set — the system constant LCFIELDF_KEY defines that bit position.

By default, the comparison is for equality with the value in the fieldlist. There are several flag bit values you can combine with LCFIELDF_KEY to modify the resulting WHERE expression. Table 9-1 shows the meanings of these flags.

Table 9-1 Meanings of LCFIELDF_KEY...

<table>
<thead>
<tr>
<th>Flag name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCFIELDF_KEY</td>
<td>This is a key field for the search. If this bit is not set, the field is not included in the WHERE clause even if other KEY flags shown below are used.</td>
</tr>
<tr>
<td>LCFIELDF_KEY_NE</td>
<td>Select records where the value of the field is “not equal” the value supplied in the fieldlist.</td>
</tr>
<tr>
<td>LCFIELDF_KEY_LIKE</td>
<td>Select records where the field matches the pattern matching string supplied in the fieldlist.</td>
</tr>
<tr>
<td>LCFIELDF_KEY_GT</td>
<td>Select records where the value is greater than or equal to the value in the fieldlist.</td>
</tr>
<tr>
<td>LCFIELDF_KEY_LT</td>
<td>Select records where the value is less than or equal to the value in the fieldlist.</td>
</tr>
</tbody>
</table>
The names of the “GT” and “LT” flags are a little deceptive because the initials sound like greater than and less than. To do a greater than or a less than comparison, combine these flags with LCFIELDF_KEY_NE. For instance, LCFIELDF_KEY + LCFIELDF_KEY_GT + LCFIELDF_KEY_NE means “greater than”.

For instance, suppose you have a fieldlist, flKey, containing the following fields:

1. Status: text
2. DateIssued: date/time
3. DateIssued: date/time (there’s no problem having the same name multiple times provided you only use the fieldlist as a key).

First, we should define LCField objects to store pointers to these fields. If we’re creating the fieldlist ourselves, it’s easiest to do this as we add them to the list.

```vba
Dim flKey As New LCFieldList(, LCFIELDF_KEY)
Dim fStatus As LCField, fStartDate As LCField, fEndDate As LCField
Set fStatus = flKey.Append("Status", LCTYPE_TEXT)
Set fStartDate = flKey.Append("DateIssued", LCTYPE_DATETIME)
Set fEndDate = flKey.Append("DateIssued", LCTYPE_DATETIME)
```

The first line of the above code establishes that every field added to this fieldlist will be a key field, unless we override that by explicitly setting its Flags property. We next can specify what type of comparison we want to do with each entry.

```vba
fStatus.Value = "OK"
fStartDate.Flags = LCFIELDF_KEY Or LCFIELDF_KEY_GT
fStartDate.Value = Datenumber(2001, 1, 1)
fEndDate.Flags = LCFIELDF_KEY + LCFIELDF_KEY_LT + LCFIELDF_KEY_NE
fEndDate.Value = Datenumber(2002, 1, 1)
```

Since fStatus.Flags = LCFIELDF_KEY, this will be a simple “equals” comparison, translated to HTML as STATUS = ‘OK’. However, fStartDate.Flags = LCFIELDF_KEY + LCFIELDF_KEY_GT. The test here is not for equality; we’re selecting only records where the value in the record is greater than or equal to the value in the fieldlist, in this case 1 January 2001. In DB2 SQL, that would be DATEISSUED <= ‘2001-01-01’. Then, fEndDate.Flags = LCFIELDF_KEY + LCFIELDF_KEY_LT + LCFIELDF_KEY_NE says that the value in the record must less than the value in the fieldlist, 1 January 2002; in DB2 SQL: DATEISSUED < ‘2002-01-01’.

Finally, we’re only interested in the equipment records of agents who appear in the AGENTSTATUS table with a status of “ACTIVE”. The Lotus Connectors are not able to generate this type of SQL, so you have to add this yourself using the Condition property:

```vba
con.Condition = {ID in (SELECT ID FROM schema.AGENTSTATUS WHERE STATUS = 'ACTIVE')}  
con.Metadata = "schema.AGENTEQUIP"
Call con.Select(flKey, 1, flDest)
```

The resulting SQL would be:

```
SELECT * FROM schema.AGENTEQUIP WHERE STATUS = 'OK' AND DATEISSUED >= '2001-01-01' AND DATEISSUED < '2002-01-01' AND (ID in (SELECT ID FROM schema.AGENTSTATUS WHERE STATUS = 'ACTIVE'))
```

If the first argument to Select is Nothing, or if there are no fields in the key fieldlist flagged as LCFIELDF_KEY, then the Condition property is the entire WHERE clause. If the Condition property is also blank, all records in the table are selected.
9.3.5 Insert a new record

Use LCConnection.Insert method to create a new row in the relational table. This method takes as argument a fieldlist containing the data to be inserted. It doesn't matter whether any of these fields are flagged as key fields using LCFIELD_KEY, since no lookup is performed.

As regards mapping the fields in the fieldlist to the fields in the table, there are two modes of operation. Use con.MapByName = False (the default) to insert the first field in the fieldlist into column 1, the second field into column 2, and so on. Use con.MapByName = True to match the fields based on their fieldnames. Mapping by name is slightly less efficient, but it continues to work if the table definition changes.

Note: Typically, the order of fields in a relational table is unlikely to change. However, editing a Notes form and changing the order of the fields there, will change the order when you access the document using the Notes Lotus Connector via the LSX. We recommend using name mapping when working with a Notes connection.

There are several ways to create the fieldlist to use in the Insert. One of the easier ways, is by using Select. This creates a list that's the same fields, the same data types, in the same order as in the current metadata. You can then refer to the fields you want by their names. For instance:

```vbscript
Dim con As LCConnection
Dim insertFieldlist As New LCFieldlist
...
con.Condition = "0 = 1" ' will not select any records
con.Select Nothing, 1, insertFieldlist
con.Condition = ""
...
insertFieldlist.DogID = 7708
InsertFieldlist.Name = "Rex"
InsertFieldlist.BirthDate = Datenum(1998, 3, 14)
con.Insert insertFieldlist
```

Alternately, you can construct the fieldlist yourself, which saves you an unnecessary Select at the cost of slightly more complex code.

```vbscript
Dim dogID As LCField, dogName As LCField, dogDate As LCField
...
Set dogID = insertFieldlist.Append("DogID", LCTYPE_INT)
Set dogName = insertFieldlist.Append("Name", LCTYPE_TEXT)
Set dogDate = insertFieldlist.Append("BirthDate", LCTYPE_DATETIME)
...
dogID.Value = 7708
dogName.Value = "Rex"
dogDate.Value = Datenum(1998, 3, 14)
con.Insert insertFieldlist
```

In this case we took advantage of the fact that the LCFieldlist.Append method returns a handle to the new LCField object. We store those field handles in LCField variables so that we can later refer to them through those variables, rather than having to use the
fieldlist.fieldname syntax each time. Referring to them by name still works, it’s just a little less efficient.

The properties of the fieldlist you create using Append or Insert might not be identical to the properties of the fields in the table — for example, you might have an Integer field where the table has a Float, or the table might have a size limit on a field where you have not specified a limit in your fieldlist. This doesn’t matter so long as the data you insert is able to be converted to the type of data the table expects — the conversion happens automatically.

It’s possible to insert multiple records simultaneously by using a fieldlist that’s dimensioned to contain multiple rows. For an example, refer to the Domino Designer Help page for this method.

### 9.3.6 Delete records that meet certain criteria

The LCConnection.Remove method is used to delete rows from a relational table, or Notes documents from a .nsf file if you’re using a Notes connection.

The Writeback setting of the connection affects how the Remove method works. If con.Writeback = True, then the last record Fetched will be removed. If not in Writeback mode, then the fieldlist you supply as an argument will be used as a key fieldlist to construct a WHERE clause specifying which records to delete. The rules for constructing this WHERE clause are the same as for the Select method (see 9.3.4, “Select records that meet certain criteria” on page 198).

**Important:** Caution when using the Remove method in non-Writeback mode: if none of your fields are flagged as keys and you have no Condition specified, you will remove every record in the table.

For example, suppose you want to delete all the records from the AGENTEQUIP table where Status is “Lost”. You might code this as follows:

```vba
Dim con As New LCConnection("...")
Dim removeKeylist As New LCFieldList(1, LCFIELDF_KEY)
Dim removeStatus As LCField
Dim count As Long
Set removeStatus = removeKeylist.Append("STATUS", LCTYPE_TEXT)
...
con.Metadata = "AGENTEQUIP"
removeStatus.Value = "Lost"
count = con.Remove(removeStatus)
Print "removed " & count & " records."
```

By using a multi-row fieldlist and passing a starting index and count to the Remove method, you can do multiple Removes with a single operation.

In Writeback mode (con.Writeback = False), the key fieldlist and Condition property are ignored — the record last Fetched is the one deleted.

### 9.3.7 Update a record using writeback

The LCConnection.Update method is used to change the values of fields in a table row. When using Writeback mode, you can update just one record at a time — the record most recently Fetched.
The following script selects all the records in a table, reads them one by one, and determines whether they need to be marked as “Completed” by calling a custom function.

```vba
Dim con As LCConnection
Dim flSrc As New LCFieldlist
Dim fldStatus As LCField
...
con.Writeback = True
con.Metadata = "...
con.Select Nothing, 1, flSrc
Set fldStatus = flSrc.Lookup("STATUS")
  ' get a handle to STATUS field to avoid lookups inside loop.

While con.Fetch(flSrc)
  if isCompleted(flSrc) Then
    fldStatus.Value = "Completed"
    con.Update flSrc
  End If
Wend
```

The LCConnection.MapByName property does apply to the Update method — when the changes are written to the table, if MapByName is False, field 1 of the fieldlist will be written to column 1 of the table, field 2 to column 2, and so on, regardless of the names of the fields. If MapByName is True, the names in the fieldlist are used to match the fields to the corresponding columns. Mapping by position is slightly more efficient, but more prone to stop working if the table definition is changed. In the above example, we are quite safe using position mapping, since the fieldlist created by the Select operation is automatically in the same order as the table columns.

### 9.3.8 Update a record using a selection key

If Writeback mode is not active, then the LCConnection.Update method performs a keyed update. This means that the connector constructs a WHERE clause using the Condition property of the connector and any fields in the fieldlist that are flagged as keys. This is done the same way as for a Select operation (refer to 9.3.4, “Select records that meet certain criteria” on page 198).

Since the keys may match multiple rows, it's possible to update many rows — or all of them — with a single statement.

**Important:** In non-Writeback mode, if you don’t have any key fields in your fieldlist, and the Condition property of the connection is blank, you will update every record in the table.

As an example, suppose that you want to update all the rows of the DOGS table where DOGID = 652, changing the Name field to “Princess Throwrug”. The following code will do that:

```vba
Dim con As New LCConnection("...")
Dim flUpdate As New LCFieldlist
Dim dogID As LCField, dogName As LCField
con.Writeback = False ' the default value
con.Metadata = "DOGS"
con.MapByName = True ' so that we don't have to give a value for every field!
...
Set dogID = flUpdate.Append("DOGID", LCTYPE_INT)
dogID.Flags = LCFIELDF_KEY
dogID.Value = 652
Set dogName = flUpdate.Append("NAME", LCTYPE_TEXT)
```
dogName.Value = “Princess Throwrug”
con.Update flUpdate

If you’re an SQL devotee, here’s the corresponding SQL statement:

UPDATE DOGS SET NAME = ‘Princess Throwrug’ WHERE DOGID = 652

By using a multi-row fieldlist, you can also perform multiple updates with different key values and data values in a single operation. Refer to the Domino Designer Help for details.

Note: If the Update selection criteria don’t match any records, this is not an error condition. The Update method will just return a value of 0 to indicate that no records were updated.

### 9.3.9 Control Commit and Rollback

You may have applications that require that you update multiple records at a time and complete all the updates before you commit any of them. This may especially be the case when there’s data in multiple tables and you need to commit your changes only once the data in all the tables are in a consistent state.

The LCConnection class contains properties that let you control when commits happen. The LCConnection.CommitFrequency property can be used to control how frequently commits happen in that connection — a value of zero makes the connector not commit until you explicitly tell it to using the LCConnection.Action method.

Normally, if your script is accessing data in multiple tables (or views) of the same database, you might create a separate LCConnection object for each table. This saves you having to change the Metadata property as you switch among the tables. However, if you use more than one LCConnection to do updates, then there’s not a way to guarantee commit of all the updates. If it’s critical that commits on multiple tables happen simultaneously (or that you be able to roll them all back), then you must use a single LCConnection object for all your operations, and use the CommitFrequency property and Action method as described above to control when commits happen.

**Example 9-1** Insert records into two tables and commit them simultaneously

```vbnet
Function InsertDogData(con As LCConnection, dogRec As LCFieldList, _
    awardRecs As LCFieldList, awardCount As Long) As Integer
    On Error Goto HandleError
    con.CommitFrequency = 0 ' only commit when we say to commit.
    con.Metadata = "DOGS"
    con.Insert dogRec
    con.Metadata = "DOGSHOWAWARDS"
    con.Insert awardRecs, 1, awardCount
    con.Action LCACTION_COMMIT ' commit changes to both tables.
    InsertDogData = 1 ' code for success
    Exit Function

HandleError:
    ' In case of error, roll back any changes already made.
    con.Action LCACTION_ROLLBACK
    InsertDogData = 0 ' code for failure
    Exit Function
End Function
```

Note:
If the Update selection criteria don’t match any records, this is not an error condition. The Update method will just return a value of 0 to indicate that no records were updated.
9.3.10 Call a stored procedure

Use the LCConnection.Call method to invoke a stored procedure. First, set the Procedure property of the connection to the name of the procedure you want to call. If the procedure has arguments, you must supply a fieldlist containing the input arguments. Like the Select and Execute methods, you also may supply an output fieldlist to receive the definition of the result set.

Some connectors will let you receive a result set or output parameters from a stored procedure (for example, DB2). Others will let you retrieve output parameter values but not a result set (for example, Oracle). In either case, you obtain the output values by passing the Call method an output fieldlist and then calling Fetch to retrieve the values. In the case of output parameters, there's only one row of results to fetch. For a result set, you can continue to fetch until you exhaust the result set.

At least in some connectors, you can call a stored procedure using the LCConnection.Execute method, but depending on the connector, you may not be able to get any data back that way.

If you can’t get a result set back any other way, you can always have the procedure write its results to a temporary table, and select and fetch from that table after you run the procedure.

The Notes, File and Text connectors don’t support Stored Procedures.

Look for an upcoming article in LDD Today (http://www-10.lotus.com/ldd/today.nsf) for a more complete description of stored procedure usage.

9.3.11 Using database timestamps

One of the challenges in synchronizing operations between any two systems, is that any two computers may have a different idea about what time it is. Frequently, you want to issue queries for records that are new or modified since the last time you checked. To do this, you have to be able to tell what time it is on the database server, as opposed to the Domino server. When you update a record that contains a timestamp field, you must store the database server's timestamp, not the Domino server's time.

The relational database connectors have three fields that you can use to select and update records based on the database server's current time value. These are:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StampField (string)</td>
<td>The name of a field in the current Metadata that contains a timestamp showing when the record was last updated.</td>
</tr>
<tr>
<td>BaseStamp (date/time variant)</td>
<td>The Select method will only select records where the value of the stamp field is later than this date and time. When</td>
</tr>
<tr>
<td>MaxStamp (date/time variant)</td>
<td>The Select statement will assign this property to the current server timestamp.</td>
</tr>
</tbody>
</table>

Example 9-2 shows how you might use these properties to do a “new and modified” selection on a relational database. It uses a Notes profile document to remember the server timestamp value from the last time this agent ran. If there is a value recorded, it uses the StampField and BaseStamp properties to restrict the selection to only those records that were modified since that time. After the select (and after committing any changes it makes as a result of
processing the source records) the code updates the profile document with the new MaxStamp value.

Example 9-2  Selecting new and modified records using a timestamp

```vbscript
Dim ses As New NotesSession
Dim db As NotesDatabase
Dim docProfile As NotesDocument
Dim lces As New NotesSession
Dim lccon As New LCConnection("db2")
Dim fElephantRecord As New LCFieldList
lccon.Database = "Sample"
lccon.Metadata = "AGUIRARD.ELEPHANTINFO"
lccon.Connect
Set db = ses.CurrentDatabase
Set docProfile = db.GetProfileDocument("MasterProfile")
If docProfile.LastElephantSearch(0) <> "" Then
    lccon.StampField = "MODIFIED"
    lccon.BaseStamp = docProfile.LastElephantSearch(0)
End If
Call lccon.Select(Nothing, 1, fElephantRecord)
    ' ...
Wend
docProfile.LastElephantSearch = lccon.MaxStamp
docProfile.Save True, True, True
```

Unfortunately, the LC LSX does not currently support automatically updating a timestamp field with the server timestamp when writing the record. As a work-around, you can do one of the following:

- Use the LCConnection.Execute method instead of Update or Insert. Use the SQL syntax of the end database to tell the database server to insert the current time. For instance, in DB2 you might write:

  ```sql
  UPDATE schema.tablename SET NAME='Fred', MODIFIED=CURRENT_TIMESTAMP
  WHERE ID=5066
  ```

  Different relational databases use different keywords instead of “CURRENT_TIMESTAMP”

- Use the Select method to find out the current time on the server — you don’t have to fetch any records, just Select so that you get a MaxStamp value — and then copy that into your output field. The resulting timestamp might not be identical to the server’s time, since there’s a small delay between your getting the time value and updating the record. To compensate for this, you might add a few seconds to the server time. Otherwise, there’s a small chance that another process will fail to notice the update because it did its last check between the reported update time and the actual time.

- Use a stored procedure to do the update, passing it the list of field values using the Call method, and have it add the timestamp.

9.3.12  Execute SQL that you create in your program

The Execute method takes an SQL statement, which can be either static text or a variable containing a value you calculate. This is your way of doing anything that you can’t do with the other methods that the LSX provides. If the command produces a result set, you can typically read the result set using Fetch. Here’s an example of the use of Execute.

```vbscript
Dim con As New LCConnection("...")
Dim f1Dest As New LCFieldList
```
Dim fld As LCField
...
Call con.Execute(_
    "select AGENTS.ID, FULLNAME, DESCRIPTION, STATUS from schema.AGENTS," & _
    " schema.AGENTEQUIP WHERE AGENTS.ID = AGENTEQUIP.ID", _
    fldDest _
)
While con.Fetch(fldDest)
    For i = 1 To fldDest.FieldCount
        Set fld = fldDest.GetField(i)
        Print fldDest.GetName(i) & " = " & fld.Value(0)
    Next
Wend

There is no way to join data from multiple tables using the Select statement, so Execute is the only way to do something like the statement shown here. (You could create a view in the relational database and query the view with a Select, but you don’t necessarily want to create a view for every multiple table fetch you want to do!)

### 9.3.13 Work with binary data

The same class — LCStream — is used to work with text values and binary data. It's easy to get the text data out of a text field and convert it to a LotusScript string using the Text or Values property of LCField, so you generally don’t need to work with LCStream objects to manipulate text fields.

Binary fields are a different story. Because LotusScript has no built-in type to represent binary information, there’s no way to work with these values except what the LSX provides.

The Binary datatype is used to represent "BLOB" values. These values may have some internal structure, but there’s no way to get at the individual bytes and read or modify them, and no way to create a blob from scratch in the LSX. You can read them from one database, and write them to another database.

There are certain exceptions. If you happen to know that the data in the blob is characters in a particular character set, then you can tell the LSX what character set to use, and convert the characters into a different character set or into a LotusScript string. For instance, suppose you know the data in a blob is a Unicode string, and you want to copy the information to a text field. The LSX doesn’t know that the data is Unicode, so if you ask to have it converted to text, even Unicode text, the LSX doesn’t know how to do it and you get an empty string as a result.

To tell the LSX that the data already in the buffer is in Unicode format, you could use LCField.SetFormatStream as follows:

```languagemarkup
Call fldBlob.SetFormatStream(, , LCSTREAMFMT_UNICODE)
```

This will let you read off the contents of the field — now, the LSX knows that this is Unicode text, and you can ask for its Text property, copy it to another field, or write it directly to another database, where it will be converted to whatever character set is used there.

A binary stream is also used to represent some Notes field types that have no corresponding relational types. The format constants to indicate these field types are shown in Table 9-3.

<table>
<thead>
<tr>
<th>Format name constant</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCSTREAMFMT_COMPOSITE</td>
<td>Notes rich text.</td>
</tr>
</tbody>
</table>
You will only ever see these types when you Select data from a Notes LCConnection, and the “list” types are only used if the MultiValueAsText property of the connection is True. However, you can use these constants in your code. If you read some blob data from a relational table, and you happen to know that the data is the exact binary representation of one of these four special binary field types, then you can set the format of the field as shown above, and either write the data into a Notes document, or use the methods of LCStream to read the values in the field.

For instance, suppose you have read the fieldlist flSrc from an Oracle database, and you know that the TASKDATES field is a copy of the binary data from a Notes multivalue date/time field. You want to know whether any of the dates is later than a given date. One way to code this is as follows:

```vbscript
Function IsTaskDateLaterThan(flSrc As LCFieldlist, endDate As LCDatetime) As Boolean
    Dim fldDate As LCField
    Dim stream As LCStream
    Dim aDate As New LCDatetime
    Dim i As Integer
    Set fldDate = flSrc.Lookup("TASKDATES")
    Set stream = fldDate.GetStream(1, LCSTREAMFMT_DATETIME_LIST)
    For i = 1 To stream.Valuecount
        stream.DatetimeListGetValue(i, aDate)
        If aDate.Compare(endDate) > 0 Then
            IsTaskDateLaterThan = True
            Exit Sub
        End If
    Next
End Function
```

### 9.4 Using the Notes Connector

In addition to the connectors allowing access to relational databases, file systems and text files, the Lotus Connectors include a Notes connector that you can use to access data in Notes databases.

In some cases, it's appropriate to use the LotusScript built-in classes of the Domino Object Model – NotesDatabase, NotesDocument, NotesView and NotesItem are the most useful ones.

However, when doing data transfers between Notes databases and other data sources accessed via the Lotus Connectors, it's generally more efficient to access the Notes database via the Notes connector. As we discussed in 9.2.4, “LCFieldlist Class” on page 194, you get your best performance by avoiding “touching” the data as much as possible. If you read Notes data directly into a fieldlist, then you can write it out to (let's say) a DB2 table without having to individually assign each field of your output fieldlist.

In addition, using the Notes connector, a binary object in your Notes database – in particular, the contents of a rich text field – can be copied directly into a BLOB field in a relational database, allowing back-up of Notes information into a relational table. Using the Domino
Object Model, you can extract the text from the rich text item, but you can’t copy the exact binary representation to another location outside of Notes, as you can with the LC LSX.

9.5 Connection pooling

Connection pooling is the ability of the LSX to “hold open” a connection after a script issues a Disconnect command. The next time a script requests an identical connection (same database, userid and password), the matching connection is taken from the “pool” and reused. If there’s no matching connection in the pool, the LSX creates a new connection.

The purpose is to improve performance by avoiding the overhead of making a new connection every time a process needs to use the database. This can be an expensive operation, on the order of a second or more, which can be a noticeable wait.

As an example of where connection pooling would be useful, imagine a Domino Web agent which is the WebQuerySave on a form. The agent uses a field from the current Notes document as a key to look up information in a relational table, and stores that information in fields of the Notes document. Since Domino agents are not memory resident, this agent can’t create a connection once and just hang on to it until the server is shut down. But it’s a big waste of time to create a new connection just to do a single lookup. You can speed up this agent considerably by just activating connection pooling. To do this, just set the ConnectionPooling property of the LCSession object to True. Figure 9-1 graphically shows the connection pooling algorithm.

**Note:** Connection pooling is turned off by default. Each script that wants to use the connection pool must activate connection pooling using this statement:

```plaintext
lcSession.ConnectionPooling = True
```
With connection pooling, the LCConnection.Connect method first checks the pool for an existing compatible connection. If one is not available, the LSX creates a new connection. Two connections are compatible if all of their “required” connectivity properties are the same. Different connectors have a different set of required properties, but they all require a userid and password. This prevents a connection originally established for one user, later being used by another. For example, a connection to DB2 which was originally created using John’s user name and password, and then released to the connection pool, would not be issued if a new connection was requested using Katy’s user name and password.

A connection is removed from the pool when you call the ‘Connect’ method of LCConnection. A connection is returned to the pool when you call the ‘Disconnect’ method, or when the LCConnection object is deleted.

**Note:** In LotusScript, if no explicit ‘delete’ occurs on an object, it is automatically deleted when it falls out of the scope of the function, subroutine, or script.

Keep in mind that a connection that is returned to the pool does not disconnect from the external system. Code that takes advantage of connection pooling must anticipate this behavior. The important issues to remember are related to what may happen automatically during a normal disconnect from an external system. For example, disconnecting from an RDBMS may trigger a commit of records inserted, updated, or deleted, since the connection was first established. Likewise, there may be rollback or other database operations that take place automatically as a result of disconnecting. When connection pooling is enabled, these events do not take place because the connection is not actually dropped. Therefore, if you
expect and want these types of operations to take place, the processing portion of the script must explicitly perform them (using the LCConnection.Action method).

The life span of a pooled connection is dependent on the LSX. Within Notes and Domino, an LSX is loaded when the execution of the Uselsx statement occurs in a script. The LSX is not unloaded until the Notes or Domino process terminates. Once a connection is pooled, it remains available until the associated process terminates.

The connection pool defaults to a maximum of 20 pooled connections for a given external system. When the maximum number of connections have been created and are already in use, a request for an additional connection will be granted but the connection will not be pooled. The default may be overridden using settings in the notes.ini file.

You configure the size of the connection pool using the EIConnectionPool variable in the server's INI file. The syntax for this setting is a comma-delimited list containing the Lotus Connector name, the pool size (integer), and the data source maximum (integer). For instance:

```
EIConnectionPool=oracle,30,13,db2,25,5
```

The default pool size for all connectors is 20. The data source maximum value indicates the limit of allowed connections to a single database. This value cannot be greater than the total pool size for a given connector. For example, a DB2 pool size of 10 and a data source maximum of 5 indicates the pool will hold no more than 10 connections to DB2 and of the 10, no more than 5 will be to any one database.

**Note:** Documentation in the Domino Designer Help and elsewhere, states that the data source maximum value is optional. However, for versions of LEI through 6.5.1, you must supply this value.

### 9.6 Performance tips

Many of the following points are demonstrated in 9.6.1, “Performance code samples” on page 212.

1. Don’t write your own SQL. There is an LCConnection.Execute method that lets you send an SQL statement to the relational database. However, it’s better to avoid using this method if you can find a way to get the connector to come up with the SQL for you. The connector code is written in C and C++, so it can create the SQL text faster than any LotusScript code you can write. Also, the APIs of the various relational databases let a C program pass in field values without converting them to text, a considerable time savings.

2. Avoid copying values from one field to another. Statements such as:

```
fieldlistA.FullName = fieldlistB.Name
```

are a waste of time. Instead, use any of several LCFieldlist methods to create multiple references to the same field from different fieldlists. This lets you use the same field in connection with multiple data sources — for instance, to read it from one and write it to another — without your code having to handle the value and convert it to a LotusScript type and back again to an LCField.

3. Define your text LCStream fields using a character set that matches either the source or the target table. As much as possible, the Lotus Connectors try to avoid converting text data, but handle it in its original binary form. If you read data from a database that uses the Japanese character codes, and write it into another database that uses the same codepage, there’s no reason to convert it to Unicode in between. Unless you need to
manipulate the contents of the strings — for instance, to change it to uppercase — just read it into a text LCField with a matching format, and don’t peek at the value.

The easiest way to make the container match the format of the data, is to use the fieldlist returned by an LCConnection method. For instance, the Select method returns a fieldlist with the exact types, formats and flags of the fields in the data source. This fieldlist is the most efficient to use for your fetches.

4. Searching for a document by key in the Notes database is fairly slow. Any Select, Remove or Update operation using key fields is going to be less efficient than the corresponding operation in a relational database, because there is no equivalent in Notes to an indexed column and because the Notes connector has to apply the selection formula to all the documents in the database — there’s no equivalent to a relational table.

You can use a view to select Notes documents, which tends to be more efficient because the view index is calculated in advance by the server. However, you can’t combine a view selection with a keyed select — you have to use one or the other.

When updating or deleting records in a Notes database, it’s more efficient to use writeback mode.

5. Sometimes you need to read data from a source and either insert or update it in the destination, depending whether there’s already a destination record with the corresponding key. Rather than do a Select on the destination to search for a matching record, it’s faster and easier to try a keyed Update first, and if that returns zero (no records updated), try Insert instead.

6. When scanning records from one database and checking for records with the same key fields in another database, it’s usually more efficient to select a result set from each source that’s sorted by the key values, and fetch records from the two sorted lists to compare. This is true regardless of whether one or both of the connections are to Notes databases (though the difference may be more noticeable with Notes). This is how the LEI Replication activity works. This is an alternative to point number 5 above.

7. Native sorting using the OrderNames property of a connection, or by reading from a sorted view, is substantially faster than using the Order metaconnector. Use the metaconnector when there’s no other way to make the data in two different data sources sort consistently. For instance, because of their different native character sets, DB2 on OS/400 and most other databases have different ideas about which come earlier in the collating sequence — digit characters or letters. Having identical sort orders is important in doing efficient “replication” code (see previous list item).

8. Use Connection Pooling (see 9.5, “Connection pooling” on page 208), especially when doing only a couple of transactions in each execution. The time it takes to establish a connection can easily be more than the time it takes to actually do the transaction. Remember to manually commit or cancel any transactions you make in a pooled connection, or they may be left over for the next user of that connection.

9. Most LCConnection methods work with fieldlists that may contain multiple rows of data. These methods accept a Index and Count parameters to tell which of the rows you want to operate on and how many rows you want to process.

If the logic of your application permits, it’s generally more efficient to fetch and write multiple rows of information at a time. There’s overhead involved in sending a request and receiving a response from the database server, so if you can send fewer requests to receive the same total amount of data, you’ll gain in performance.

Not all connections support multiple-row operations, so you will not necessarily gain much in performance to compensate you for the slightly more complex program logic. But the majority of them do, and even those that don’t will still manage to carry out your request — they’ll just split it up into multiple one-row commands.
10. Before entering a long “Fetch” loop, use the LCFieldlist.Lookup method to get “handles” to the LCField objects that you will reference inside the loop. You can reference a field value using `fieldlist.fieldname` syntax, but this is not efficient because LC must search the fieldlist for that field each time. Furthermore, the “extended syntax” converts the field’s value to an array of LotusScript primitive datatypes; if you use the LCField object you can often work with the value without having to convert it. This is more efficient, and also less likely to lead to precision loss or to the addition of extraneous times to date values.

### 9.6.1 Performance code samples

To illustrate the above tips, we’ve supplied a series of LotusScript agents that show some of them being applied. We begin with a script that, although functional, does everything wrong. We then revise it step by step into a sleek, speedy version that incorporates several performance enhancements. The results from an informal timing test are shown here:

<table>
<thead>
<tr>
<th>Comments</th>
<th>Runtime (sec)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Original version</td>
<td>14.53</td>
<td></td>
</tr>
<tr>
<td>2. Use SELECT instead of creating own SQL.</td>
<td>12.34</td>
<td>17.70%</td>
</tr>
<tr>
<td>3. Avoid unnecessary SELECT by testing</td>
<td>6.35</td>
<td>94.18%</td>
</tr>
<tr>
<td>return value from keyed Update.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. UseMapName to share LCField objects</td>
<td>4.80</td>
<td>32.36%</td>
</tr>
<tr>
<td>instead of copying data field by field.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Multiple-row Fetch</td>
<td>4.66</td>
<td>2.93%</td>
</tr>
</tbody>
</table>

These results were done with a local DB2 database, average over several runs. Results would vary based on circumstances, of course. A multiple-row fetch will help a lot more with a remote database, and sharing LCField objects will help more if you have more fields than the few in the sample database. But this should give you some idea which types of changes will do you the most good.

The following sections contain details of the different agent versions.

### Original “Progressive Improvement” agent

Here’s the source code showing how not to copy data from one database to another. Portions of the code that are the same in all versions of the agent are omitted, but you can see the entire code in the sample database.

**Example 9-3  Example showing how not to use the LC LSX (Progressive1 agent)**

```source
conSrc.Connect
conDest.Connect
Call flDest.Append("COMPANY", LCTYPE_TEXT)
Call flDest.Append("LNAME", LCTYPE_TEXT)
Call flDest.Append("FNAME", LCTYPE_TEXT)
Call flDest.Append("MNAME", LCTYPE_TEXT)
Call flDest.Append("TELEPHONE", LCTYPE_TEXT)
```

*Note: We’ve found that multiple-row fetches don’t work reliably with the Order and Broker metaconnectors (Broker is not documented in this release, but is still available). This has been reported to development. Other connectors do not have this problem.*
Call flDest.Append("EMAIL", LCTYPE_TEXT)

sql = {SELECT COMPANY, LASTNAME, FIRSTNAME, MI, PHONE, EMAIL FROM } + _
conSrc.Metadata & { WHERE COMPANY = '}' & SQLQuote(companyName) & {'}

count = conSrc.Execute(sql, flSrc)
Do While conSrc.Fetch(flSrc)
  flDest.COMPANY = flSrc.COMPANY
  flDest.FNAME = flSrc.FIRSTNAME
  flDest.LNAME = flSrc.LASTNAME
  flDest.MNAME = flSrc.MI
  flDest.TELEPHONE = flSrc.PHONE
  flDest.EMAIL = flSrc.Email
  flDest.GetField(1).Flags = LCFIELDF_KEY ' Company name is a key for the dest table.
  flDest.GetField(2).Flags = LCFIELDF_KEY ' so is each part of the name.
  flDest.GetField(3).Flags = LCFIELDF_KEY
  flDest.GetField(4).Flags = LCFIELDF_KEY

  ' look up the record to see whether it already exists in the destination
  sql = {SELECT FNAME FROM } & conDest.Metadata & { WHERE COMPANY = '}' & _
  SQLQuote(companyName) & {' AND FNAME = '}' & SQLQuote(flDest.FNAME(0)) & {' AND LNAME = '}' & _
  SQLQuote(flDest.LNAME(0)) & {' AND MNAME = '}' & SQLQuote(flDest.MNAME(0)) & {'}
  'Print sql
  Set flTmp = New LCFieldList
  count = conDest.Execute(sql, flTmp)
  If count < 0 Then
    ' can't tell yet how many records matched
    count = conDest.Fetch(flTmp)
  End If

  ' if the record already exists in destination, update it, else insert.
  If count = 0 Then
    'Print "insert " & debugstr(fldest, True)
    conDest.Insert flDest
  Else
    'Print "update " & debugstr(fldest, True)
    conDest.Update flDest
  End If
Loop

Function SQLQuote(src As String) As String
  Dim result
  result = Replace(src, {'}, {''})
  SQLQuote = result
End Function

The SQLQuote function shown here is something you might want to copy in cases, where you
do want or need to use the LCConnection.Execute method. In this case, though, Execute is
being used unnecessarily.

It could actually have been worse than it is, since we are using the Insert and Update
methods instead of coding those using Execute also. Even so, we can get some improvement
by replacing the Executes we do have with Select statements. The first Select isn't harming
us much since it's only done once, but the one inside the loop is bad news.

The Select inside the loop isn't really necessary at all, by the way, but we'll deal with that
issue later.
Progressive2: use Select instead of Execute

The second agent improves on the first by avoiding the use of the Execute method where possible. Most of the code is omitted to just highlight the differences. Code deleted from the Progressive1 agent is commented out and appears in italics. New code lines are shown in boldface.

Example 9-4  Progressive2 agent was changed to use Select instead of Execute

```vbscript
... Dim flKey As New LCFieldList(1, LCFIELD_KEY) Dim fCompanyKey As LCField

Set fCompanyKey = flKey.Append("COMPANY", LCTYPE_TEXT) fCompanyKey.Value = companyName

'sql = {SELECT COMPANY, LASTNAME, FIRSTNAME, MI, PHONE, EMAIL FROM } + _ 'conSrc.Metadata & { WHERE COMPANY = '} & companyName & {'}
'count = con被执行人.Execute(sql)
conSrc.FieldNames = "COMPANY, LASTNAME, FIRSTNAME, MI, PHONE, EMAIL"
count = conSrc.Select(flKey, 1, flSrc)
...
flDest.GetField(1).Flags = LCFIELD_KEY ' Company name is a key for the dest table.
flDest.GetField(2).Flags = LCFIELD_KEY ' so is each part of the name.
flDest.GetField(3).Flags = LCFIELD_KEY
flDest.GetField(4).Flags = LCFIELD_KEY

' look up the record to see whether it already exists in the destination

'sql = {SELECT FNAME FROM } & conDest.Metadata & { WHERE COMPANY = '} & companyName & {' AND FNAME = '} & flDest.FNAME(0) & {' AND LNAME = '} & SQLQuote(flDest.LNAME(0)) & {' AND MNAME = '} & flDest.MNAME(0) & {'}
'conDest.Execute(sql, flTmp)

Set flTmp = New LCFieldList
count = conDest.Select(flDest, 1, flTmp)
If count < 0 Then
' can't tell yet how many records matched
    count = conDest.Fetch(flTmp)
End If

' if the record already exists in destination, update it, else insert.
If count = 0 Then
    conDest.Insert flDest
Else
    conDest.Update flDest
End If
...
```

Besides the changes shown here, we deleted the SQLQuote function, which was unused. Overall, we've reduced the amount of code as well as making it faster.

A Select call takes as parameter a key list (flKey in this example). This key list, together with the Condition property of the LCConnection object, determine the WHERE clause of the SQL Select statement. In addition, the following LCConnector properties affect other parts of the SELECT statement:

- FieldNames gives the list of fields to retrieve; it appears right after the SELECT keyword. If this property is not assigned, LC does a SELECT *.
OrderNames, if assigned, appears in the ORDER BY clause of the SELECT statement. It may include ASC and DESC clauses.

The code demonstrates a search based on equality with the target value. For example, if the variable companyName contains “Jiffy Auto Parts”, then the SQL generated by the first SELECT call would use “WHERE COMPANY = ‘Jiffy Auto Parts’”. By concatenating the LCFIELDS flags, you can specify that the key

Of course, the above assumes that the connector you’re using connects to a system that uses standard SQL. The Notes connector, File connector, and Text connector do not support SQL — but they do implement the Select method in their own ways. Different relational databases also have small differences in their SQL languages. So besides performance, another reason to avoid Execute where possible is to make your code less dependent on what type of data source you’re connecting to.

Progressive3: Remove unneeded SELECT

We next examine the code inside the loop for detecting whether a destination record already exists and updating or inserting, as appropriate.

This is more database activity than we really want to do. Instead, we can try an update and test the result to see how many records were updated. If zero, we should do an insert instead. Here’s the code, with changes notes ad in the previous section.

Example 9-5   Progressive3 agent removes unneeded Select

```
'Set flTmp = New LCFieldList
'count = conDest.Select(flDest, 1, flTmp)
'If count < 0 Then
'  ' can't tell yet how many records matched
'  count = conDest.Fetch(flTmp)
'End If
'If count = 0 Then
'  conDest.Insert flDest
'Else
'  conDest.Update flDest
'End If

'Try an Update. If this doesn't work use Insert.
If conDest.Update(flDest) = 0 Then
  conDest.Insert flDest
End If
```

Anytime you can remove an unnecessary database operation, you’ll get a big boost in performance.

If you’re experiencing performance difficulties, think about how you can reorganize your code to minimize I/O. The above is a very simple example, but suppose we made it more complex — in addition to updating the entries in the destination table that match the source records, you want to remove any destination records whose keys do not appear in the source. This is basically what the LEI Replication activity does. Now, obviously, you could do that as a two-step process. First, loop through the destination records, deleting any whose keys don’t appear in the source. Then, loop through the source records, updating the destination as we’re doing here. (Or, you could just start out by deleting all the destination records, then insert everything from the source, but that assumes that the destination contains only fields copied from the source and that the destination doesn’t use any events that trigger when these record are deleted or inserted).
The process described above, however, means that you have to individually lookup every key twice. That's a lot of Select and Update calls. Instead, you could just do two selects — one in the source and one in the destination — returning all the records you need in sorted order by their keys. Then, repeatedly fetch from one result set or the other depending which last returned the “smaller” value before, and decide whether to update, insert, or delete from the destination based on that.

**Progressive4: Use shared LCFields instead of copying data**

The target of our next performance improvement are the statements inside the loop that copy data from one fieldlist to another fieldlist. For instance,

```plaintext
flDest.FNAME = flSrc.FIRSTNAME
```

There are two things wrong with this. First, while you can refer to a field value using the “dot” notation `fieldlist.fieldname`, executing this requires the LSX to search the fieldlist for a field with that name at runtime. It’s more efficient if you remove this lookup from outside the loop. Second, since the data are not being manipulated, just copied from table A to table B, there’s no reason to spend the CPU cycles to convert the source data into a LotusScript datatype (variant array of strings) and then back again into an LEI datatype. Instead, we can arrange for the data to be deposited in an LCField object on fetch, and read from the same LCField when the data are Updated or Inserted into the other table. This is done by making the one field a member of both the input and the output fieldlists. There are several LCFieldlist methods that can be used for this: Map, MapName, CopyRef and IncludeField. You can read about them in the Domino Designer Help. In our example, we use MapName.

In the example code below, italics show code commented out from the previous version, and bold shows new or changed code.

**Example 9-6  Progressive4 avoids copying field data by mapping two fieldlists to use the same fields**

```vbnet
'Call flDest.Append("COMPANY", LCTYPE_TEXT)
'Call flDest.Append("LNAME", LCTYPE_TEXT)
'Call flDest.Append("FNAME", LCTYPE_TEXT)
'Call flDest.Append("MNAME", LCTYPE_TEXT)
'Call flDest.Append("TELEPHONE", LCTYPE_TEXT)
'Call flDest.Append("EMAIL", LCTYPE_TEXT)

Dim flKey As New LCFieldList(1, LCFIELDF_KEY)
Dim fCompanyKey As LCField
Set fCompanyKey = flKey.Append("COMPANY", LCTYPE_TEXT)
fCompanyKey.Value = companyName

conSrc.FieldNames = "COMPANY, LASTNAME, FIRSTNAME, MI, PHONE, EMAIL"
count = conSrc.Select(flKey, 1, flSrc)

' share lcfield objects between the source and destination fieldlists.
Call flDest.MapName(flSrc, "COMPANY,FIRSTNAME,MI,LASTNAME,PHONE,EMAIL", _
"COMPANY,LNAME,FNAME,MNAME,TELEPHONE,EMAIL")

' Set the key flags outside the loop -- more efficient
flDest.GetField(1).Flags = LCFIELDF_KEY ' Company name is a key for the dest table.
flDest.GetField(2).Flags = LCFIELDF_KEY ' so is each part of the name.
flDest.GetField(3).Flags = LCFIELDF_KEY
flDest.GetField(4).Flags = LCFIELDF_KEY

Do While conSrc.Fetch(flSrc)
  'flDest.COMPANY = flSrc.COMPANY
  'flDest.FNAME = flSrc.FIRSTNAME
```
'flDest.LNAME = flSrc.LASTNAME
'flDest.MNAME = flSrc.MI
'flDest.TELEPHONE = flSrc.PHONE
'flDest.EMAIL = flSrc.Email

' if the record already exists in destination, update it, else insert.
If conDest.Update(flDest) = 0 Then
    conDest.Insert flDest
End If

Loop

The Fetch operation does not create a new set of LCField objects — it just loads new field values into the existing objects. The fields are known by different names in the two fieldlists — for instance, FIRSTNAME in the source fieldlist is called FNAME in the destination fieldlist — but both refer to the same memory. So, each time through the loop, the code can just Fetch from the source and Update or Insert into the destination.

Figure 9-2   Diagram of data relationships in merged fieldlists

Notice also:

➤ The fields are not in the same order in the two fieldlists.

➤ We’ve designated certain fields as key fields by setting their Flag values to LCFIELDF_KEY. Since we’re not in writeback mode, the Update method uses the key values to determine which records to update. The “key” setting of a field does not affect the Fetch operation.

Progressive5: Multi-row fetch

The final version of this agent gets another little performance boost by reading up to 50 rows of data in a single operation. Depending on the connector or metaconnector you’re using, this might not run any faster, but with some connectors it’s substantially faster, and particularly when the server is accessed over a network. The reason it’s faster to fetch 50 rows at once, is that you avoid the network and processor overhead of sending 50 separate commands to the database server and processing 50 separate replies. The amount of useful data received is the same, but the amount of “overhead” is much less.

Example 9-7   Progressive5 agent uses multi-row fetch to speed operations

... Dim flSrc As New LCFieldList(50)
    Dim flDest As New LCFieldList(50)

... Dim i%
    Do
        count = conSrc.Fetch(flSrc, 1, 50)
For i = 1 To count
' if the record already exists in destination, update it, else insert.
If conDest.Update(flDest, i, 1) = 0 Then
    Call conDest.Insert(flDest, i, 1)
End If
Next
Loop Until count = 0

To use a multi-row fetch, you must have a multi-row fieldlist allocated to contain the maximum number of rows you want to work on at once. Specify the size in the New method of LCFieldlist; 50 in this example. To maximize speed, you may want to tweak this number, but we've found 50 is optimal for most database back-ends (100 for iSeries). If you go much higher than that, you're not gaining performance, just wasting memory.

The Fetch method takes optional arguments that tell how many rows you want to fetch. It will fetch the lesser of the number of rows remaining in the result set, and the number you supply. If there are no rows left to fetch, it returns a count of 0.

9.7 Tips and tricks

This section highlights helpful hints and techniques which you can use when developing or troubleshooting LC LSX programs.

9.7.1 Using the Trace metaconnector

The Trace metaconnection in LEI helps you figure out what's really happening when something goes wrong. You can trace your LC LSX programs also, and see what methods are called in what order. For scripted activities, you can also see the SQL generated in response to your method calls.

To use Trace, create your LCConnection object with the constructor argument "trace". Set the ConnectorName property of the connection to the actual database type you want to access, for example, “db2”. Use the other properties of the trace connection to control the trace output, as follows:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectorName</td>
<td>String. The name of an LC Connector, for example, “db2”. A connection document name may not be used here.</td>
</tr>
<tr>
<td>OutputRecord</td>
<td>Long. Use the logical “or” of the following values to control output:</td>
</tr>
<tr>
<td></td>
<td>1: Write trace to activity log (only works in a Scripted Activity agent).</td>
</tr>
<tr>
<td></td>
<td>2: write trace to a text file.</td>
</tr>
<tr>
<td></td>
<td>4: write trace to “standard output” (has no effect if agent is run in Notes client).</td>
</tr>
<tr>
<td></td>
<td>8: add timestamps to output — useful when trying to improve performance.</td>
</tr>
<tr>
<td>LogFilename</td>
<td>If the OutputRecord property specifies file output, this is the name of the file to write the output into. If the file already exists, the output is appended to it. The filepath must be valid where the agent code executes.</td>
</tr>
</tbody>
</table>

Suppose you want to record log output with timestamps to a file while you work with an ODBC data source on a Windows 2000 server. If your original code contains:

```vbnet
Dim con As New LCConnection("odbc2")
```
You might change it as follows:

```vba
Dim con As New LCConnection("trace") ' assumes this is not a scripted activity!
Const LOGFILE = "C:\Lotus\Domino\AgentLog.txt" ' or whatever name you choose
con.OutputRecord = 2 + 8 ' log to file, add timestamps
On Error Resume Next ' ignore error deleting file.
Kill LOGFILE ' remove old logfile contents.
On Error Goto 0 ' resume normal error handling
con.LogFilename = LOGFILE
con.ConnectorName = "odbc2"
```

Any properties that are not used by the Trace metaconnector, it passes on to the base level connector (ODBC in this case). So, you can assign other connector properties as appropriate for that connector, and the rest of your code should work without any changes.

**Note:** LEI scripted activities use a connection document name in the New LCConnection call, not a connector name. So, to use trace in a scripted activity, you must create a Trace metaconnection document in the LEI Administrator and use that instead of the original connection document.

The trace output contains a list of the database operations performed (Insert, Select, Fetch, ...), the arguments to those functions, and the values they return. For instance, after every Fetch the trace displays the fieldlist that was fetched.

The trace does not include the actual SQL statements used by the connector. There's not a way to get the SQL into the trace output. However, for a Scripted activity you can get the SQL into the activity log document along with the trace output. To do this, use the statement `con.TraceSQL = True` (where con is your LCConnection object). This will only work in a Scripted activity, since LotusScript code otherwise doesn't have an activity log. If you use the Trace metaconnector and TraceSQL in a scripted activity agent, you can see which SQL statements were generated in response to which method calls.

**Note:** The SQL generated by the Lotus Connectors contains “placeholder” characters instead of the actual data. For example, the statement recorded in the log may read:

```
UPDATE EXTENSION.WEEDS SET INVASIVE=? WHERE PLANTNAME=?
```

The database server inserts the values from a list of data fields passed separately in binary format, saving time compared to having to convert these values to text and back to dates, numbers, and so on at the server end.

If you combine TraceSQL with Trace metaconnector, you’ll be able to see what values were in the fieldlist passed to the Update call, and figure out what values were used in place of those ? characters.

### 9.7.2 Map by Name

In 9.6.1, “Performance code samples” on page 212, the code depends on the fields in the destination fieldlist being in the same order as the fields in the relational table. If they were in a different order, the Update and Insert commands would assign values based on the positions of fields within the fieldlist — the first field into column 1, the next into column 2, and so on, ignoring the names within the fieldlist.

If you know exactly what fields are in the table in what order, and you’re sure that the table definition isn’t going to change, it's marginally more efficient to write your code this way. However, if you think they might change, or if you want to skip over some fields, it makes
sense to use the LCConnection.MapByName property to make LEI read and write fields based on their names rather than their positions in the list.

The statement `con.MapByName = True` turns on name mapping for all transactions done using the LCConnection object `con`. `False` is the default value.

### 9.7.3 Sorting data

There are two ways to sort data in your result set: you can use the OrderNames property of the LCConnection, or you can use an Order metaconnector. OrderNames is by far the more efficient method, but the metaconnector lets you select which character codepage will supply the collating sequence, which is occasionally helpful.

The OrderNames property of a relational connection is copied directly into the “ORDER BY” clause of the SQL Select statement. Therefore, you may use the ASC and DESC keywords or whatever other options the specific version of SQL supports. For instance, to retrieve all records in the current metadata sorted by FlounderID (ascending) and then by ScaleSize (descending), you might write the following:

```vbnet
Dim con As New LCConnection("db2")
...
con.OrderNames = "FlounderID ASC, ScaleSize DESC"
count = con.Select(Nothing, 1, flFishes)
```

To use the Order metaconnector, create an LCConnection object using “order” as the constructor argument. Assign the additional properties shown in Table 9-6, then assign the properties of the “back-end” connection, such as Userid, Password, and Metadata. Any properties you assign that are not shown in the table, are passed through to the actual connection.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectorName</td>
<td>The connector type that you would like to sort data from, for example, “db2”. This is the value that you would use in the New method if you weren’t using Order.</td>
</tr>
<tr>
<td>OrderNames</td>
<td>The names of fields you’d like to sort by. The ASC and DESC keywords may not be used here.</td>
</tr>
<tr>
<td>SortDescend</td>
<td>Boolean, default False. Set to True to sort in descending order. Note that this applies to all fields mentioned in OrderNames — there’s no way to sort some ascending and some descending.</td>
</tr>
</tbody>
</table>
| SortText            | Integer. Controls the technique used to sort the data:  
                      - 0: Binary sort order based on the character code number  
                      - 1: Case sensitive  
                      - 2: Non-case sensitive  
                      If you use binary sorting (the default), the SortCharacterSet option is ignored. |
| SortCharacterSet    | Text, containing the name of the codepage to use for sorting, for example, “ASCII”, “ISO88596”, or “NATIVE”. This is ignored if you use SortText option 0. The default is “NATIVE”. Refer to “List of Supported Character Sets” in the Domino Designer 6 Help for a complete list. |

For instance, instead of:

```vbnet
Dim con As New LCConnection("db2")
con.Metadata = "S_CARGO.BILLS"
```
con.MapByName = True
...

You might write:

Dim con As New LCConnection("order")
con.ConnectorName = "db2"
con.OrderNames = "SENDER_ID"
con.SortText = 2 ' not case sensitive
con.SortCharacterSet = "UNICODE"
con.Metadata = "S_CARGO.BILLS" ' assign any remaining properties as you normally would.
con.MapByName = True
...

Use an order metaconnector when the “ORDER BY” clause of the database doesn’t support the kind of sorting that you need, and in particular when you need data from different sources to sort in the exact same order, and one of them is DB2 running on an AS400 with the EBCDIC character set, and the other is not.

Restriction: Writeback does not work in conjunction with the Order metaconnector. Since the intermediate layer fetches all the records and then sorts them in memory, the record you just fetched in your program is generally not the same one that was last fetched from the back-end system.

We have also observed some problems doing multiple-row fetches with this connector. (In any case, there is no performance benefit to using the multiple-row fetch with an Order metaconnector).

9.7.4 Updating key fields

The Update method of the Lotus Connectors are unable to generate SQL such as the following (at least not without a little help):

UPDATE table SET FIRSTNAME = 'Victor' WHERE FIRSTNAME = 'Victoria'

If you need to update a value in a key field, you can do it in one of three ways:

➢ Use a writeback update (9.3.7, “Update a record using writeback” on page 201). However, this may be inefficient because it only updates one record at a time.
➢ Write the SQL yourself and use LCConnection.Execute to run it.
➢ Write the SQL for just the WHERE clause yourself, and use LCConnection.Condition property to get it into the statement.

Between the last two options, it may not seem to you that there’s much difference, but if you must update a fifty-column table or one containing binary columns, you might find the Condition property preferable.

Here’s an example of using the Condition property. The newValues fieldlist contains the values you want to write to the table, including a different value for FIRSTNAME, and the variable oldFirstname contains the name that’s already stored in the table.

con.Condition = '{FIRSTNAME = '} & SQLQuote(oldFirstname) & {'
Call con.Update(newValues)

The SQLQuote function used in this example is shown in Example 9-3 on page 212. It converts single quote characters to two single-quotes, which prevents syntax errors due to incorrect quoting.
For a more complete discussion of WHERE clauses and how the connectors generate them, see 9.3.4, “Select records that meet certain criteria” on page 198.

9.7.5 Offload database access to server

Some applications call for database access in real time in response to events on the user’s workstation. For instance, suppose you want to post a transaction in a relational table every time a Notes document is edited and saved.

Since the LC LSX comes with Notes, you can run LC LSX code on the user's Notes client. The problem is that the code will want to use a relational database, so the user's workstation needs access to the database — it must have a database client installed and configured to allow access to the data source. This could be ODBC if the data source supports ODBC, but the workstation must have a data source configured to reach the database.

This is not a problem for the LSX if it's not a problem for you. But most organizations prefer to avoid the administrative overhead of making sure that every user’s workstation has database connectivity installed, that it's at the right version, and correctly configured to connect to all the databases the user might need to connect to.

So you might prefer to have all database access happen on the server, so that you only have to install and configure the database client once. There are different ways to make this happen.

Use Advanced RealTime

Depending what you need to do on the server, you may be able to minimize the amount of code you need to write by taking advantage of LEI’s Advanced RealTime capability. For instance, in the example above, where you want to create a transaction whenever a document is edited, you might use a Virtual Fields activity with only the Update event enabled. If just writing certain fields to a table doesn’t satisfy the requirement, you could use fill in a stored procedure name in the Event Options tabs of the activity form, and write code on the server to do whatever needs doing.

One disadvantage to this approach; if the user works in a local replica, the database operation will not happen. If you’re using the Update or Create event, the Virtual Fields activity will trigger as soon as the new data replicates to the server, however.

So you can use this approach in a completely offline mode, supporting mobile users — provided that you don’t need the data from the server right away.

Batch mode updates to local caches

If your application requires you to retrieve information from the server — for instance, a keyword list lookup for a field — consider whether you could store a copy of the information in a Notes profile document or other locally available storage. If you expect the keyword selections to change infrequently, perhaps it would be good enough to have an agent that runs nightly to update the profile. Users of local replicas would receive the change by replication.

Likewise, in the “transaction on save” example above, if there’s not a need for transactions to get in to the database the instant they are entered, you might write a server agent that runs “After documents are created or modified,” and creates the transactions for those documents. Once again, remote users could work offline and their changes would be processed when they replicate their databases (or within a short time after replication).
NotesAgent.RunOnServer
The RunOnServer method lets a LotusScript program run a server agent on the server and
wait for it to finish. The LEI Administrator application uses this technique, for instance, to let
you browse lists of metadata on a DB2 database without having the DB2 software installed
locally.

You can pass parameters into the agent by passing it a note ID as a parameter; the agent
uses the ParameterDocID property to read the note ID, and
NotesDatabase.GetDocumentByID to locate the document. It can send values back to the
caller by updating fields in the document and saving those changes.

For an example of RunOnServer, you might look at the (SQL Executive) agent and the
SQLExecute form in the sample database. The form contains an action (Execute) which
stores data in a parameter document and calls the agent. The agent updates the parameter
document with the results it reads from the database, and the action then uses that document
to display updated information to the user. This agent works by opening the actual document
on screen, but you could also write code to open the document “in memory” and pull out the
returned information.

One drawback to RunOnServer is that it only works to call a server agent. The function is not
usable on an agent in a local replica. There is a work-around for this: the code that calls the
agent can check whether it’s running in a local replica, try to locate the server replica of the
same database, and call the agent in the server replica. This is a little more complicated, but
it will work provided there is a network connection to the server available.

Web Service
Using LotusScript, you can write a Web service on your server that will accept requests via
HTTP, parse out arguments, use the LC LSX to carry out database operations, and return
XML data to the sender.

At the other end, of course, you must create and send the request, and parse the XML result
to get the returned data, if any.

For instructions on creating both ends of this connection, refer to “Web Services” in the
Domino Designer Help, where you will find example code and links to other resources.

9.8 LEI power programming

The examples in this section include some applications that are more advanced, and some
that are particularly useful in conjunction with LEI activities. Not all of these examples use the
LSX — some of them take advantage of the Virtual Realtime features of LEI to access or
update relational data by reading and writing Notes documents that are being managed by
virtual activities.

9.8.1 LEI Programming 1: Transfer data from profile documents

This is the code for the scripted activity example Update Keyword Lists, from 7.5, “Scripted
activity” on page 125. The script reads two multi-value fields from a Notes profile document
and uses them to populate a relational table called KEYWORDS. There are three constants
at the beginning of the code that you can use to customize the connection name, table name,
and name of the index it creates against the Type column.

The code comments should be self-explanatory. This script does the following things that you
may want to look at.
Deletes all the records in a table.
Detects whether a table exist and if not, creates it.
Creates a relational index.
Inserts records.
Logs advanced debugging information, including the names and values of all fields in a fieldlist.
Uses connection pooling, including a final manual commit to avoid leaving “hanging” transactions in the pooled connection.

Example 9-8  Update Keyword Lists agent

Option Public
Option Declare
Uselxs "\lxlc"

Const CONNECTION_NAME = "Sample DB" ’ customize these three lines
Const KEYWORD_TABLE = "KEYWORDS" ’ you may instead specify owner.name syntax.
Const KEYWORD_INDEX = "Keywords_Ind"

Use "ServerAgentDebug" ’ contains debugStr function, handy for logging variable contents.

Function CreateKeywordFieldlist(Byval recordcount As Long) As LCFieldList
  Dim fKeytype As LCField, fKeyvalue As LCField
  ' This routine creates a new fieldlist that contains the fields that are
  ' supposed to be in the keyword table.
  Set CreateKeywordFieldlist = New LCFieldList(recordcount)
  Set fKeytype = CreateKeywordFieldlist.Append("Type", LCTYPE_TEXT)
  Call fKeytype.SetFormatStream(LCSTREAM_TRUNCATE, 20, LCSTREAMFMT_NATIVE)
  fKeytype.Flags = LCFIELDF_NO_NULL + LCFIELDF_TRUNC_DATA + LCFIELDF_KEY
  Set fKeyvalue = CreateKeywordFieldlist.Append("Value", LCTYPE_TEXT)
  Call fKeyvalue.SetFormatStream(LCSTREAM_TRUNCATE, 70, LCSTREAMFMT_NATIVE)
  fKeyvalue.Flags = LCFIELDF_NO_NULL + LCFIELDF_TRUNC_DATA
End Function

Sub HandleError(lcSes As LCSession)
  ' Called in case an error occurs, this routine logs the Notes error code and message.
  ' Then for good measure, if the error was an LC error, it logs the LC error also.
  Dim stat As Long, msg As String
  Call lcSes.LogText(0, "Error " & Err & " line " & Erl & ": " & Error, stat)
  stat = lcSes.GetStatus
  If stat Then
    Call lcSes.Log(stat)
  End If
End Sub

Sub Initialize
  On Error Goto trap
  Dim lcSes As New LCSession("any name you choose")
  lcSes.ConnectionPooling = True
  Dim con As New LCConnection(CONNECTION_NAME) ’ load a connection document from LEI Admin
  Dim flKeylist As LCFieldList
  Dim fKeytype As LCField, fKeyvalue As LCField

  Dim ses As New NotesSession
  Dim profile As NotesDocument
  Dim values As Variant

  con.Metadata = KEYWORD_TABLE

  Dim ses As New NotesSession
  Dim profile As NotesDocument
  Dim values As Variant

  con.Metadata = KEYWORD_TABLE
con.MapByName = True  ' match fields by name rather than by position.
con.Connect
  ' show all the connection properties in the log.
lcSes.LogText LCLOGSTREAMF_EVENT, "Connected: " & debugstr(con, True), 0

' Step 1: define the list of fields we want to use.
Set flKeylist = CreateKeywordFieldlist(1)  ' this is the list of fields in the metadata.
  ' alternately, to read the fieldlist from the actual table, we could do:
  'Set flKeylist = New LCFieldList(1)
  'Call con.Select(Nothing, 1, flKeylist)
  ' However this only works if the table already exists, and we don't want to assume that.

' Step 2: Remove all data currently in the keyword table.
On Error LCFAIL_INVALID_METADATA Goto createTable
  ' if table isn't there when we try to use it, go create it.
Call con.Action(LCACTION_TRUNCATE)

FetchNotesData:
  On Error Goto trap
  ' Step 3: get the lists from the profile document.
  Set profile = ses.CurrentDatabase.GetProfileDocument("MasterProfile")
lcSes.LogText LCLOGSTREAMF_EVENT, "Cleared old keyword list.", 0
  ' For ease of reference, get a pointer to the two fields in the fieldlist.
  Set fKeytype = flKeylist.Lookup("Type")
  Set fKeyvalue = flKeylist.Lookup("Value")

  ' Step 4: insert "specialties" values into table.
  values = profile.GetItemValue("Specialties")
  If values(0) <> "" Then
    ' there is a list of specialties in the profile document.
    fKeytype.Value = "SPEC"
    ' for each entry in profile's list of key values, insert record in relational table.
    ' Note: we could dimension a fieldlist large enough to insert all values at once,
    ' for better performance, but this is good enough for a short list.
    Forall keyval In values
      fKeyvalue.Value = keyval
      ' log the data being inserted.
      Call lcSes.LogText(LCLOGSTREAMF_EVENT, 
                     {inserting: } & debugstr(flKeylist, True), 0)
      Call con.Insert(flKeylist)
    End Forall
  End If

  ' Step 5: insert "equipment" values into table.
  values = profile.GetItemValue("EquipName")
  If values(0) <> "" Then
    ' there is a list of equipment in the profile document.
    fKeytype.Value = "EQUIP"
    Forall keyval In values
      fKeyvalue.Value = keyval
      ' log the data being inserted.
      Call lcSes.LogText(LCLOGSTREAMF_EVENT, 
                     {inserting: } & debugstr(flKeylist, True), 0)
      Call con.Insert(flKeylist)
    End Forall
  End If

  ' Step 6: commit changes -- don't leave uncommitted transactions in pooled connection.
Call lcSes.LogText(LCLOGSTREAMF_EVENT, "Committing changes", 0)
Call con.Action(LCACTION_COMMIT)
9.8.2 LC LSX Example 2: Update Virtual Fields in Notes views

One drawback to using Virtual Fields: if you retain copies of some virtualized fields for use in Notes views, the copies don’t stay up to date with the relational data. To see data that’s up to
date, you must actually open the Notes document (in a window or in preview mode). See
Figure 8-12 on page 167 for an example.

The script in this example runs as a scheduled agent or Scripted Activity on the Domino
server, or from a user’s workstation on demand (though this may be slow for large
databases). The agent scans a Notes view for key documents that are out of date with their
corresponding relational table rows. The Virtual Fields activity must be running at the time.

The algorithm is quite simple, because when you access an existing NotesDocument in
LotusScript, the Virtual Fields activity activates, filling in the values of the virtual fields. The
agent requires a view that contains a column for each Virtual Field you want to keep up to
date; the view need not be sorted. The agent scans view entries from first to last, comparing
the Columnvalues in each row with the field values in the document. When it finds one that’s
different, it just saves that document. This writes the updated field values into the Notes
storage. The RDB is also updated, if the activity intercepts the update event, but its value
doesn’t change because the field values are those just read from the RDB.

Tip: If there’s some reason you don’t want to update the relational table, note that the
Virtual Fields activity form contains an option in the Event Options / Update tab, “Only
Update Changed Fields”. Use this option to avoid an unnecessary write. There is a
performance penalty to doing this test, but you may have to, to avoid triggering “record
modified” events in the relational database server.

This agent will work with forms that employ multiple Virtual Fields activities, provided that the
view displays the retained fields for those activities also.

Example 9-9  Agent to keep Virtual Fields key documents up to date

Option Public
Option Declare
' use Option Nocase if you don't care about uppercase/lowercase differences in the data.

Const VIEWNAME = "VFAgentFieldUpdate"

Function isEqual(a As Variant, b As Variant) As Integer
    Dim i%  
    If Isarray(a) Then  
        If Isarray(b) Then  
            If Ubound(a) <> Ubound(b) Then  
                Exit Function ' returning default False value  
            Else  
                For i = 0 To Ubound(a)  
                    If a(i) <> b(i) Then Exit Function  
                Next  
            End If  
        Else  
            If Ubound(a) > 0 Then Exit Function  
            If a(0) <> b Then Exit Function  
        End If  
    Elseif Isarray(b) Then  
        If Ubound(b) > 0 Then Exit Function  
        If a <> b(0) Then Exit Function  
    Elseif a <> b Then  
        Exit Function  
    End If  
    isEqual = True  
End Function

Sub Initialize  
Dim ses As New NotesSession  
Dim db As NotesDatabase  
Dim vu As NotesView  
Dim doc As NotesDocument, nextDoc As NotesDocument  
Dim vuval As Variant, docval As Variant  
Dim fldname As String  
Dim column As NotesViewColumn  
Dim colno As Integer  
    Set db = ses.CurrentDatabase  
    Set vu = db.GetView(VIEWNAME)  
    vu.AutoUpdate = False ' more efficient to not update view contents until all done.  
    Set doc = vu.GetFirstDocument()  
Do Until doc Is Nothing  
    Set nextDoc = vu.GetNextDocument(doc)  
        ' for each column in the current row  
    For colno = 0 To vu.ColumnCount - 1  
        Set column = vu.Columns(colno)  
            ' if the column contains a field value (i.e. not a formula)  
        If column.IsField Then  
            ' compare the column value with the field of the same name.  
            ' When reading field from NotesDocument, VF kicks in to supply value from RDB.  
            vuval = doc.ColumnValues(colno)  
            docval = doc.GetItemValue(column.ItemName)  
            If Not isEqual(vuval, docval) Then  
                ' view doesn't match document; re-save with new values just read by VF.  
                Call doc.Save(False, False)  
                Exit For  
            End If  
        End If  
    Next  
End If
9.8.3 LC LSX Example 3: Fix unvirtualized documents

As described in 8.6.7, “Domino replication involving virtual documents” on page 178, if documents are created on or replicated to the Domino server while a Virtual Documents activity is not running, they will be created as “regular” Notes documents even if they use a form that’s supposed to be virtualized.

To correct this after the fact, you must delete and re-create the Notes document. Like the previous example, this code relies on the activity to be running. When this agent uses the NotesDocument methods to create a new document using the virtualized form, the activity will trigger, and virtualize the new document.

You can tell whether a document is virtual by examining its 32-bit note ID; the bit at the 2^30 position is set if the document is virtual. This agent finds each such document, copies its fields to a new document, saves the new document and deletes the original.

Since the database may contain a great many virtual documents and only a few that failed to be stored in the table, it’s inefficient to examine each document and check its Notes ID, every time the agent runs. To find such documents efficiently, we’ve created a view, Virtual Docs\Failed Virtualization, whose selection formula finds only those documents that the agent must process. Here’s the formula:

```
SELECT Form = "Agent Record VD" & @Matches(@NoteID; "NT{0-3}"")
```

The agent simply scans through this view, replacing each document with a copy. The code is shown below.

**Example 9-10 Repair Failed Virtualizations agent**

```
Option Public
Option Declare
Uselsx "*lsxlc"

Sub Initialize
    Dim ses As New NotesSession
    Dim db As NotesDatabase
    Dim vu As NotesView
    Dim doc As NotesDocument, nextDoc As NotesDocument, newVDoc As NotesDocument
    Set db = ses.CurrentDatabase
    Set vu = db.GetView("FailedVirtualization") ' alias
    vu.AutoUpdate = False
    ' we'll delete all docs in the view; don't wait for the view to refresh each time.
    Set doc = vu.GetFirstDocument()
    Do Until doc Is Nothing
        ' pre-fetch next document because we will delete current document.
        Set nextDoc = vu.GetNextDocument(doc)
        ' Copy all the fields from the existing document to a new document.
        Set newVDoc = doc.CopyToDatabase(db)
        If (newVDoc Is Nothing) Then
            MsgBox "Repair failed for doc ID " & doc.NoteID & _
                " -- probable invalid field values.", 0, "Repair Failed Virtualizations"
        Else
            If Not newVDoc.NoteID Like "[4-9A-F]???????" Then
                ' the new copy is not virtual!
```

Set doc = nextDoc
Loop
End Sub
9.9 Notes client coding considerations

Previously, the LC LSX was not installed on the Notes client. Starting with Notes 6, however, the LC LSX is installed if you enable Domino Enterprise Connection Services (DECS) while installing the Notes/Domino Designer client. You don’t have to enable DECS locally, but installing with this option gives you copies of the libraries that you need to compile or run LC LSX code on your workstation.

If you use the LC LSX in a Notes client application, all code that runs on the workstation and uses database functionality, must have the database client software installed locally on the workstation. Any data sources used must also be defined locally.

LEI Advanced Realtime activities get around this problem by doing the database access on the server, so if you can use Virtual Fields or Virtual Documents to access the information you need, then you can avoid having to install database access locally.

Another strategy (used by the LEI Administrator), is to write a server agent that does the required database access, and invoke that agent using the RunOnServer method. The agent stores the results in fields of a Notes document, and the client reads the results and works with them. You can see an example of this approach in the sample database that comes in the Additional Materials; the form SQLExecute uses RunOnServer to call the agent SQL Exec, taking advantage of database clients and data sources available on the server but not necessarily on the user’s workstation.

For Domino servers running the HTTP task, you can also write agents to be run with an ?OpenAgent URL command. Such an agent, which could even be a Web service, would accept information defining its task either from the URL or as a Post request, and return data by writing an output Web page or XML output to the caller. LotusScript can call such an agent using the NotesDatabase.GetDocumentByURL method, or using LS2J and the XML classes of Domino 6.

9.10 Other APIs

This section highlights additional Application Programming Interfaces, other than the LC LSX, and which can be used when accessing relational data and working with LEI activities.

9.10.1 LS:DO

LotusScript contains another LSX, the LSXODBC, which provides an alternate interface for working with relational databases via the ODBCConnection class.

The characteristics of the LS:DO are:
Allows access to ODBC data sources only.

There are no Select, Update, Insert... methods — only Execute. The calling program must create the SQL statement.

There's not a way to associate fields in one data source with fields in another, as you can do in the LC LSX by creating multiple references to the same LCField object.

Basically, then, there's nothing you can do using LS:DO that you can't do using the LC LSX, faster and usually with less code. Plus there's plenty you can do with the LC LSX that you can't do with LS:DO. This interface is retained for compatibility.

### 9.10.2 JDBC

Notes/Domino supports Java programming, both through a built-in Java development environment and by letting you call Java code through the LS2J (refer to Domino Designer Help for details). Therefore you can use any available Java tools for accessing relational data, including JDBC, and you may want to if those tools are more familiar to you.

Java technology may also be helpful in other ways, particularly when you need a rapid query and response system on a Web server, where an applet is the natural choice. You can write a LotusScript agent to respond to queries the way a servlet does, but the agent is not memory-resident; it has to load into memory and do its initialization every time the server receives a URL request for that agent. You can save some time by using connection pooling (refer to 9.5, “Connection pooling” on page 208) but a servlet will still get you better performance.

The LC LSX has advantages when adding tasks to existing LEI activities, since you can write scripted activities to use the same connection documents used by other activities. That gives you a single location to store connection information and passwords. As noted previously, the LC LSX also has the advantages of not requiring you to calculate SQL expressions in most cases, the ability to use the full capabilities and performance of native database drivers, and to set up linked fieldlists to automate copying from source to destination within efficient C code instead of copying data using an interpreted language.

### 9.10.3 @DbLookup and @DbColumn functions

The macro functions @DbLookup and @DbColumn support an “ODBC” syntax, which you can read about in the Domino Designer Help. These commands are useful when it's not convenient to use LotusScript or Java; for instance, they're excellent for pulling in a list of keywords in a Domino Web form. Like all of these tools including the LC LSX, their usefulness for the Notes client is somewhat restricted by the fact that the Notes client workstation must have software installed and a data source defined locally to access this information.

### 9.11 Additional information about the LC LSX

For a reference on using the LotusScript classes for Lotus Connectors, use the Lotus Connector Lotus Script Extension Guide (lsxlc6.nsf). This guide is available for download from the Lotus Developer Domain Documentation Library Web site at:

http://www.lotus.com/ldd/notesua.nsf

Note: With Notes 6, the information in this guide is also available in the Domino Designer Help. You may still choose to use the separate documentation database to narrow your searches.
When you install LEI, you also have an option to install a sample application, which includes a couple of scripted activity agents in the leiv1t6.nsf database.

Lotus Developer Domain’s “Sandbox” has downloads available that show sample LC LSX code. Go to:

http://www-10.lotus.com/ldd/sandbox.nsf

Search for “LC LSX”.
Removing Lotus Enterprise Integrator (LEI)

This appendix covers the removal of LEI from the iSeries platform. Removal of any software is never as simple as just removing the code off the desired workstation or server. There are a number of issues that require addressing prior to physical removal, otherwise users of the environment could find access to systems that are expected to be available are no longer responding.

Issues covered in this appendix are not discussed in detail as each situation and response will be different at each site. The topics discussed include:

- Considerations that require addressing prior to removal of a software resource
- The process of removing LEI 6.0.1 from the iSeries platform
Considerations prior to removing LEI

Whether LEI has been operating as a pilot environment, or has been employed as an enterprise solution for a number of years there are issues that must be resolved prior to removing the LEI infrastructure. Issues that are to be resolved include:

- How is access to data, presently provided through LEI, going to continue to be provided?
- Is this server part of an LEI cluster?
- Is it envisioned that the previous Domino Enterprise Connection Services (DECS) environment is going to be reinstated?

Access to data

The removal of LEI from a server is likely to have issues in relation to the data being accessed by users or other servers. Issues that should be reviewed include:

- Is the information being accessed going to continue to be available through other means?
- Do users need to be informed of removal of access to information?
- Have users successfully been redirected to a different LEI server for the information?
- Is there a requirement for restoring Domino Enterprise Connection Services (DECS)?
- Are the new measurements in place before removing LEI?

LEI clustered environment

The first issue with LEI clustering that is required to be identified is whether the LEI server being removed is administered by a remote administrator, or whether it is the administration server administering the other LEI servers in the cluster.

If the server being removed is remotely administered by a different LEI server it is fine to remove the LEI software from the server. After removing the LEI software the configuration document for the removed LEI server will require deleting from the LEI Administrator database on the administration server.

The software can not be removed if the LEI server is identified as the administration server for an LEI cluster. The LEI server configuration documents are not able to be reconfigure to be administered by a different server.

Domino Enterprise Connection Services (DECS)

If DECS is desired to be reinstated, the operator will have to be aware of selecting the radio button to delete the LEI Administrator and LEI databases, displayed in Figure A-8 on page 239, followed by selecting the checkbox for enabling DECS as an addin task.

**Note:** This will reinstate the DECS administration database that was backed up at the time of installing LEI 6. This database is usually named decsadmin.nsf. The process does not migrate new Advanced RealTime activities or connection documents back to the DECS administration database, they will require to be recreated. The only connection documents and Advanced RealTime activities that will be present are the ones that were present at time of backup.
Process of removing LEI

Removing LEI is performed on a Win32 based workstation, similar to the process of installation. On the product media used to install LEI, is a file called iSetupUninstall.exe. This file is used to initiate the Java based UninstallShield that will remove LEI from the iSeries server.

1. From a Win32 workstation, locate and run the uninstall program iSetupUninstall. The first screen presented is the window indicating that the Java Virtual Machine (JVM) is being prepared as shown in Figure A-1.

![Figure A-1](Java Virtual Machine - Informational message)

2. When the JVM is prepared, the setup for LEI uninstall welcome screen is presented as shown in Figure A-2. This is informing you that you are about to initiate the uninstall process for LEI. Click Next to continue.

![Figure A-2](Setup for LEI Uninstall on iSeries)

3. The setup wizard now confirms the IP address of your workstation as shown in Figure A-3. The IP address of the current workstation is entered by default, confirm that the address is correct and click Next to continue.
4. The next screen is primarily an informational screen. As displayed in Figure A-4 the status of setup for uninstall is presented. Do not click any buttons unless intending to quit the process of removing LEI.
5. A signon window as shown in Figure A-5 requests you to sign onto the iSeries server that LEI is being removed from. Enter the iSeries system name, the OS/400 profile user and password. Click OK to continue.

![Signon to the Server](image)

*Figure A-5  iSeries server sign on*

6. The setup wizard now opens a number of windows as displayed in Figure A-6. These windows include:
   - The Remote AWT Daemon controlling the connection between your workstation and the iSeries server.
   - A DOS window which is used to launch the Remote AWT Daemon.
   - The installation progress window (displayed as a grey window).
   - The uninstaller window (displayed as a white window).

   **Note:** These windows take a while to load, be patient. It is important that none of these windows are closed unless they are closed by the program itself or a prompt has displayed indicating that it is okay to close the window.
7. The UninstallShield window presents the user two options. The option to choose depends on the issues raised in “Considerations prior to removing LEI” on page 234. If you are choosing to leave the LEI Administrator databases, as displayed in Figure A-7, click Next to continue.

If the decision is to delete the LEI Administrator and LEI databases, as shown in Figure A-8, then you have to select whether to:

- Enable DECS as an addin task.
- Delete the sample and documentation databases.
- Delete the script vault database.

Click Next to continue.
8. The UninstallShield now confirms what will be deleted and where from the system displayed in Figure A-9. Click **Next** to continue.
9. The UninstallShield now begins the process of removing the files and displays an informational window to inform the operator that it is now uninstalling as shown in Figure A-10. Do not close this window.
10. When UninstallShield has completed removing LEI from the iSeries server it will display the successfully completed window shown in Figure A-11. Click **Finish**.

![UninstallShield - Completion](image)

**Figure A-11** UninstallShield - Completion

11. The setup wizard now displays that it has completed the uninstall environment, as shown in Figure A-12. Click **Finish**.

![Setup - Completion](image)

**Figure A-12** Setup - Completion
12. All windows should now be closed, any remaining open can be manually closed at this point.
13. Restart the Domino server.
Troubleshooting

This appendix discusses the options and tools you have for troubleshooting Lotus Enterprise Integrator (LEI) 6.0.1 activities.

This appendix discusses the following tools:

- Using the LEI log (leilog.nsf)
- Using the Domino log (log.nsf)
- Enabling log conflicts in replication activities
- Logging SQL statements
- Using the trace metaconnector
- Using Notes Server Diagnostic (NSD)
Using the LEI log (leilog.nsf)

The LEI log is the most straightforward tool available to troubleshoot an LEI activity. However, it offers the least information about the error. You can access the LEI log using the View Log button from any activity document (see Figure B-1) or the Current Activity Execution Log button from the activity views in the LEI Administrator (see Figure B-2).

![View Log button from an activity document](Figure B-1)

![Current Activity Execution Log button](Figure B-2)

When accessing the log button from the activity views in the LEI Administrator, you must select the activity document to review and then click the Current Activity Execution Log button.
For example, the highlighted activity document is Test Integrated Credentials. When the log button is clicked from either within the activity document or from the LEI Administrator activities view, the result will be as displayed in Figure B-3.

**Activity: Test Integrated Credentials**

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/20 03:08:38 PM - 05/20 03:08:38 PM</td>
<td>Activity started</td>
</tr>
<tr>
<td>05/20/2003 14:08:38</td>
<td>The connection document does not exist: Database = LEI Administrator Document = Test Integrated Credentials Document = DB2 Connection to EMPLOYEE.</td>
</tr>
<tr>
<td>05/20/2003 14:08:38</td>
<td>Activity terminated with an error</td>
</tr>
</tbody>
</table>

**Statistics**

- **Elapsed Time:** 00:00:00
- **Records Fetched From All Connectors:**
- **Records Inserted From All Connectors:**
- **Records Updated From All Connectors:**
- **Records Removed From All Connectors:**

*Figure B-3 Activity Log document*

Another way to retrieve the log document displayed in Figure B-3 is by opening the LEI Log database (leilog.nsf) directly. Use the “Log” link in the bottom right corner of the LEI Administrator screen. All log documents are available from the LEI log views (Figure B-4). This lets you refer back to messages from an earlier run.

*Figure B-4 LEI log database*

Scripted activity log entries are a little different from those for other activities. Because a scripted activity creates an LC “session” and chooses its own name for the session, which may differ from the name of the activity, there will be two log documents for each execution of the scripted activity — one with the name of the activity document, and one with the name of the session created by the agent. If you open the activity document log, however, it will display the agent’s log entries also (starting in Version 6.0.2).
Using the Domino log (log.nsf)

Sometimes, you can find additional information about the activity by examining the Domino server log. This is particularly true for Scripted Activities; if they use Print or MessageBox, or if they terminate with an error which your code doesn't trap and deal with, that output will end up in the Domino log, not the LEI log.

To find the log messages, open the Domino log database, log.nsf, on the server that runs the LEI process. Open the “Miscellaneous Events” view as shown in Figure B-5, and look for the log document whose time period includes the time you ran the activity.

You can double-click on the document highlighted in the view to examine the contents. Figure B-6 displays the contents of Domino log document dated 5/29/2003 at 7:07 AM until the current time. Here you can see that an error has occurred and you also have more detailed information about the error. The “Agent message” line represents output created with a LotusScript MessageBox or Print statement. The lines in boldface represent an unhandled error condition which made the agent terminate.

05/29/2003 10:07:55 Opened session for leidomsrv/ITSO (Release 6.0.1)
05/29/2003 10:07:56 LEI: Activity started
05/29/2003 10:07:57 Agent message: Before executing sqlStmt CREATE TABLE
EMPLIB.CASE_FILES ( CASENUM SMALLINT NOT NULL , DATEOPENED DATE DEFAULT NULL , TITLE CHAR(50) CCSID 37 DEFAULT NULL , DESCRIPTION FOR COLUMN DESCR00001 CHAR(100) CCSID 37 DEFAULT NULL , STATUS CHAR(15) CCSID 37 DEFAULT NULL , OPENEDBY CHAR(20) CCSID 37 DEFAULT NULL , ASSIGNEDTO CHAR(20) CCSID 37 DEFAULT NULL , CLIENT CHAR(20) CCSID 37 DEFAULT NULL , CONTACT CHAR(20) CCSID 37 DEFAULT NULL , CONTACTPHONE FOR COLUMN CONTAC00001 CHAR(12)
05/29/2003 10:07:57 Agent 'Case Files Direct Transfer' error: Error: CASE_FILES in EMPLIB type *FILE already exists., Connector 'DB2 Connection', Method -Execute- (-601)
05/29/2003 10:07:57 LEI: Activity finished: CaseFiles_DirectTransfer
05/29/2003 10:07:57 LEI: Activity finished
05/29/2003 10:08:04 Closed session for leidomsrv/ITSO|Databases accessed: 2
Documents read: 0 Documents written: 0
05/29/2003 10:08:04 LEI: Activity finished: Direct Transfer Scripted Activity for Case Files
05/29/2003 10:08:04 LEI: Activity finished

Figure B-6  Domino log document
Generally speaking, it's better to trap errors using the On Error statement and handle them so that you don't get messages in the Domino server log. Use the LCSession.LogText method to add the error information to the agent log entry in the LEI log database, so that there's only one place you need to look to see what happened in your agent. In addition, it's a good idea to make your message include the module name and line number where an error occurred, so that you can go directly to the right place in the code. Sample code and discussion of best practices for writing easy-to-debug code can be found at:

http://www-10.lotus.com/ldd/today.nsf/0/a2b75e1747c115c285256d6e00602957?OpenDocument

**Enabling log conflicts in replication activities**

In the replication activity document you have the option to enable conflict logging. This option is available through the Replication Options tab, Conflict Options sub-tab (Figure B-7). By enabling this option, you can use the entries created in the activity log to troubleshoot the problem. Logging entries will also appear in your Domino server console.

![Figure B-7](image)

The Logging Options specify how conflict logging will be performed.

- **As Event** — logs conflicts as errors in the activity log.
- **As Error and Continue** — logs conflicts as errors in the activity log, but continues the replication activity.
- **As Error and Stop** — logs conflicts as errors in the activity log and stops the replication activity

**Note:** Using the As Error options is slower, since each log entry is written to disk. Using the As Event option is much faster and is recommended for production environments.

The Limit to field is used to limit logged conflicts to a useful number for the given activity. When this number of conflicts has been logged, additional conflicts are not added to the activity log. The default is 500. Specifying a value of zero indicates no limit (log all conflicts).

Figure B-8 shows an example of the output in the activity log.
Implementing IBM Lotus Enterprise Integrator 6

Logging SQL statements

In relational database connection documents, you have the option to output SQL statements to the activity log. This option is available through the Logging Options tab under the Connectivity section (Figure B-9). This option is useful in troubleshooting if you want to inspect the generated SQL statements.

![Figure B-8 Conflict logging output](image)

This causes the SQL statements to be printed out in the activity log document when the activity that uses that connection document is started. Figure B-10 shows an example activity log of a timestamp replication activity.
Using the trace metaconnector

The trace metaconnector lets you trace events associated with a specified connection. You may specify options including where to capture data (either to the LEI console, LEI log, or to a specified file name) and whether or not to include a timestamp with each trace log entry.

If there is a problem with your activity or connection, you can “trace” it to find out exactly what it’s doing. Lotus support staff may need this information to assist you.

For more information about using the trace metaconnector, refer to 6.6.4, “Trace MetaConnector” on page 103.

Using Notes Server Diagnostic (NSD)

The NSD is a Lotus tool that gathers information from a Domino server when it crashes. If you contact Lotus Support, they will want a copy of this file.

Interpreting the NSD file is something of an art — you might not be able to make much sense out of it yourself, and you probably won’t be able to tell which activity caused the crash. But if you already know which activity failed, then from looking at the call stack, you might be able to tell what it was trying to do at the time — construct a fieldlist, select records, fetch....

For more details about NSDs on AS/400, refer to Section 3.2.4 of the Lotus Domino for AS/400: Problem Determination Guide, SG24-6051. Other operating systems will have information that is formatted differently, but there are some common features, so this redbook may be a useful reference for any server type. You can download it from:

http://www.redbooks.ibm.com/
Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246067

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select Additional materials and open the directory that corresponds with the redbook form number, SG246067.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEISampleData.nsf</td>
<td>Notes database containing example data, agents and forms</td>
</tr>
</tbody>
</table>

How to use the Web material

Download the files and place them in the data directory on your Domino server, in a subfolder named “demo”. For example, on a Windows 32-bit server you might put them in C:\Lotus\Domino\Data\demo. The files don’t have to be in a subfolder to work, but this will let you follow the examples exactly.

The databases come with Manager access for -Default-. You should adjust this access so that you and the ID you use to run LEI, have Manager access. If you want to test Notes users other than yourself, they will need at least Editor access.
The file LEISampleData.nsf contains data that you migrate to a relational database of your choice, using LEI batch activities. Once the data is available in a relational database, you can use the database to access the information from the Notes client.

Further useful downloads

Go to the Lotus LEI Web site at:

http://www.lotus.com/lei

Click “Downloads” to find many other useful files, including trial versions of the LEI software.

The install program for LEI also offers you the option of installing a LC LSX sample database, which can be helpful.
Glossary

**Activity.** A distinct task performed by LEI, represented and defined by an activity document in the LEI Administrator Notes database.

**Base Connection.** When used in reference to a metaconnection, refers to the connection document to the actual data source, which the metaconnectors mediate. See 6.6, “Metaconnectors” on page 97.

**Code page.** A system of representing characters in digital form — different languages use different code pages that contain representations of the characters used in that language. The most commonly used code pages in English are ASCII, UTF-8 and Unicode. Lotus Notes employs LMBCS.

**Connection Document.** A document in the LEI Administrator that contains the necessary parameters (server name, database name, userid, password, ...) for establishing a connection. Refer to Chapter 6, “Connection documents” on page 73.

**Connection Pooling.** With reference to the LC LSX, connection pooling lets you keep a connection active between invocations of an agent so that you can avoid the overhead of establishing the connection each time the agent is called.

**Connection.** An active “login” to a relational database or other datasource, via a connector.

**Connector.** See Lotus Connector.

**Document.** In Notes, a container for data, roughly corresponding to a row in a relational table. The document is the actual data. The Notes form is a description of the data, similar to a table “metadata” in a relational table. It is not correct to use the words form and document interchangeably.

**Domino.** The server part of the Notes/Domino client/server system. This term is also commonly used in referring to a Web application implemented in Domino Designer.

**Drop.** In a relational database, to remove a row, table, index or other object from the database.

**Fieldlist.** Term used in the LC LSX to designate a collection of fields, which may represent the columns of a table, the parameters to a stored procedure, or other group of field information.

**Form.** In Notes, a combination of UI design and application logic, used to specify what fields will be stored in a particular kind of document, their datatypes, default values, and validation rules. Contrast with document.

**Full-text index.** Notes databases can optionally be indexed for rapid searching. This lets you quickly find all documents that contain a particular term in any field, including rich text fields and (optionally) file attachments. One may also construct queries to find search terms in specific fields or using date ranges on date/time fields.

**Index.** In a relational database, an index is a pre-calculated sort of one or more columns, allowing those columns to be efficiently searched. See also view index and full-text index.

**Insert.** Refers to the operation of inserting new records into a relational table. The same terminology is used when referring to record creation via non-relational connectors, for example, the creation of a Notes document, if done via the Notes connector.

**Key Document.** See “Stub Document”.

**Key field.** With reference to an RDB, a key field is a field we are using at the moment to locate records via a Select. Key fields are ordinarily, but not necessarily, indexed.

**LC LSX.** A LotusScript interface to the Lotus Connectors, letting you write agents and other LotusScript code that efficiently access and update relational data. See Chapter 9, “Getting Started with the LC LSX” on page 191.

**LMBCS.** Lotus Multi-Byte Character set. A code page used internally by Lotus Notes, supporting essentially the same character set as Unicode. The characters in this representation use variable numbers of bytes, so that characters in the ASCII set can be represented with a single byte.

**Lotus Connector.** An API — a set of functions — allowing access to a specific source of data via a common set of functions. There is a DB2 connector, an Oracle 7 connector, ODBC connector, and so on, plus some non-relational sources such as Lotus Notes, text file, and file system.

**LSXLC.** See LC LSX. LSXLC is the actual name of the LSX library file.

**LSXLEI.** Now obsolete LEI-specific version of the LC LSX. The functionality of LSXLEI and LC LSX have been merged in LEI 6.0.
Metaconnector. A special type of connector that serves as a filter for the data or commands sent to another connector. For instance, the order metaconnector lets you sort data you read from any other connector type.

Metadata. A property of a connection — the generic term Metadata refers to a RDB table or view, or whatever data object in the source system resembles a table. For instance, the Metadata in a Notes connection is the name of a Notes form.

Multivalue field. In Notes, fields can be designated as “multivalue,” meaning they may contain a list of values instead of a single value. RDB fields typically do not support multiple values — instead they would use a separate key table in a many-to-one relationship with the main record table. LEI contains tools for converting from one to the other representation.

Notes. The dedicated client of the Notes/Domino client/server system.

Persistent Connection. A connection that LEI doesn’t “log out” from, but instead keeps active even when it’s not in use, so that when it’s needed it’s available instantly. Refer to 2.3.4, “Persistent connections” on page 35.

RDB. Relational Database, such as DB2 or Sybase.

Realtime Notes. See “Virtual Fields”

Replication. When used in reference to Notes databases, refers to the ability for changes in one database to transfer to and from a “replica” on a different Domino server or on a user’s local hard disk. LEI also supports a Replication activity that lets you do the same sort of bidirectional or unidirectional updates between different relational database types or between a RDB and a Lotus Notes database.

Rich Text. “Word Processor” style text containing formatting, font changes, tables, pictures, file attachments and so on, stored in a “Rich Text Field” in a Notes document. When copied to relational database systems, this information can be stored in a binary or “BLOB” field in the RDB, or in a text field. There’s no way to use the binary data without a Notes client, but if you store just text, you lose the formatting.

Stub Document. A Notes document used to store key information for a Virtual Fields activity.

Table. The basic method of organizing information in a relational database.

Truncate. In a relational database, to remove all the data in a table.

Update. Making a modification to an existing RDB record, or a similar operation via a non-relational connector, for example, modifying a Notes document via the Notes connector.

View Index. In a Notes database, a cached “image” of a Notes view. For performance in displaying view contents, Notes stores copies of the information in each of the view columns. When a view is used, the column values are recalculated only for any documents that are new or changed since the last use. A view index is similar to a RDB index in that it supports rapid searching of pre-sorted columns.

Virtual Documents. An activity type in which all the field data is stored in relational databases and presented using the Notes UI, so there is no “stub” Notes document.

Virtual Fields. (Formerly known as “Realtime Notes”) An activity type in which the Notes database contains a set of “stub” documents with key values corresponding to keys in a relational table. When a Notes document is opened, the activity uses the key value(s) to look up the values of the other fields. When the document is saved, the key is used to perform a keyed update to the relational back end. Likewise, creation and deletion of Notes documents may be captured to perform corresponding changes in the relational data.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 256. Note that some of the documents referenced here may be available in softcopy only.

- Implementing IBM Lotus Enterprise Integrator 6 on the IBM @server iSeries Server, SG24-6941
- Performance Considerations for Domino Applications, SG24-5602
- Lotus Notes and Domino R5.0 Security Infrastructure Revealed, SG24-5341

Other publications

These publications are also relevant as further information sources:

- Indexing Strategies for DB2 UDB for iSeries white paper, which is found at:
- Data Sorting Considerations with Lotus Enterprise Integrator (LEI) 6 on the IBM @server iSeries Server, which is found at:
- Inside Notes: The Architecture of Notes and the Domino Server, which is found at:
  http://www-10.lotus.com/ldd/notesua.nsf/0b345eb9d127270b8525665d006bc355/ec73cbf1c6392ba385256856005bd224
- Optimizing LEI 6 performance with Virtual Documents and DB2 (LDD Today), which is found at:
  http://www.lotus.com/ldd/today.nsf/a2535b4ba6b4d13f85256c59006bd67d/62c6432de2aa12785256960554c5a?OpenDocument
- “LEI: Addressing Sort Order Issues when Replicating DB2/400 Data to a Notes Database”, Technote 110127, which is found at:

All of the following publications can be downloaded from the IBM Lotus Developers Domain Documentation Library at:

http://www.lotus.com/ldd/doc

- IBM Lotus Enterprise Integrator for Domino (LEI) Release Notes (various versions)
- IBM Lotus Enterprise Integrator for Domino (LEI) Installation Guide
- IBM Lotus Enterprise Integrator for Domino (LEI) Activities and User Guide
- Lotus Connectors and Connectivity Guide
Online resources

These Web sites and URLs are also relevant as further information sources:

- Lotus Enterprise Integration
  http://www.lotus.com/ei
  http://www.lotus.com/lei
- Lotus Developer Domain LDD Today
  http://www.lotus.com/ldd/today.nsf
- iSeries Information Center
  http://www.ibm.com/eserver/iseries/infocenter
- iSeries performance resources
- DRDA and DB2 Information Integrator
  http://www.ibm.com/eserver/iseries/db2
- Lotus Developer Domain’s Sandbox
  http://www-10.lotus.com/ldd/sandbox.nsf
- LDD Today: sample code and discussion of best practices
  http://www.lotus.com/ldd/today.nsf/0/a2b75e1747c115c285256d6e00602957?OpenDocument

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks
Index

A
access control list
  for Credentials database 29
  for LEI Administrator 27
Accessing data in Notes databases 207
ACL
  See access control list
activities
  chaining together 109, 124
  Direct Transfer 111
  running on a schedule 109
  setting execution priority 109
  troubleshooting 71
Administration
  Administrator database 47
  Administrator Configuration document 52
Advanced RealTime activities 151
  differences between virtual documents and virtual
  fields 151
  introduction 9
Advanced Realtime activities 105
Agent to Execute
  in Notes connection 84
Application architecture
  application logic tier 3
  data tier 3
  presentation tier 3
Architectural scenario
  Decentralized Advanced RealTime access 25
  LEI 6 Advanced Realtime plus Notes replication 24
  LEI access to multiple relational back-ends 24
  Remote database servers 23
  Single Domino/LEI server 22
attachments
  in regular Notes documents 157
  virtual 156

B
Batch activities 105
  editing configuration to run test activities 105
  Example - Setting up a Command Activity 106
  example using Direct Transfer activity 112
  running via Notes connection document 79
Batch LEI activities
  introduction 8
  binary data 206

C
clustering 53
command activity
  example as a batch activity 106
  Notes connection 107
  Configuration 50
Connection
  Persistent 35
connection document 73
  example using DB2 connection document 74
  how to use in LC LSX 192
  overview 6
  supported document types 73
connection pooling 208
  code example 224
connectivity
  testing connectivity to relational databases 67
Connectivity section 80
CONTEST 70
Create table
  using LC LSX, example 224
Creating an external key table
  considerations 175
Creating an index
  using LC LSX, example 224

D
Data Connection Resource
  See DCR
Data Journalling 75
data migration 58
data migration from earlier versions 57
data tier 3
Data type
  correspondences between systems 18
DB2
  data journaling 75
DB2 UDB
  on iSeries 67
DCR 5
debugging
  logging SQL 77
DECS 4
  capabilities in relation to the features available in LEI 5
DECS, DCRs, and LEI
  comparison between using 10
decadm.nsf 6
DECSDatabase custom class 192
deleting records
  in LC LSX 224
deletion stub 177
dependent activity 109–110, 124
designated server
  of batch activity 109
Direct Transfer 111, 120
Direct Transfer activity 112
  creating a secondary Direct Transfer activity 122
  establishing a chain of activities 124
E
encryption
  of database passwords  75
error messages
  Field count mismatch  168
  unable to retrieve external record  176
Execute method
  apostrophes in data  213

F
filter formula  162

I
Installation
  Pre-installation activities  37
Installation of LEI  37, 41
integrated credentials  12, 27, 185
  credentials database  28
  performance  32, 34–35
  using in Advanced RealTime activity documents  31
Invalid index error message  126

J
Java  9
JDBC  230

L
LC LSX  6, 191
  Notes Client Coding Considerations  229
  Performance Tips  210
LC LSX, installing on Notes client  229
LCCurrency Class  196
LCDateTime Class  196
LCField Class  194
LCFIELDDF_KEY  198
LCFieldlist Class  194
LCNumeric Class  196
LCSession Class  193
LCStream Class  195
LEI
  business case for  2
  configuration  50
  removing  233
LEI Administrator database  47
LEI cluster  53
LEI clustering  53
  setting up during install  42
LEI Installation Guide  57
LEI log  244
LEI LSX  9, 58, 192
LEI performance  32
LEI server
  starting and stopping the server  55
LEI.INI
  retirement of LEI.ini file  13
leidm.nsf  14
  See aso decsadm.nsf
leilog.nsf  244
leivlt.nsf
  See Script vault
leivlt6.nsf  149
log
  SQL statements  77
log entry
  for activity, finding  49
log messages
  using LC LSX, example  224
Lotus Connectors
  LC LSX programming interface to  191
LotusScript
  xi, 191, 264
LS
  DO  229
LSX Classes  192
LSXLEI
  See LEI LSX, LC LSX
LSXODBC  229

M
MapByName
  no effect on Select keys  200
Metaconnection documents
  migrating from LEI 3.x  59
metaconnectors  97
  introduction  7
  stacking  98
metadata
  activity creates automatically  119
  creating automatically  119
migrating
  pre-migration activities  57
migrating LEI LSX code  192
migrating to LEI 6  57
migration
  troubleshooting  65
migration considerations  58
migration.log  66
monitor order
  for virtual fields activity  152
  in Advanced Realtime activities  162
multi-row  33, 119
multivalue fields
  in virtual documents  173

N
NCONTEST
  See CONTEST
New features
  what's new with LEI 6.0 through 6.0.2  12
newlines
  in relational data  162
Note ID of a Virtual Document
  Virtual Document note ID  189
Notes connection document
  connectivity section  80
Notes connector  207
NotesPump 6

O
ODBCConnection class 229
Order metaconnection properties 220
Order metaconnections 135
OS Priority of batch activity 109

P
passwords encryption 75
Performance considerations 32
Persistent connections 32, 35
Polling activity 149
presentation tier 3
Primary key replication 130
example RDB to Domino 132

R
Redbooks Web site 256
Contact us xiv
Remote administration
See LEI clustering replication
troubleshooting 141
revirtualization
virtual documents activity 176

S
scenario
Decentralized Advanced RealTime access 25
LEI 6 Advanced Realtime plus Notes replication 24
LEI access to multiple relational back-ends 24
Remote database servers 23
Single Domino/LEI server 22
Script
See LotusScript, Scripted Activity, LC LSX
Script vault 58
Scripted 125
Scripted Activity 125
example 126
Scripted Activity agents
special considerations 126
Secondary Replication activity 144
security
integrated credentials 27
of database passwords 27
SQLQuote function 213
starting and stopping the server 55

T
Terminology
comparing across RDB, LEI and Lotus Domino 16
Timestamp replication 131
performance 33
Troubleshooting 243
activity 71
migration from LEI 3.x 65
Troubleshooting replication activity 141

U
User assistance wizard 48
user assistant 8
Using the Domino log (log.nsf) 246

V
virtual agents activity 181
virtual attachments 156
reattaching when key docs deleted 158
recovery 164
virtual documents
deletion 177
Domino replication 178
External keys 174
key fields, editing in Notes 158
multivalue fields 173
Note IDs 189
virtual documents activity
limitations and considerations 181
virtual fields 152
adding key documents 154
filter formula 162
keeping stubs up to date 131
monitor order 152
updating retained fields 156
virtual fields activity 152
event options 154
example 158
setting up 153
Implementing IBM Lotus Enterprise Integrator 6

Architecting and designing enterprise solutions using LEI 6

Installing and administering LEI 6

Getting the most out of the Advanced RealTime activities

This IBM Redbook helps you to implement Lotus Enterprise Integrator (LEI) 6. It is targeted for system administrators who plan to implement or upgrade to LEI 6 in their organization. This book provides tips and techniques to help you successfully deploy and administer LEI 6.

We begin by providing a brief introduction to what’s new with LEI 6 and cover some architectural scenarios of how LEI 6 infrastructure could be implemented in a customer’s environment. We then provide detailed information on how to install and administer LEI 6, as well as how to migrate or upgrade from LEI 3.x to LEI 6. Tools, techniques and utilities for debugging activities are described in depth, for testing connectivity to relational databases and testing connection documents in the LEI Administrator. Working examples are provided to illustrate how to use the Advanced RealTime activities that are new in LEI 6. It also provides examples of some of the batch activities previously available in LEI 3.x such as a scripted activity and a replication activity. Finally, the book concludes with a discussion of the LC LSX, the LotusScript programming interface to the Lotus Connectors which lets you program any functionality you need that isn’t available through the LEI forms.