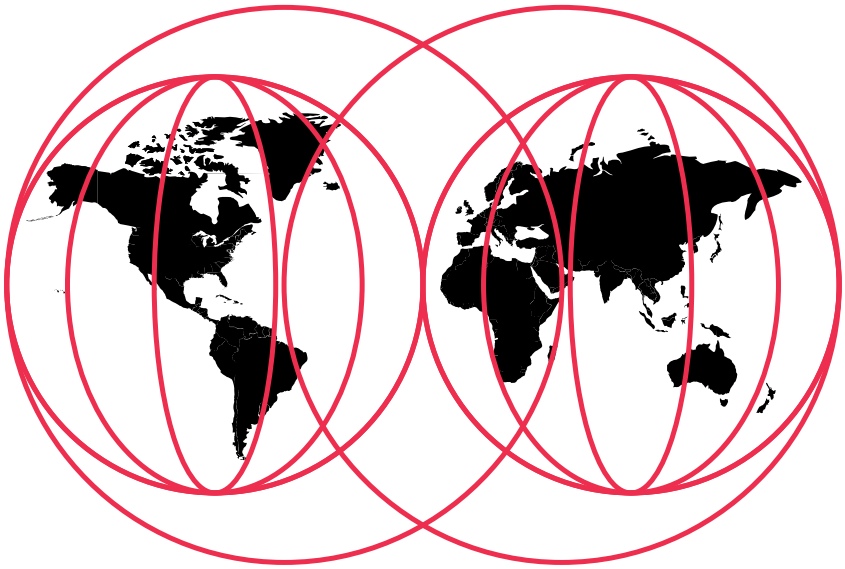
Lotus

IBM

Lotus Domino R5.0 Enterprise Integration: Architecture and Products

Søren Peter Nielsen, Theo Barkhuizen, Laurisa G. Maldonado, Robert Perron, Peter Sing



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5593-00

SG24-5593-00

International Technical Support Organization

Lotus Domino R5.0 Enterprise Integration: Architecture and Products

July 1999

Take Note!
Before using this information and the product it supports, be sure to read the general information in the Special Notices section at the back of this book.

First Edition (July 1999)

This edition applies to Lotus Domino Release 5.0 and Lotus Enterprise Integrator Release 3.0.

Comments may be addressed to: IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Preface

Lotus® provides services for connecting Domino™ to a variety of enterprise systems, such as relational database management, transaction processing, resource planning applications and unstructured data. This redbook describes Domino's open architecture for enterprise integration and the associated products. It presents detailed discussion about using Domino Enterprise Connection Services (DECS), which provide a real-time forms-based interface to enterprise data, and Lotus Enterprise Integrator (LEI) the successor to NotesPump™, which provides scheduled and event-driven high-speed data transfer capabilities between Domino and enterprise systems. Also discussed are the programming interfaces used in enterprise integration, including the Lotus Connector LotusScript® Extension, the Lotus Connector Java classes and the Lotus Connector Toolkit, as well as the product-specific LotusScript Extensions (SAP R/3, MQSeries™, DB2® and ODBC). Additional integration methods are described in detail, including Java Database Connectivity, CORBA, ActiveX Data Object, NotesSQL™, and Servlets. Finally, this redbook includes Enterprise Integration decision charts, and leads you through the steps necessary to select the best integration tools for your particular requirements. This book is useful for anyone interested in planning for, implementing, or administering Domino integration into an enterprise system.

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 Center at Lotus in Cambridge, Massachusetts, USA.

Søren Peter Nielsen works for the International Technical Support Organization at Lotus Development, Cambridge, Massachusetts. He manages projects whose objective it is to produce redbooks on all areas of Lotus products. Before joining the ITSO in 1998, Søren worked as an IT Architect for IBM® Global Services in Denmark on solutions for a wide range of industries. Søren is a Certified Lotus Professional at the Principal level in Application Development and System Administration.

Theo Barkhuizen is a Senior I/T Specialist with IBM Global Services in Johannesburg, South Africa. Theo started to work with Lotus Notes® in 1997 and is a Principal Certified Lotus Professional. He is involved with the development of systems that support the processes of a outsourced Help Desk environment.

Laurisa G. Maldonado works with the IBM Lotus Integration Center (ILIC) in Dallas, Texas where her focus is Domino on AIX®. She holds a degree in Computer Science from the University of Texas at El Paso and has been with IBM for four years. During the majority of that time, she has focused her attention on Domino and its related connector products. She has contributed to the development and instruction of Domino course material in two separate IBM/Lotus workshops. Laurisa can be reached at laurisa@us.ibm.com.

Robert Perron is a User Assistance Architect with Lotus in Cambridge, MA. He has developed documentation for Lotus Notes and Domino for over five years with a primary concentration on programmability. He developed the documentation for the LotusScript and Java Notes classes, including the LS:DO classes, and coauthored the book 60 Minute Guide to LotusScript 3 - Programming for Notes 4.

Peter Sing is a Senior I/T Specialist with IBM Canada's e-business Services area. Peter is the Database Connectivity Competency Leader for Lotus Notes within e-business and specializes as a technical design and conceptual design architect/developer for Notes Enterprise Integration products. Peter's MVS roots give him the added value of dealing in depth with EI products that allow MVS integration. Peter has worked with Lotus Notes since 1995 and quickly focussed his skills on MQSeries, MQLSX, and NotesPump (known now as a combination of DECS and LEI).

A number of people have provided support and guidance. In particular, we would like to thank **Martha Hoyt, Mary Peterson** and **Lauren Wendel**, Product Managers for Domino Enterprise Integration as well as **Carl Hero**, Lotus Business Partner Technical Enablement, for making his enterprise integration lab available to the redbook team. In addition, we would like to thank the following people from Lotus and IBM:

- Mark Field
- John E. Martin
- Evelyn McKay
- Peter Orbeton
- Paul Orsillo
- Glen Salmon
- Alison Chandler, ITSO Poughkeepsie
- Graphic Services, Lotus North Reading

Comments Welcome

Your comments are important to us!

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

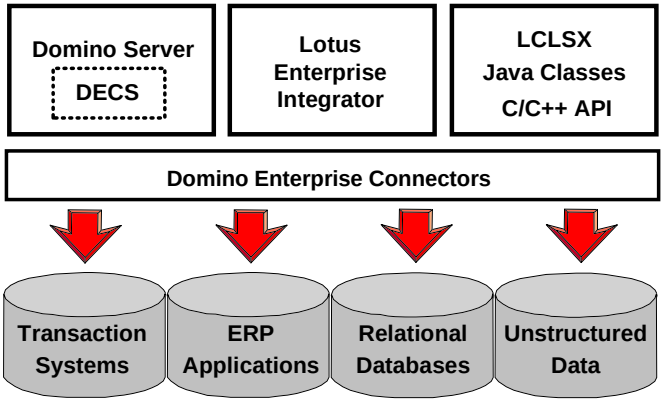
- Fax the evaluation form found at the back of this book to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an Internet note to redbook@us.ibm.com.

Chapter 1
Overview

Lotus provides services for connecting Domino R5.0 to relational database management systems (RDBMS), transaction processing (TP) systems, enterprise resource planning (ERP) applications, and unstructured data. These services present the user with a common interface to the many enterprise systems. The basic services are as follows:

- Domino Enterprise Connection Services (DECS): provides a real-time forms-based interface to enterprise data. DECS is included with the Domino R5.0 server.
- Lotus Enterprise Integrator (LEI): provides scheduled and event-driven high-speed data transfer capabilities between Domino and enterprise systems. LEI is a separate product.
- Lotus Connector LotusScript Extension (LC LSX): provides LotusScript access to enterprise systems. The LC LSX is available on the "Lotus Enterprise Integration" Web site, and is bundled with Domino R5.0 and LEI 3.0.
- Lotus Connector Java classes: provides Java access to enterprise systems. The LC Java classes are available on the "Lotus Enterprise Integration" Web site, and are bundled with LEI 3.0.
- Lotus Connector API: provides C/C++ access to enterprise systems. The LC API is available on the "Lotus Enterprise Integration" Web site.

Underlying these services are individual connectors for the supported enterprise systems. The following figure shows Lotus Enterprise Integration Architecture and you can see where the different services fit in the architecture.



For current users, LEI is an upgrade of NotesPump and connectors are analogous to links. DECS is functionally similar to the RealTime Notes™ activity in LEI and NotesPump. DECS and LEI are supported by Domino R4.63 and above as well as R5.0.

For LotusScript and Java programmers, the LC LSX and LC Java classes provide a common interface to all enterprise systems. You can still use the ODBC (LS:DO), DB2, MQSeries, and SAP R/3 LSXs available for Domino R4.x and compatible with R5.0.

Lotus Connectors

Lotus connectors exist to permit access to external data sources from Domino for relational database management systems, enterprise resource planning systems, transaction processing systems, directory services, and other services. The connectors are briefly described in the remainder of this section; . see the appendix “Lotus Connectors” for details.

Relational Database Management Systems

The list of available relational database management systems (RDBMS) connectors follows. These connectors are bundled with DECS and LEI.

- DB2: transfers data to and from DB2 Universal Database sources using DB2 Connect Personal Edition Version 5, DB2 Connect Enterprise Edition Version 5, or Client Application Enabler (CAE) Version 2.1.2 or later. You can also connect to DB2 on an AS/400™ or mainframe using a DDCS gateway.
- Oracle: transfers data to and from Oracle sources.
- Sybase: transfers data to and from Sybase sources.

- ODBC: transfers data to and from ODBC-compliant sources, including MS-SQLServer. ODBC drivers must be appropriate to the operating system. They must be thread safe. The ODBC Administrator must be present and must define the data sources under "System DSN."

Enterprise Resource Planning Applications

The list of available enterprise resource planning (ERP) connectors follows. These connectors are all separate products.

- J.D. Edwards OneWorld: integrates Domino with J.D. Edwards OneWorld ERP systems. The connector performs data transfers through the application layer to preserve all business logic validations. The Domino or LEI machine must contain J.D. Edwards OneWorld Client version B732 or later.
- Oracle Financial Applications: integrates Oracle Applications Business Modules data into Domino environments.
- PeopleSoft: enables Domino applications to exchange data with PeopleSoft Application data. You need the PeopleSoft Message Agent and ODBC drivers appropriate for the Domino or LEI machine. The drivers must be 32-bit for Windows NT and OS/2®. The ODBC Administrator must be present and data sources for PeopleSoft must be defined.
- SAP R/3: integrates Domino and SAP applications. Reading and writing data is always through the application layer, maintaining all RFC and R/3 Transaction business rules.
- Lawson Enterprise/400: integrates Domino with Lawson Enterprise/400 systems.

Transaction Processing Systems

The list of available transaction processing (TP) system connectors follows. These connectors are all separate products.

- CICS: provides Domino applications access to CICS DPL programs and CICS 3270 transactions running on any CICS server that can communicate with one of the supported CICS clients.
- IMS: connects Domino applications with IMS applications.
- MQSeries: connects Domino applications with most MQSeries-enabled applications, and also provides explicit support for the MQSeries MVS/ESA bridges MQSeries-IMS and MQSeries-CICS. The MQSeries connector uses the MQI to invoke MQSeries services, and can be used with either the MQSeries queue manager or client.

- Transarc Encina TXSeries: connects Domino applications with Transarc Encina applications.
- BEA Tuxedo: connects Domino applications with Tuxedo-managed domains and data sources. You need Tuxedo version 6.x or later for the operating system containing Domino or LEI. Both native (local to the Tuxedo domain) and workstation client (remote connections) versions of Tuxedo are supported; choose one or the other.

Directory Services

The list of available directory services connectors follows. These connectors are bundled with LEI.

- Domino Directory : transfers data to and from Domino Directories — formerly known as Public Name & Address Books (NAB).
- Lightweight Directory Access Protocol (LDAP): transfers data to and from an LDAP Directory Service.
- Novell Directory Services (NDS): transfers data to and from Novell directories.

Other Services

The list of additional connectors follows. These connectors are bundled with DECS and LEI.

- File System: allows you to connect to a file system accessible to the Domino or LEI computer.
- EDA/SQL: transfers data to and from sources supported by EDA Servers from Information Builders (http://www.ibi.com). You need the EDA/Client software for the operating system containing DECS or LEI. You must have an EDA Server on the platform containing the EDA database. The EDA software must be release 3.2 or later and must be 32-bit for NT and OS/2.
- Notes™: transfers data to and from Domino (Notes) databases.
- Text: transfers data to and from text files. You define formats for the source and destination data in a ZMerge Information Description (ZID) script.

Platforms

The platforms supported are Windows 95, NT 4.0, Digital NT Alpha, AIX 4.14 and later, OS/2 Warp™, HP-UX 11.0 and later, Sun Solaris 2.51 and later, S/390, and AS/400. Not all systems work on all platforms. See the appendix “Lotus Connectors” for details. Lotus connectors can access enterprise systems on their own platforms.

MetaConnectors

A MetaConnector is a special connector that processes data prior to transfer. The following MetaConnectors are provided:

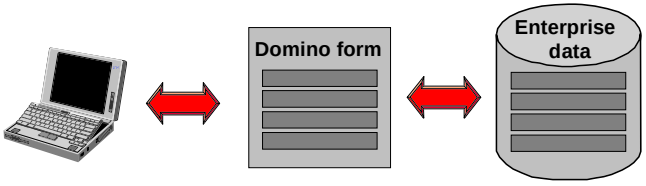
- Collapse/Expand: collapses multiple records from a source table into a single field, and expands a form field into multiple records.
- Connection Broker: authenticates individual username and password parameters for external connector sources, and also directs data to and from different connections based on contact information supplied with each row of data.
- Metering: collects statistical usage data.
- Order: orders data sets from different server sources.

Character Sets

The enterprise services support over 200 character set mappings, including US English, Asian, Far East, Western European, and Eastern European languages, as well as multi-lingual character sets such as LMBCS and Unicode. Support is provided for translating characters from one set to another. See the appendix "Character Sets" for a list of the supported character sets.

Domino Enterprise Connection Services

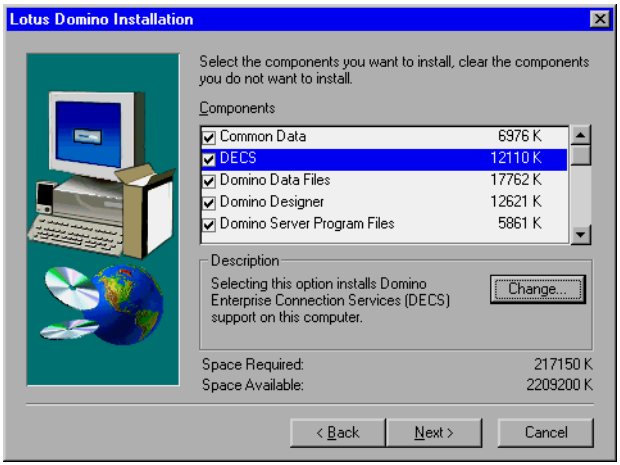
Domino Enterprise Connection Services (DECS) extends the Domino application to query and modify enterprise data. When a Notes or Internet client accesses data through a Domino form connected to an enterprise system, DECS transparently retrieves and updates the enterprise data. The client needs no additional software and needs no knowledge of the enterprise system. DECS requires no programming.



This section briefly introduces DECS. See the chapter "Domino Enterprise Connection Services" for details.

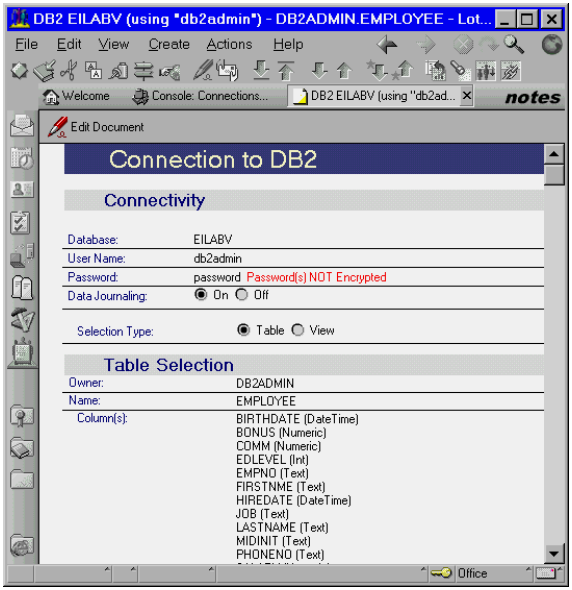
Administration

The Domino administrator includes DECS as part of the Domino install process. DECS runs as an add-in task with an administration interface through the "DECS Administrator" (decsadm.nsf) Domino application.



Connections

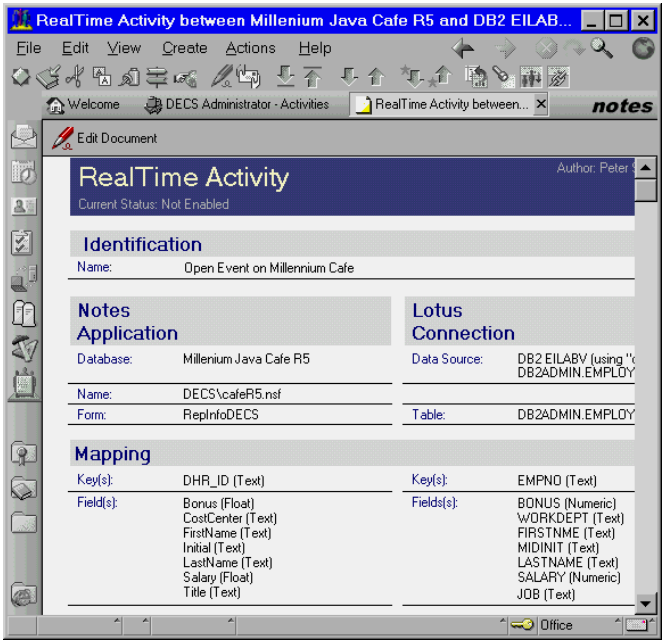
A connection specifies the information necessary to connect Domino and a particular enterprise system. A wizard will assist you in properly completing a connection document. Connection documents may be used by several DECS real-time activities.



Activities

A DECS real-time activity defines the way you wish your Domino application to interact with your enterprise data. You administer DECS real-time activities through Activity documents. Each document describes one activity by:

- Associating a connection with a form in a Domino database
- Mapping keys and data fields between the Domino form and the enterprise source
- Monitoring an event such as Create, Open, Update, or Delete



When a Notes or Internet client accesses data through a form using a monitored event, DECS intercepts the Domino form processing. DECS processes the associated data on the enterprise source and returns any results to Domino. Domino then interacts with the client in the usual manner.

Lotus Enterprise Integrator

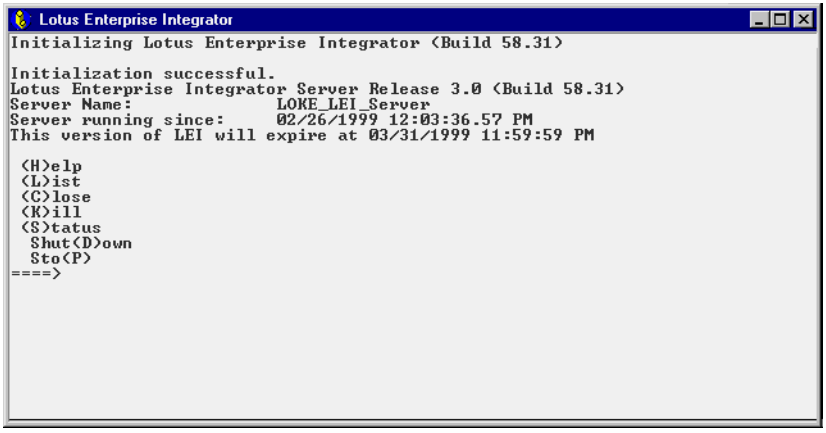
Lotus Enterprise Integrator (LEI) performs data transfer, data synchronization, and other activities between data sources. A data source can be Domino or any supported enterprise system. Data activities occur on a scheduled or event-driven basis and are capable of high-volume, high-speed transfers.

LEI 3.0 can be administered through a Domino R4.6 server or later, and can move data from any Domino or Notes server. LEI 3.0 is an upgrade of NotesPump 2.5a.

This section briefly introduces LEI. See the chapter “Lotus Enterprise Integrator” for details.

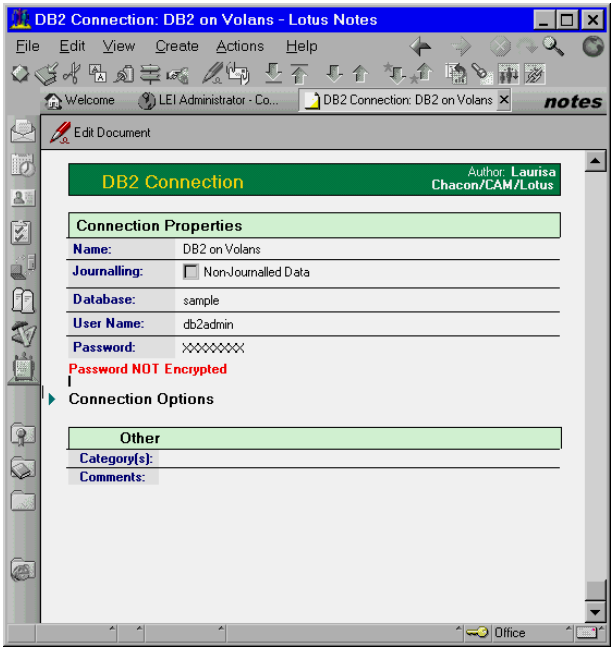
Administration

If the RealTime Notes™ Activity is not used, you can install LEI as an independent server on any computer that contains Domino or Notes software (Domino or Notes does not have to be running). If the RealTime Notes Activity is used, you must install LEI on the Domino server containing the applications that interact with the enterprise data. You administer LEI from the LEI Administrator (leiadm.nsf) Domino application and (for limited functions) the LEI server command window, unless LEI is a Domino add-in task.



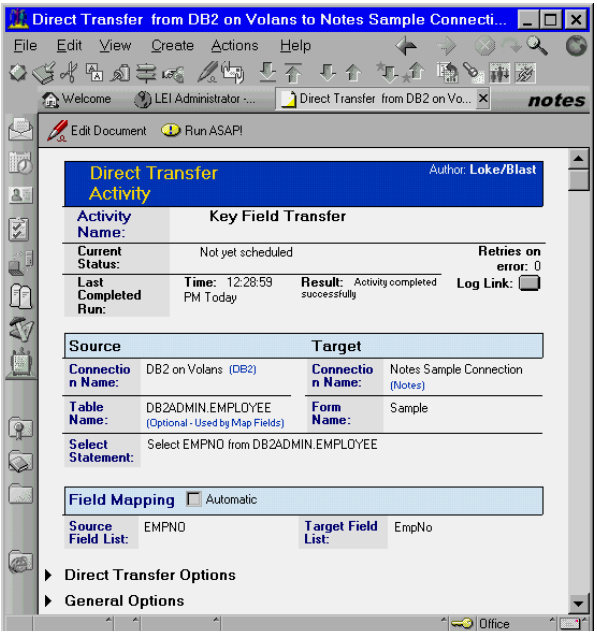
Connections

You administer connections (previously called “links” in NotesPump) to enterprise systems through connector documents in a connections view in the LEI Administrator database. Each document contains the necessary information to access one enterprise source.



Activities

You administer activities through activity documents in an activities view in the LEI Administrator database. Each document describes one activity.



The document presented to you depends on the type of activity. The following activities are provided:

- Admin-Backup: creates a backup copy of the LEI Administrator database (leiadm.nsf), and, optionally, the LEI Script Vault database (leivlt.nsf).
- Admin-Purge Log: purges the LEI Log database (leilog.nsf) of documents older than a specified number of days.
- Archive: moves data from one database to another, deleting the original records.
- Authority Propagation (AS/400 only): transfers database authorities from one database to another.
- Command: executes operating system, database and SQL commands.
- Direct Transfer: moves data from one database to another. Its execution can be immediate using the Run ASAP command or scheduled through activity scheduling options.
- DPROPR: moves data from a DPROPR CCD staging table in a relational database to any Enterprise Integrator supported database.

- Java: launches a Java application.
- Polling: polls a database and executes an activity if a condition exists.
- RealTime Notes: transparently accesses enterprise data in response to Create, Open, Update, and Delete document events on a Domino database. This activity is functionally similar to DECS.
- Replication: synchronizes databases by updating one database from the data in another.
- Scripted: executes a LotusScript script.

Development Client

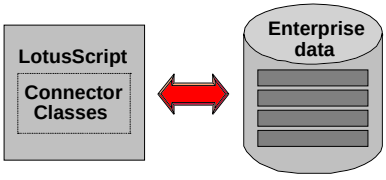
The LEI Development Client is a version of the LEI Administrator database and a local LEI server subset that resides on your development computer. The Development Client enables you to develop, run, and test LEI activities, and develop LotusScript agents using the LC LSX. The Development Client also enables you to use field mapping operations from a Notes client computer when constructing a Transfer Activity in the LEI Administrator.

Lotus Connector LotusScript Extension

The Lotus Connector LotusScript extension (LC LSX) provides programmatic access to Lotus Connector enterprise data through a common set of classes. The programmer uses a single model no matter the enterprise source. Programming provides more control and additional capabilities over DECS and LEI.

The classes use the same connectors as DECS and LEI to access the enterprise data.

This section briefly introduces the LC LSX. See the chapter "Connector Programming Interfaces" for details.



Loading the Connector Classes

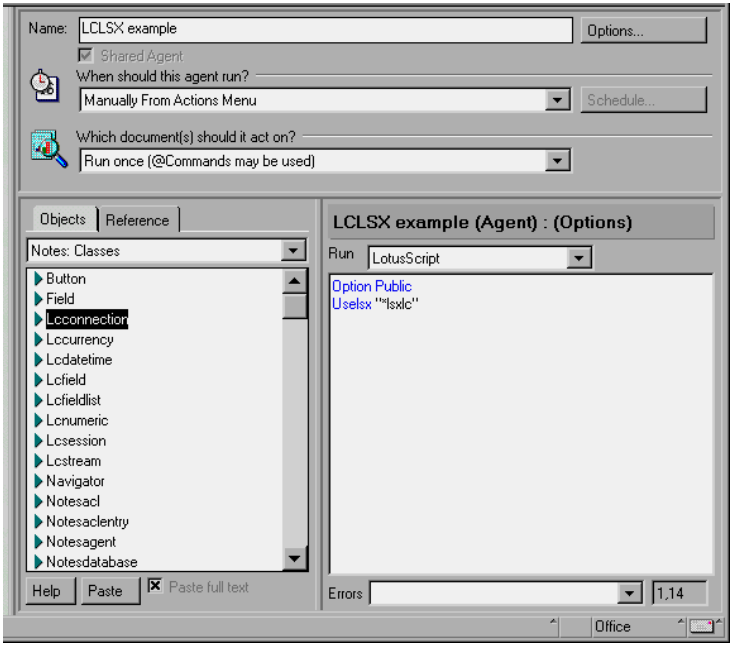
The following LotusScript statement, placed in the Options event, loads the LC LSX:

```
Uselsx "*LSXLC"
```

When the LC LSX is loaded, the LC classes become available:

- LCConnection: represents an instance of a connector to provide access to an enterprise system.
- LCCurrency: represents fixed point data.
- LCDatetime: represents date and time data.
- LCField: represents one or more values.
- LCFieldList: binds fields together with names and an implied order.
- LCNumeric: represents floating point data.
- LCSession: provides connection and error information.
- LCStream: represents text and binary data.

The LC classes appear under the Reference tab in the programmer's pane. The Uselsx statement should be entered under the Options event of the LotusScript object being coded.

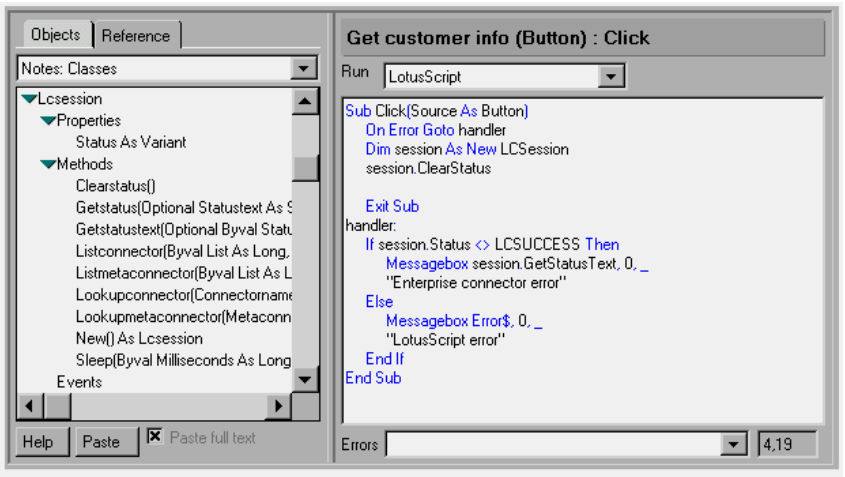


Coding a Script Using the LC LSX

The following sections present a simple script using the LC LSX. The script is attached to a button on a form that has the fields: Customer, Address, City, State, and Phone.

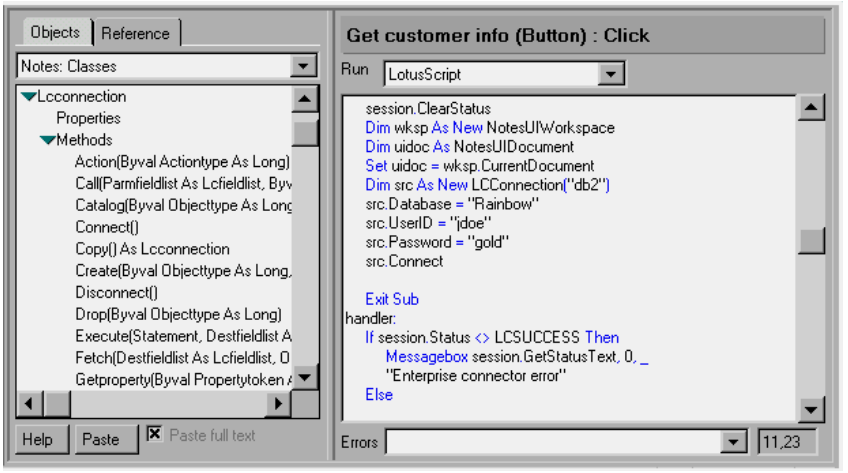
Setting Up a Session

The following figure shows how to use the LCSession class in conjunction with an error handler to handle LC errors.



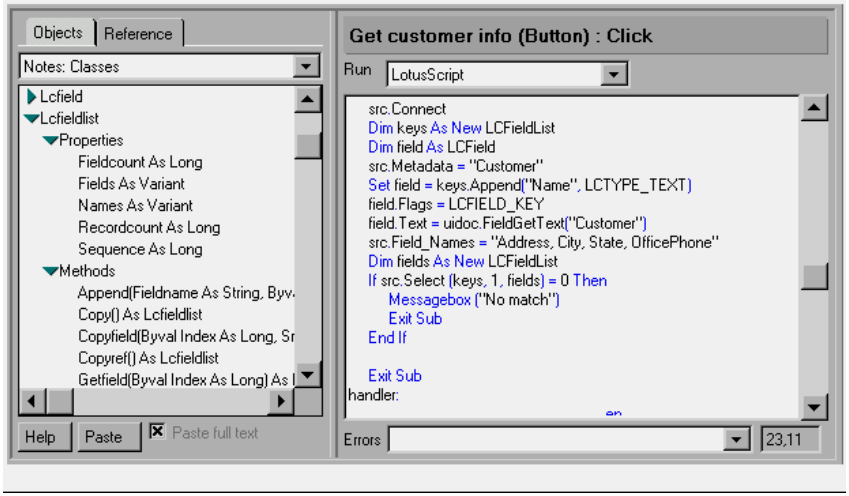
Connecting to a Data Source

Each enterprise connector has properties as specified in the documentation. The LCConnection class treats a connector property like a LotusScript property, although these properties do not appear under the Reference tab. The example in the following figure sets the Database, UserID, and Password properties for a DB2 database, then establishes a connection.



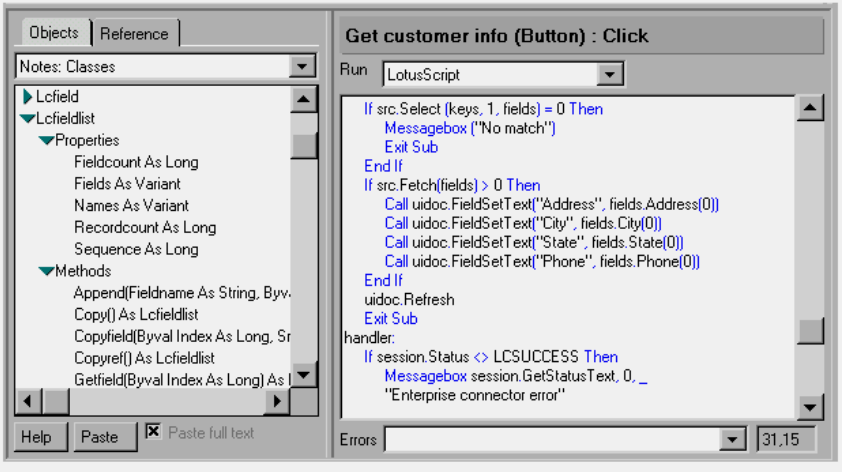
Selecting Data to a Result Set

The script sets up a key by appending the Name field from the DB2 Customer record to an LCFieldList object named “keys.” A Select operation locates the record or records whose Name field equals the Customer field of the Domino document, and builds a result set consisting of the Address, City, State, and OfficePhone fields of the DB2 Customer record. The result set is returned to a second LCFieldList object named “fields.”



Fetching Data from a Result Set

You can access each field in the result set with an LCFieldList property name that is the same as the field name. The result set property is an array indexed starting at 0. The example in the following figure expects the result set to contain just one record.



LEI Extensions

The following LC classes have expanded properties when LEI is installed:

- LCConnection Class
- LCFieldlist Class
- LCSession Class

The LCSession class also provides additional methods to support Activity level properties and logging.

When an LEI Scripted Activity runs, the script can access the property fields of the Activity form. When a Scripted Activity is used in conjunction with the LEI CGI program and an HTML form, the form's field values such as data entered by the user appear in the session object as read-only properties.

System-Specific LotusScript Extensions

The system-specific LotusScript extensions available in previous versions of Domino are still available. These LSXs are tailored for the enterprise system to which they apply, and may provide enhanced performance.

- LS:DO: the LotusScript Data Object provides full read and write access to ODBC data sources.

- DB2 LSX: enables Domino applications to access DB2. The DB2 LSX ships in Domino R4.6.4 and R5.0 on Windows NT only.
- MQSeries LSX: interacts with IBM MQSeries Messaging Software (IBM MQSeries), which can interact with any MQSeries-enabled application including CICS and IMS applications.
- SAP R/3 LSX: provides for bi-directional data flow between Lotus products such as Domino (Notes) and SmartSuite® applications, and the R/3 system.

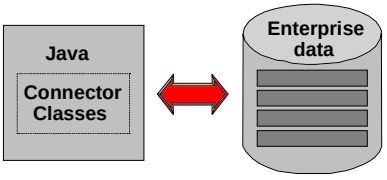
See the chapter "Lotus EI Product Updates" for more information. You can also find additional product information on the "Lotus Enterprise Integration" Web site:

`http://www.lotus.com/enterpriseintegration`

Java Classes

The Lotus connector Java classes provide programmatic access to enterprise data through a common set of classes. The programmer uses a single model regardless of the enterprise source. Programming provides more control and additional capabilities over DECS and LEI.

The classes use the same connectors as DECS and LEI to access the enterprise data.



This section briefly introduces the Java classes. See the chapter "Connector Programming Interfaces" for details.

Loading the Connector Classes

The following Java statement imports the connector classes:

```
import lotus.lcjava.*;
```

The LC classes become available:

- LCConnection: represents an instance of a connector to provide access to an enterprise system.
- LCCurrency: represents fixed point data.
- LCDatetime: represents date and time data.

- LCDatetimeParts: supports LCDatetime.
- LCField: represents one or more values.
- LCFieldList: binds fields together with names and an implied order.
- LCNumeric: represents floating point data.
- LCSession: provides connection and error information.
- LCStream: represents text and binary data.

Coding a Program Using the Connector Classes

Presented here is a short example that shows you how to connect to a data source. See the chapter "Connector Programming Interfaces" for an example of a query.

The example below is for an imported Domino R5.0 agent and also imports lotus.domino. For Domino R4.6, you import lotus.notes.

Create an LCSession object. Most of the LC methods throw LCException, which extends Exception. If you catch LCException, you can examine the LC error code through LCException.getLCErrorCode and LCSession.GetStatusText.

This example gets all the connectors accessible to the program by calling Session.listConnector iteratively.

```
import lotus.domino.*;
import lotus.lcjava.*;

public class jctest extends AgentBase
{
    public void NotesMain()
    {
    try
        {
            // Create LC session
            LCSession session = null;
            try {
                session = new LCSession(0);
                System.out.println("Session ready");
            }
            catch (LCException e) {
                int err = e.getLCErrorCode();
                String errmsg = session.GetStatusText(err);
                System.out.println(errmsg);
            }

            // List connectors
            int status = 0;
            int lcList = LCLIST.FIRST;
            LCStream lcConnectorName = new LCStream();
```

```
                Integer lcConnectorCode = new Integer(0);
                LCStream lcIdentifyFlagList = new LCStream();
                LCStream lcIdentifyNameList = new LCStream();
                boolean done = false;
                System.out.println("List connectors");
                do
                {
                    try {
                        // Get first or next connector
                        status = session.listConnector(lcList,
                        lcConnectorName, lcConnectorCode,
                        lcIdentifyFlagList, lcIdentifyNameList);
                    }
                    catch (LCException e) {
                        int err = e.getLCErrorCode();
                        String errmsg = session.GetStatusText(err);
                        System.out.println(errmsg);
                        done = true;
                    }
                    // Print connector name
                    String name = lcConnectorName.toJavaString();
                    if (name != null)
                        System.out.println("\t" + name);
                    lcList = LCLIST.NEXT;
                // Loop while listConnector returns no error
                } while
                  (status == LCFAIL.NO_ERROR && done == false);

        }

    catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Additional Enterprise Integration Tools

Additional tools are outlined below.

Lotus Connector Toolkit

The Lotus Connector API supports both building and using connectors. It supports high performance, allows product-specific capabilities, provides independence from platform and locale, provides independence from the format of the data being accessed, and provides programming language independence.

The Lotus Connector API classes can be divided into three groups:

- Data classes are in three groups – number, datetime, and stream.
- Metadata classes define properties for and contain data. There are two metadata classes – Field and Fieldlist.
- Context classes define the LC API state and the context within which other classes are used. There are two context classes – Connection and Session.

The Lotus Connector API works in both single-threaded and multi-threaded environments.

The toolkit provides connector template LCX source code for building connectors.

ActiveX Data Object

You can access enterprise sources using Microsoft's ActiveX Data Object (ADO). You must install and register the ADO objects on a Windows or NT system. LotusScript will invoke OLE Automation to interact with the ADO Objects. Use the LotusScript CreateObject statement to access the object hierarchy. See the chapter "Enterprise Integration Additions" for details.

MQSeries Enterprise Integrator

The MQSeries Enterprise Integrator (MQEI) provides tools for developing and managing applications running under CICS and IMS. These include an EI LSX, a built-in capability to access unmodified CICS and IMS applications, enhanced message-building facilities, and security integration. See the redbook, *Lotus Solutions for the Enterprise, Volume 4: Lotus Notes and the MQSeries Enterprise Integrator*, SG24-2217.

JDBC

You can access enterprise sources using Sun's Java Database Connectivity (JDBC) API. JDBC permits you to send SQL statements to enterprise systems and retrieve results. The JDK 1.1 includes JDBC (java.sql package). See the chapter "Additional Integration Methods" for details.

NotesSQL

NotesSQL is an SQL API to Domino with full Level I ODBC 2.0 compliance and Level II extensions. See the chapter "Enterprise Integration Additions" for details.

Domino Driver for JDBC

The Domino Driver for JDBC is a Type 2 JDBC driver that allows you to access Domino databases from Java. See the chapter "Additional Integration Methods" for details.

Documentation

Refer to "Related Publications" in the back of this book for:

- Other redbooks about Lotus enterprise integration
- Documentation supplied with DECS and LEI

You can also visit the Lotus Enterprise Integration Web site

`http://www.lotus.com/enterpriseintegration`

to search for information and software.

Summary

This chapter has given an overview of the Lotus enterprise integration architecture and the products using the architecture. We have explained how different products can use the same set of connectors to access other data sources when they are developed under the architecture. We have also described other Lotus products for enterprise integration to give a complete overview of Lotus tools available for enterprise integration.

Chapter 2
Domino Enterprise Connection Services (DECS)

Domino Enterprise Connection Services (DECS) is a new function that has been added to Domino R4.6.3 and R5.0. It runs as a server task within Domino and provides live access to external enterprise data sources. DECS administration is visually mapped (forms-based) for easy setup. Those of you familiar with Lotus NotesPump (now known as Lotus Enterprise Integrator), will soon recognize this interface and easily adapt to the features. DECS provides a unique function similar to Lotus Enterprise Integrator's real-time Activity. It runs in much the same way, by capturing Domino document events and performing the functions you set up within the DECS administration database.

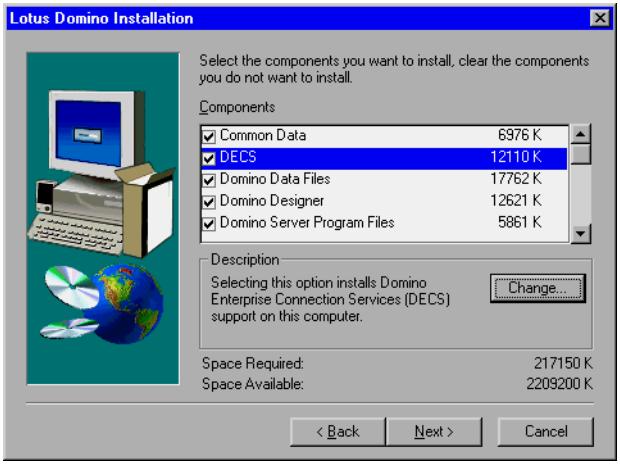
In this chapter we will discuss:

- How to install DECS
- How to set up DECS
- Examples of DECS usage (query, create, update and delete data)

Installing DECS

DECS can be installed automatically with Domino. To make sure you are installing DECS, choose the "Domino Server" install as shown in the following figure.

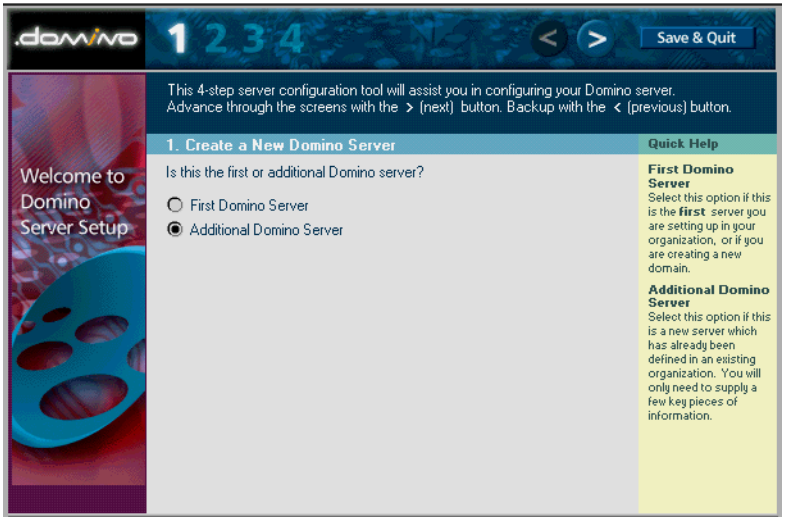
Click Customize and you will see the following dialog box:



The DECS check box is already selected for you. By clicking the Change button you will see the components that install with DECS, however they will already be selected for you. Click Next> and follow the rest of the Domino installation process.

Once Domino is installed on your server, start it. To start your Domino server on your operating system, you must be familiar with that operating system's method for starting tasks. The first time you start Domino after the

software install process, Domino will proceed with the server setup process. The setup process uses a Lotus Notes run-time client to help you configure your Domino server. This process uses the Domino database setup.nsf. The first screen you will see is represented in the following figure.



Click the radio button that best describes your Domino setup. We have selected Additional Domino Server for our example. After this screen the process takes two paths. Both paths allow you to set up DECS. They are:

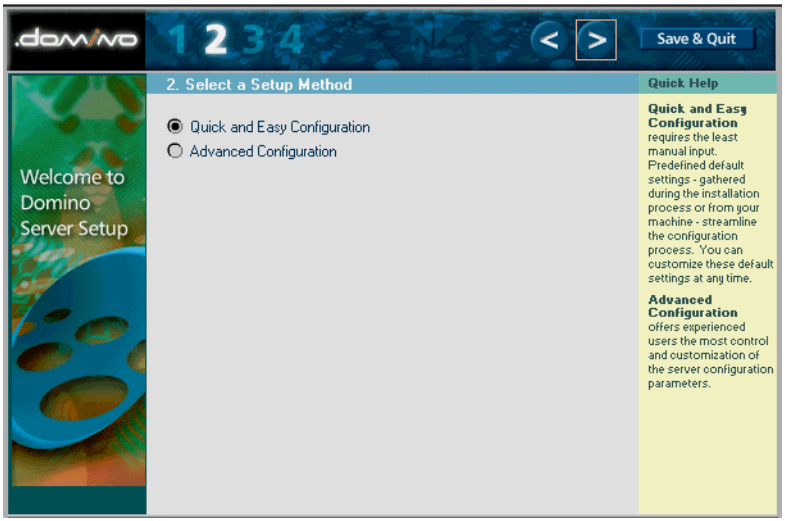
1. Quick and Easy Configuration
2. Advanced Configuration

Both of these paths indicate whether DECS is requested by your setup. They will be discussed in the next sections.

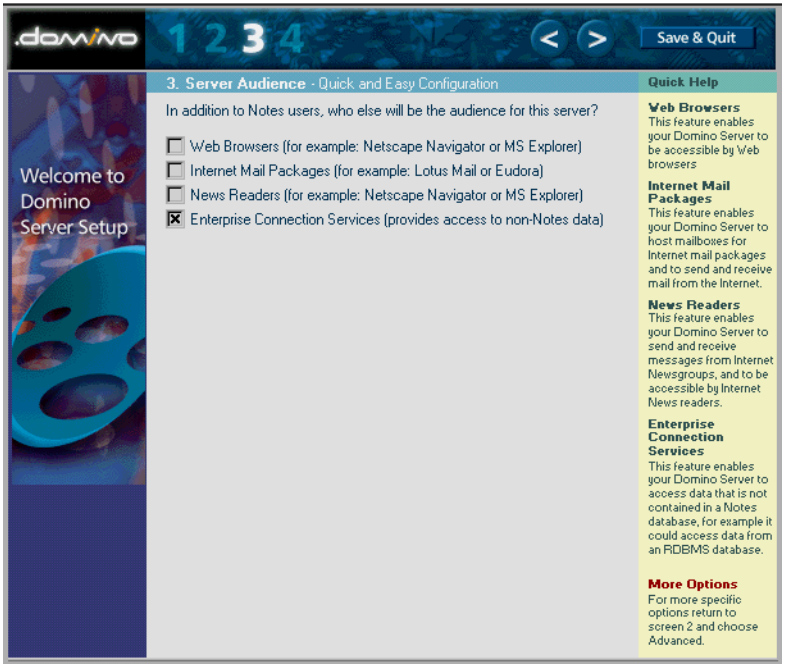
Domino Quick and Easy Configuration

The Quick and Easy Configuration option allows you to select the least number of options to get your Domino server running. As the title suggests, Domino will be up and running quickly, and once you are familiar with your server, you can further configure the more advanced options.

To move forward from screen 1, click the > (next) button to move to screen 2 of your Domino setup, shown in the following figure.



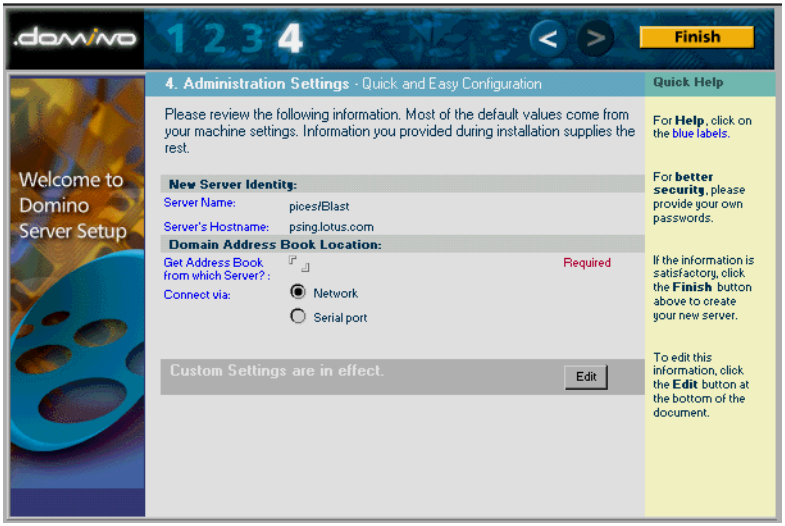
Select the Quick and Easy Configuration radio button. Click the > (next) button to proceed to screen 3 of your Domino setup.



The Server Audience screen is the first point in the Domino setup where you specify if you require DECS to run on the server. Click on the check box that says Enterprise Connection Services (provides access to non-Notes data).

When you pick this option, DECS will be added as a server task. If you do not select this option, do not worry; you can manually add the task later.

Continuing with the Quick and Easy Configuration, click the > (next) button on the screen shown in the previous figure. You will see screen 4 , following.



Fill in the required fields for screen 4. The setup parameters for this screen do not impact DECS.

Note We highly recommend that a network connection be used for any Domino server running DECS.

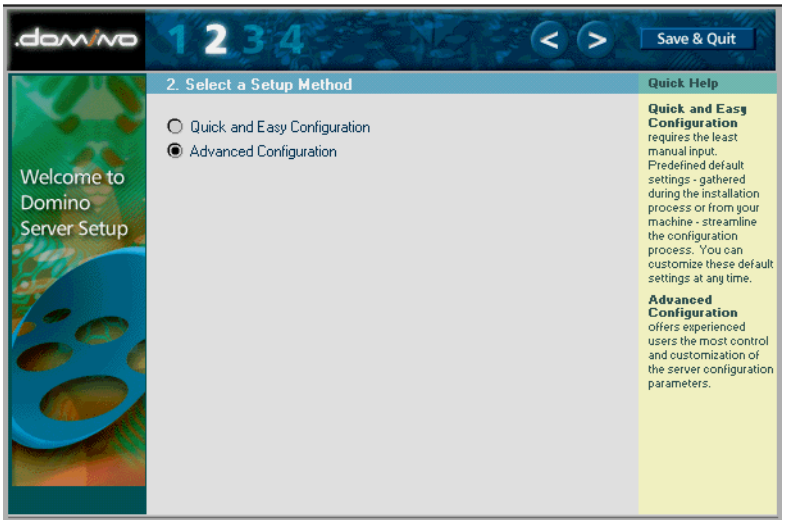
Click Finish on screen 4 to complete the Domino server configuration.

Now you are ready to run Domino and DECS. If you do not want to read about how to set up DECS through advanced configuration you can go to the section “Starting and Controlling DECS.”

Advanced Configuration

The Advanced Configuration path allows you to preset many settings that you are already familiar with. This is a common setup path when Additional Domino Server is selected on screen 1.

From screen 1, click the > (next) button to proceed to screen 2, shown in the next figure.

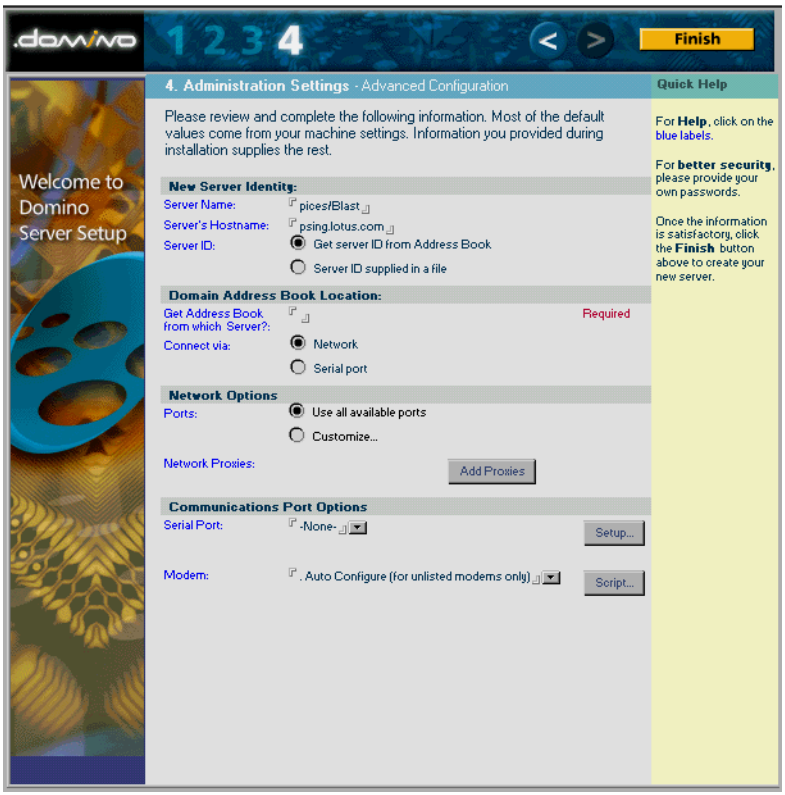


Click the Advanced Configuration radio button to select this path for Domino setup. Click the > (next) button to move to screen 3, shown in the following figure.

To make sure DECS is set up to run on your Domino server, click the check box Enterprise Connection Services on screen 3 to allow DECS to run whenever you start Domino. Selecting this option tells the setup process to

include the DECS server task in your NOTES.INI file. This means you have full control of DECS no matter what you select here. You can easily edit your NOTES.INI file to include the task later. You can also run the task manually from the Domino console. For further detail, refer to the “Starting and Controlling DECS” section in this chapter.

Complete any other setup options you require for your Domino server on screen 3. Click the > (next) button to move to screen 4, shown in the following figure.



None of the options on screen 4 are related to DECS so changing the selections on this screen will not affect your settings for DECS.

Note It is highly recommended that you use a Network connection for your Domino server when running DECS.

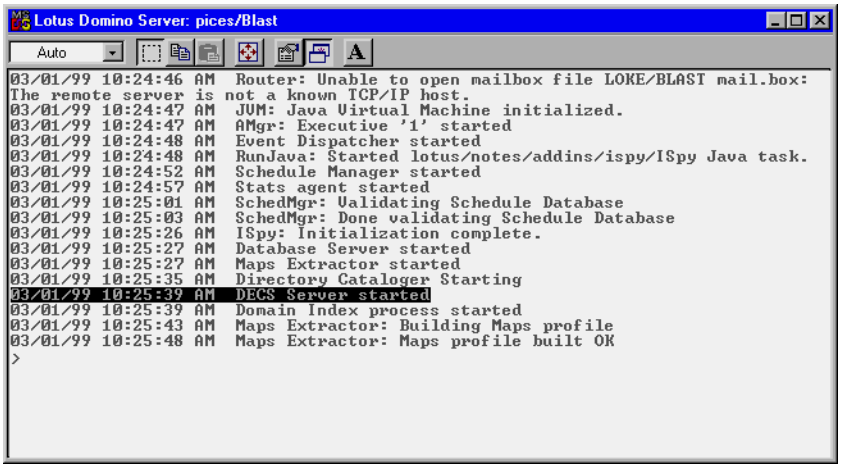
Click Finish to complete your Domino server setup.

You are now ready to run Domino and DECS. The “Starting and Controlling DECS” section in this chapter will explain how DECS is started and how you can control it.

Starting and Controlling DECS

Domino Enterprise Connection Services (DECS) will run as soon as you start the Domino task. Once you complete the setup process, using either of the above paths, DECS is ready to start with Domino.

The Domino console shows you the status of DECS during startup. Once Domino has completed the startup process, the following screen, or a similar screen, will appear.



In the screen shown, the highlighted line informs you of the status of DECS.

The DECS task is controlled within the NOTES.INI file. The ServerTasks line in the notes.ini file shows you all of the tasks that Domino will run on a continuous basis. The DECS task will be found on this line. To control the DECS task manually, you should remove the DECS entry from the ServerTasks line. After restarting Domino, DECS would not start in this situation. Once DECS is removed from the notes.ini file, you must control DECS manually, or change the notes.ini to include the DECS task again.

To start the DECS task manually, type the following at your Domino console:

```
LOAD DECS
```

Domino will respond with a task response indicating that the connection has started. The first time DECS is started, DECS will create the DECS Administration Database from the administration template that was included as part of the Domino installation process.

To manually stop the DECS task, type the following at your Domino console:

```
TELL DECS QUIT
```

Domino will respond indicating that the connection server has shut down.

DECS Variables in the NOTES.INI File

The following NOTES.INI entries can be added to help control DECS:

- DECSTranslation

  All text data within Domino databases maintains a reference to its own character set. This reference is determined automatically by the connector when the data is fetched from a database. If the data is inserted into a database with a different character set, DECS automatically translates the text data just prior to storing it. This process maintains maximum performance by eliminating any unnecessary translation. Performance may be further enhanced through DECS' three levels of translation support: no translation, LMBCS only, or full translation.

  The three available settings are:

  - DECSTranslation=0
    This setting will not translate data (except unicode). This setting can be used if all the connectors are compatible with the Lotus Multi-Byte Character Set (LMBCS), that is, ASCII characters.
  - DECSTranslation=1
    This setting is useful when all of the connectors being accessed use compatible, non-LMBCS, character sets.
  - DECSTranslate=2
    This setting is the default. All data will be translated no matter what the character sets are.

  Caution Do not use this entry if you are not sure which connectors you will be using. The default setting is the right choice.

- DECSNativeText

  This entry allows your Domino server's native character set to be overridden. The list of supported character sets for DECS and the Domino Connectors can be found in the "Character Sets" Appendix. The default setting, which is not required as an entry, is:

  ```
  DECSNativeText=Native
  ```

  The "Character Sets" Appendix specifies the character set as used within the Domino connectors LotusScript extension (LSX). To apply these settings for DECS, remove the LCSTREAMFMT_ prefix.

  Note Be careful with different integration products, as they may use compatible character sets that are not native to the operating system. This means that one or more of these products may require non-native character translation. This only applies if you mix integration products on one server.

- DECSCenturyBoundary:

  This entry allows you to control how the year in a text string is converted to a Domino datetime field when the year contains only two digits. Values greater than or equal to the century boundary are considered to be in 1900s, values less than that are in the 2000s. There are three ways to set this:

  - DECSCenturyBoundary=0
    This setting will always use "19" as the century number since all two digit years are greater than zero.
  - DECSCenturyBoundary=50
    "50" is used as an example in this case. If the value is in the range of one to 100, the following rule applies: The century number is "19" if the two digit year is greater than or equal to the century boundary; otherwise use "20" (for the year 2000). The default value for the century boundary is "50." This entry is not required in the NOTES.INI file if the default is sufficient.
  - DECSCenturyBoundary=101
    This value forces all dates with a century number of "20" (for the year 2000).

- EXTMGR_ADDINS

  The extension manager entry is used if DECS and NotesPump 2.5a (or the Lotus Enterprise Integrator) are being used on the same Domino server. For the NotesPump real-time function and DECS to work together, this entry on Windows NT would be:

  ```
  EXTMGR_ADDINS=ndecsext.dll, lnpext.dll
  ```

  For the Lotus Enterprise Integrator (LEI) real-time function and DECS to work together, this entry on Windows NT would be:

  ```
  EXTMGR_ADDINS=ndecsext.dll, nleiext.dll
  ```

  These combined entries are not required when DECS runs alone on the Domino server. The default entry for DECS only is:

  ```
  EXTMGR_ADDINS=decsext
  ```

  This entry is automatically added to the NOTES.INI file when Domino is set up.

DECS Administration

Before we can use DECS in a Domino application we need to set up DECS for that application. To do this:

1. Create a connection to the external data store if it is not already defined.
2. Create an Activity document where we specify:
   - Which documents in which database to monitor.
   - Which events to monitor.
   - Which fields are key fields externally and in Domino.
   - How external fields or records map to fields in Domino.
3. Initialize key. We need to have documents in our Domino database with key values for all the records on the external system we want to work with. Remember, to have DECS do something a document event has to be triggered. The DECS Administration application has an action called "Initialize Keys" that creates documents in the Domino database with data only in the key field. These documents are called "stub" documents.
4. Finally, the Activity has to be started to begin monitoring the documents in the Domino database.

In this section we will explain how to set up connections and activities using the DECS Administration database.

The DECS Administration database offers a wizard-based interface to create connections and activities. Online help aids you during the process when required.

Supported Data Sources

DECS provides real-time connections to the following data sources:

Source	*Prerequisites*
IBM DB2	• DB2 Connect Personal Edition • DB2 Enterprise Edition or • DB2 Client Application Enabler (CAE) 2.1.2 or later • In addition, to connect to DB2 on an AS/400 or mainframe, a DDCS gateway must be installed.
EDA/SQL	• EDA/Client software for the host operating system. The EDA/Client version must be release 3.2 or later and must be 32-bit on Windows NT and OS/2. • An EDA server on the platform where the EDA supported database resides • Connectivity to the EDA server

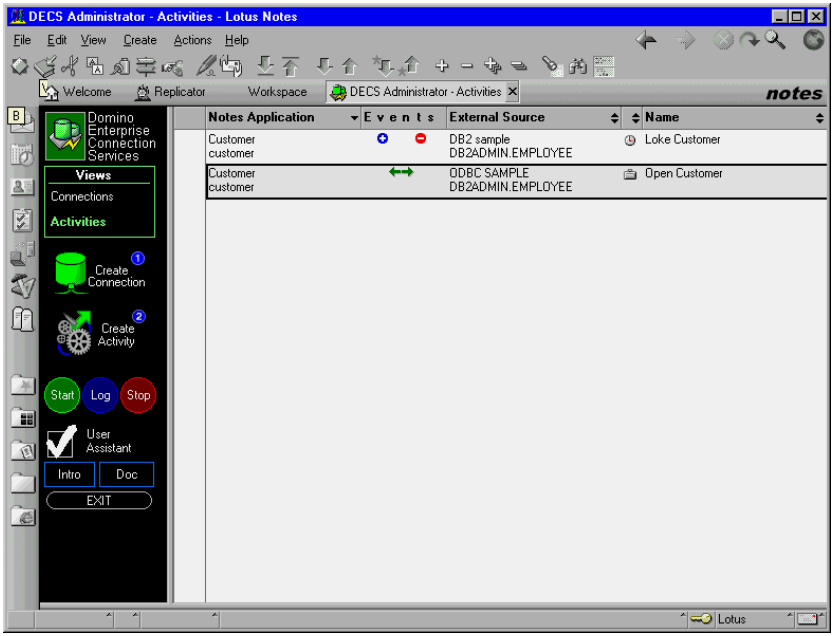
continued

Source	*Prerequisites*
Open Database Connectivity (ODBC)	• The ODBC driver appropriate to the operating system • The driver must be 32-bit on NT and OS/2 • The ODBC driver must be thread-safe • The ODBC Administrator must be present • There must be correctly defined ODBC data sources in the ODBC Administrator
Oracle	• With an OS/2-based server: Oracle SQL*Net version 2 • With a Windows NT based server: Oracle SQL*Net version 1 or 2 • In either case SQL*Net must be the same version as SQL*Net installed on the Oracle data server. A network connection must exist between the server machine and the Oracle data server machine via SQL*Net. • Native Oracle connectivity support requires Oracle version 7.2 or later • OS/2 works only with Oracle 7.3 • Oracle version 7.3 and HP-UX: You must obtain Oracle fix for bug #441647. This patch applies to Oracle's libclntsh.sl • Oracle 8: NotesPump and DECS link with Oracle 7.3 libraries, which use SQL*Net for the communications layer. If you are using Oracle 8 with DECS or NotesPump, you must install Oracle SQl*Net. You may use the SQLNET Easy Config to configure SQl*Net.
Sybase	• With an OS/2-based or Windows NT-based: System10 Netlib. • A network connection must exist between the Domino server and the Sybase SQL server via Netlib.
File System	• No additional requirements other than access to the operating system's file directory.
Notes	• This connection exists by default. Unlike LEI, DECS does not require connection documents to Domino databases.

Data source connectivity testing is available for the DECS relational connectors. You can run the program LCTEST from a command line to confirm connectivity with your data source.

The DECS Navigator

The DECS administration database navigator gives you an easy interface to create your data source connections and activities. The following figure shows the navigator along with sample activities.



The following list will help you identify each region of the navigator:

- Click Connections to see a view of available Connections.

- Click Activities to see a view of real-time Activities.
- Click the Create Connections image to create a new Connection document for an external data source. This launches a wizard that prompts you through the process of defining a Connection to an external data source. The small, blue "1" indicates that the wizard is active. You turn the wizard on or off through *User Assistant* which is explained below.

- Click the Create Activity image to create a new real-time Activity. When the wizard is active (indicated by the "blue 2"), you are prompted through the process of defining a real-time Activity between your Domino application and your external data source. With the wizardrea turned off, you are presented with a blank real-time Activity document that you can edit directly.

- Click Start to begin monitoring the documents specified in your currently selected real-time Activity. This has no effect if your current selection is already started. The Start button is disabled when in the Connection view.

- Click Log to see the status of the currently selected real-time Activity. If the current selection is running, you will see the start time and other current status. If the current selection is not running, you will see the results of the most recent run. The Log button is disabled when in the Connections view.

- Click Stop to end your currently selected real-time Activity. This has no effect if the current selection is not running. The Stop button is disabled when in the Connections view.

- Click the User Assistant Check to switch the User Assistant on or off. When turned on, this provides additional help and enables the DECS wizard. This is useful if you are a first time user. The DECS wizard guides you through creating the real-time Activity document and provides you with information to help in the creation and editing of the document.

- Click Intro to view the DECS Administrator Database's Using Document.

- Click Doc to display the DECS online user guide.

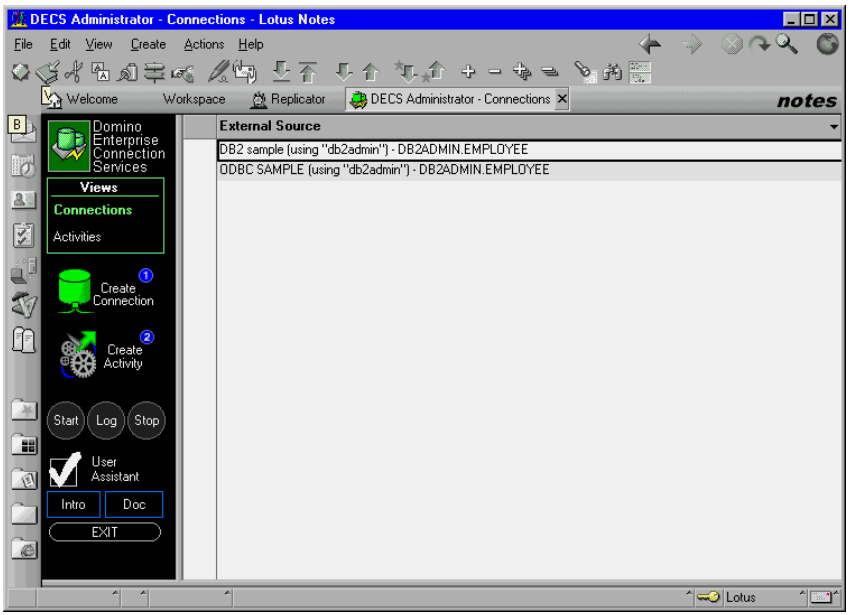
  

- Click Exit to close your DECS Administration database.

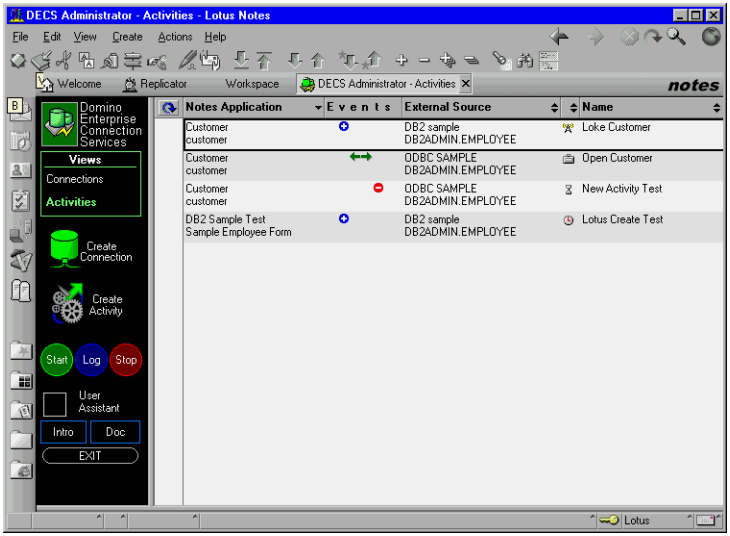
The DECS Connections View

All connection documents can be seen on the Connections view. These documents are named by DECS using a combination of external data source type, database name and the table name. You will also see any user name associated with accessing the external source. The following figure is a sample of the Connections view.



The DECS Activities View

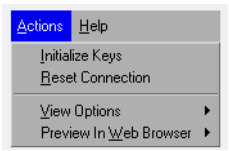
The DECS Activities view provides you with a graphical approach to activity management. The Activities view displays a list of Domino applications that have been enabled for real-time access (read, write or both) to an external Domino Connector system. There are several icons used in this view to show event triggering and activity status. Click Intro on the DECS navigator to get a complete description of the view graphics, as shown in the following figure.



The screen displays a collection of activities at various stages. This view was created to isolate these stages, but the first three activities can be collapsed into one activity. Combining or separating events for one database is an option open to you when you create the activities. In business settings where greater control is required for each event, activities can be separated to minimize any outage impact created by one activity.

DECS Administrator Menu Commands

Two commands are provided to help you manage your activities. They are shown in the following figure.



Command	*Description*
Initialize Keys	Only use this command to initialize a Domino database after a new activity is created. This command will populate your Domino database from the external data source. If you selected that some or all of your real-time fields be kept in the documents (selected via Options - General - Data Storage), then those settings will be obeyed during initialization. If you created any filter formulas, you will be prompted for a conditional formula for initialization.
Reset Connection	This command will restart your selected started activities.

Defining Connections in DECS

Domino Enterprise Connections Services (DECS) requires connection documents to enable integration with external data sources. These connections use the Lotus Connector architecture (a common set of connectors available to all Lotus integration products). Essentially, these Connection document settings provide the Domino Server with valid login information to use to connect to external data sources.

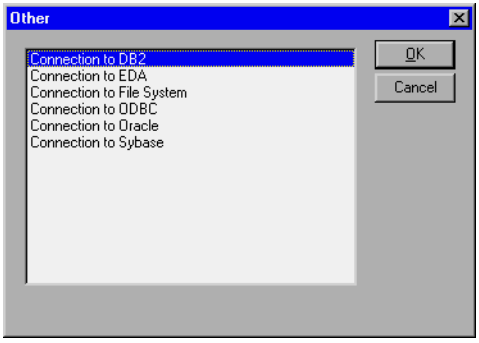
Within the DECS Administration database, you have the option to create connections using the DECS User Assistant. This assistant, termed the wizard, can be turned on or off for this process. The following section will discuss connector document creation using the wizard. Before you start this, you must check the User Assistant box in the DECS navigator.

Creating Connections with the wizard

The following steps will guide you through the connection document creation process using the DECS wizard:

1. From your Notes R5.0 client, open your Domino server's DECS Administration database.
2. With the User Assistant (wizard) turned on, click the Create Connection image.

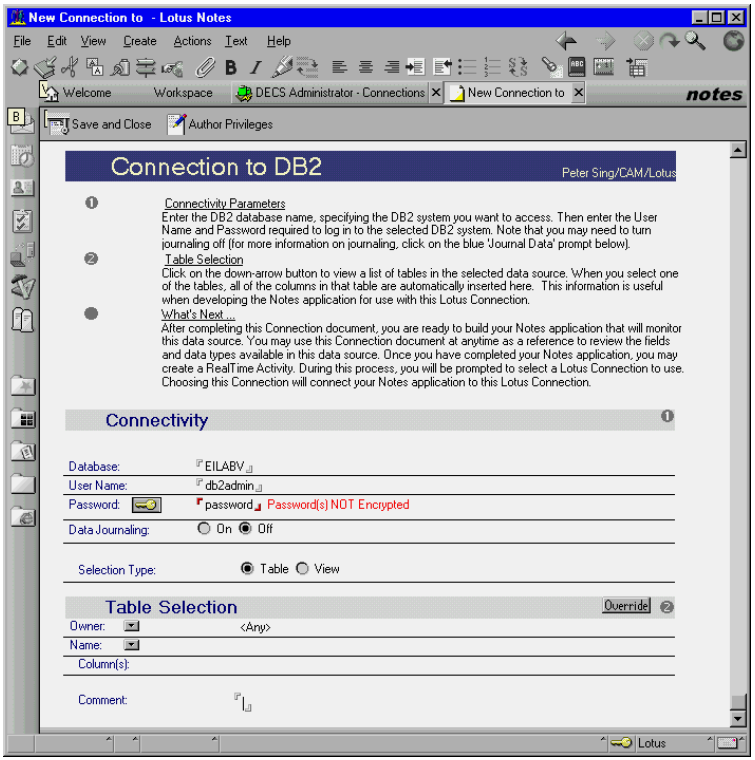
3. You will see the connection type selection dialog, shown in the following figure.



Select the connection to the external data source that you are using and click OK. In this example we connect to DB2.

Note You must establish communication to your external data source prior to creating the connection document. You can use the LCTEST utility to test your connection before creating the DECS connection document. Start LCTEST from your operating system prompt and follow the directions.

4. You will be presented with the connection document form shown in the following figure.



Fill in the first set of fields. For each field, you can click the caption (blue text) to get more information regarding the input required. The fields in the Connectivity section are:

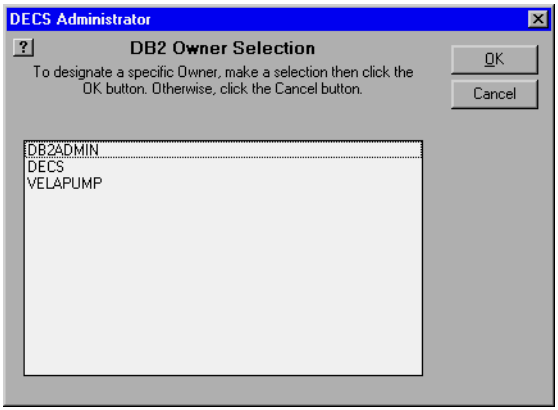
- Database: The alias name for the external database.
- User Name: The user ID you are using to access the external database.
- Password: The User Name's password. Click the "key" button to the left if you wish to encrypt the password. You will need an encryption key to do this.
- Data Journaling: Some database systems keep a journal (or log) of all changes to its databases. This is used for various purposes such as tracking changes for recovery reasons, mirroring of changes between

systems, keeping remote systems informed about the current value of a data area, usage tracking (billing) and so on. Journaling is required to run under commitment control. The default setting for all DECS activities is to run with commitment control. If the DB2 table is not journaled you must select Data Journaling Off. You will only be able to read non-journaled data in this case. If the table is journaled, the default access is read/write. If you leave this option on, and cannot write data, check to see if your external source is journaled.

- Selection Type: DECS can read data from external tables or views. Here View refers to a relational view, not a Domino view. If your external data source is defined, but not your Domino database, choose Table and develop your Domino database using an external table to Domino form mapping. The same mapping applies if your Domino database is defined but your external data source is not. Choose View if both the external data source and your Domino database are already defined and their mapping is not table to form.

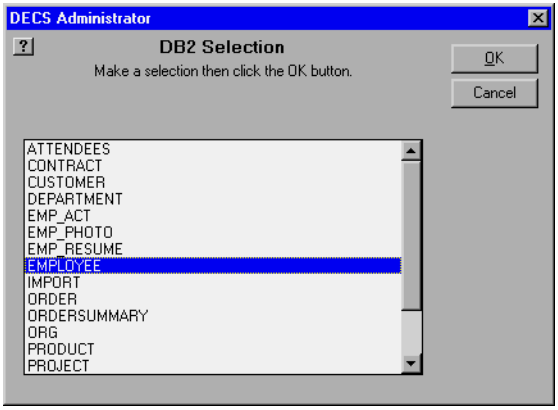
Note When using a selection type of View and an activity that updates or creates data, you must include all unique keys of the tables that the view includes in its creation SQL.

5. Click the Owner drop-down button in the Table Selection section of the connection document. You will be prompted to select the table or view owner for this connection.



The owner is used as a high-level prefix for the table or view names that DECS will be connecting to. Select the owner of the table you will be accessing and click OK. The default owner is “anyone,” which leaves the field blank and the table you access will not be prefixed. This works for accessing DB2 alias names or names that are unique to the DB2 subsystem.

6. Click the Name drop-down button. You will see a dialog box that displays all the tables or views the selected owner has access to, as shown in the next figure.



Select the table or view you wish to access with this connection and click OK. The Name field will be filled in with your selection and the Columns field will contain all of the columns defined for that table or view.

You can also click the Override button to type in the table or view name if it is not available in the selection, or if you prefer to bypass the DB2 overhead that DECS is performing when displaying the list.

7. Input anything you want for the Comments field.
8. Click Save and Close to save the connection and return to the DECS Administration navigator.

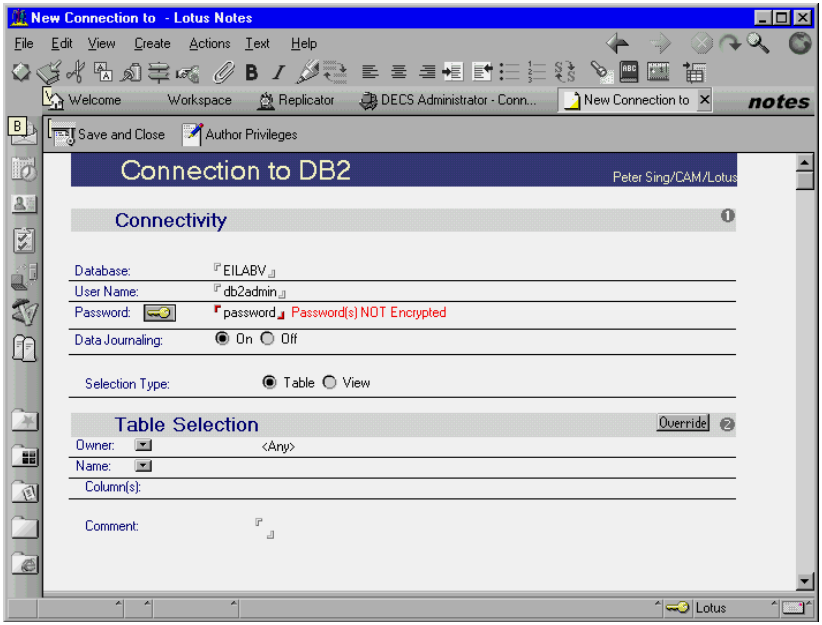
Creating Connections without the wizard

Once you are familiar with creating connections, you will find it much easier to turn the User Assistant off and build connections directly. With the wizard off, the connection document creation process is very similar. The following steps will guide you through the standard process for creating connection documents:

1. Proceed with steps 1, 2 and 3 as previously described to arrive at the step where you select a connection to DB2.

Note For step 2 above, the blue 2 icon in the Create Connection button will not be present.

2. You will be presented with the connection document form shown in the following figure.



This connection form is very similar to the wizard version of the form. This version is missing the introduction text only.

3. Proceed with steps 5, 6, 7, 8 and 9 listed in the wizard section.

Building the Notes Application

DECS does not require a connection document to a Domino database, unlike its predecessor, the NotesPump real-time Activity. The connection in DECS is defined within the DECS real-time Activity. With this in mind, you must create the receiving Domino database before you create the DECS Activity.

This section will use a sample Domino database developed at Lotus to explain the process of creating a Domino database that DECS can monitor. To briefly explain the type of database required, you will need the following elements:

- Lotus Domino Designer installed on your client workstation
- Any standard database (File-Database-New to create one)
- A Domino form that contains at least one field that maps to the external data source unique key

Note Although the key field is the only field required, you will want more fields to return retrieved data to the form, and therefore show that DECS working. Create any number of fields such that the external source table or view can map one column to one field on your form. You must set the field attributes in Domino to be acceptable by the external data source. For

example, a Domino text-editable field must map to a CHAR or VARCHAR column in the external table or view.

Important Most external data sources have fixed length columns. DECS can read these columns without any problem; however, the reverse is not true. If you want DECS to update the external data source, you must control the field lengths in Domino using Input Translation or Input Validation formulas.

Example Application — Millennium Cafe

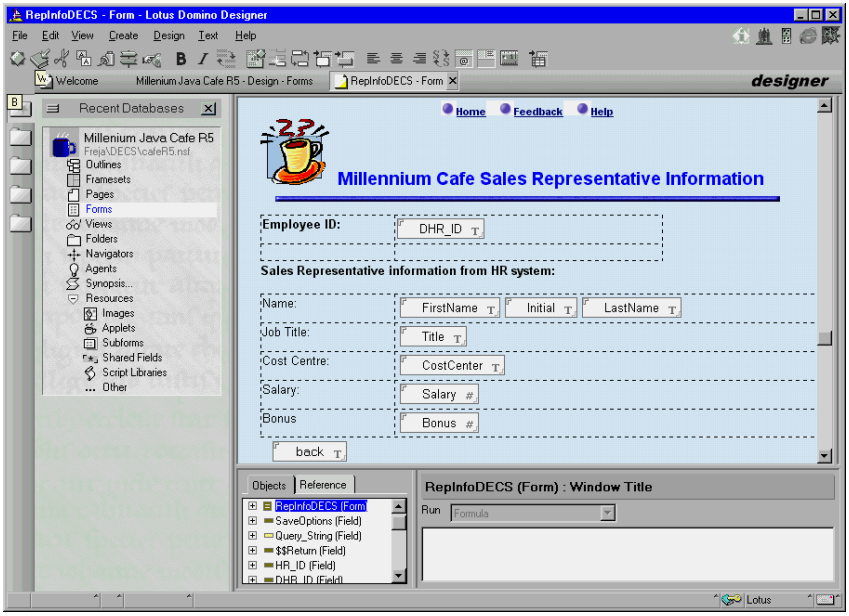
The Millennium Cafe application has been created by Lotus to demonstrate Domino's ability to connect to external relational data sources. We will use this database in our examples. The database can be downloaded from the IBM Redbooks Web site:

`http://www.redbooks.ibm.com`

The database is in the additional materials section of the redbook, *Lotus Domino R5.0 Enterprise Integration: Architecture and Products*, Lotus part number CT6QUNA, IBM form number SG24-5593. However, you do not need to have this database to understand the examples.

The Millennium Cafe in our example was developed using Domino Designer Release 5.0.

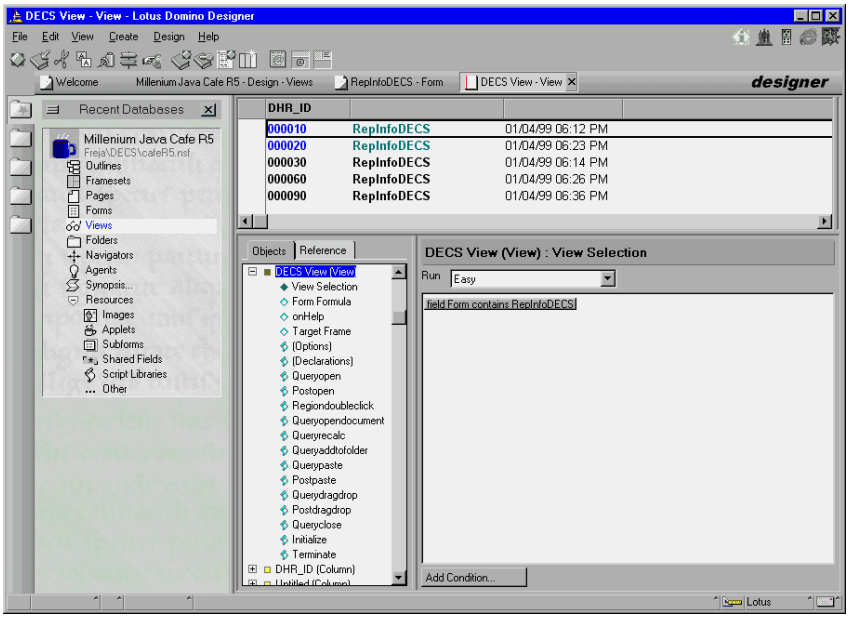
From Domino Designer the RepInfoDECS form looks like the following figure.



The field mapping from this form to the EMPLOYEE table in DB2 is shown in the next section. The DECS activity creation process has a very powerful visual mapping exercise. It is easier to create a Domino form whose field names are identical to those in your external data source. This example shows the ability of DECS to map not only to different field names, but to a subset of the fields as well. This is useful when you want to control input to your external data source. For example, if certain fields must be managed by your Human Resources center, but others can be modified by your employees, DECS can be mapped to the employee-only fields and your HR system will control the others. This is very useful when controlling confidential information as well.

For further detail on this example, please see the DECS Examples section.

Once your form is created, you will need a view to see the documents. The Millennium Cafe demonstration database contains a view just for this purpose. The view DECS View, shown in the next figure, is used in this example.



The first column of this view displays the key value. Any other columns that you require on any DECS view are your choice.

Note The fields viewed in a view column must exist on the “stub” document in your application database, otherwise the view column will be empty. See the “General Options” section in this chapter for further details.

Defining Activities in DECS

You have a connection document. You have your Domino database that will be used to retrieve data from your external data source. Now all you have to do is create the DECS real-time Activity document. This section will discuss the two ways to create an activity document:

- Using the User Assistant (wizard)
- Without the User Assistant (no wizard)

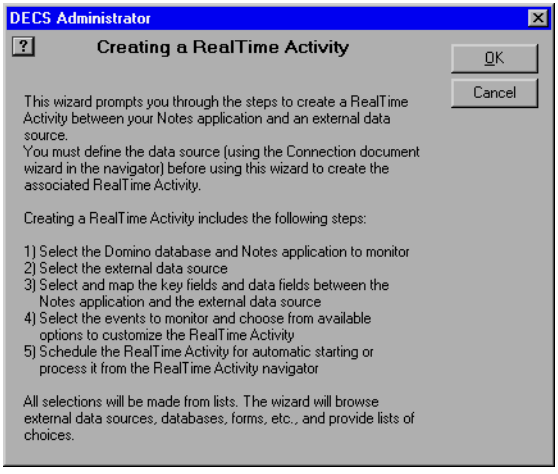
Creating Activities with the wizard

Note The activity created in this section will link the previously created connection document to the Millennium Cafe RepInfoDECS form. The activity will only be created once even though the following sections may suggest that it was created twice.

Follow these steps to create a real-time Activity using the DECS wizard:

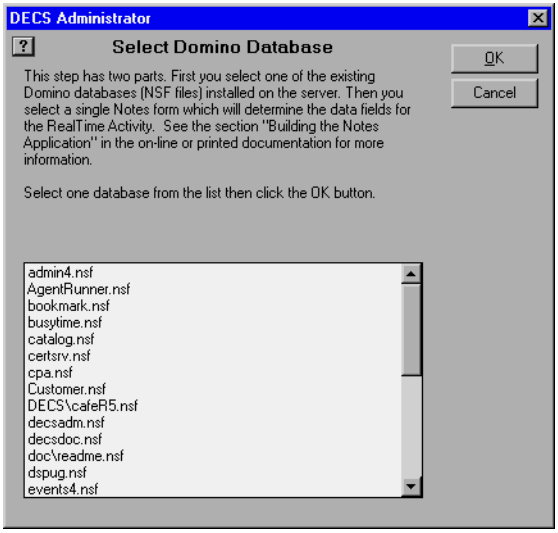
1. From your Notes R5.0 client, open the DECS Administration database on your Domino server.
2. With the User Assistant (wizard) turned on, click on the Create Activity icon.

3. You will be presented with an information box introducing you to the DECS real-time Activity wizard.



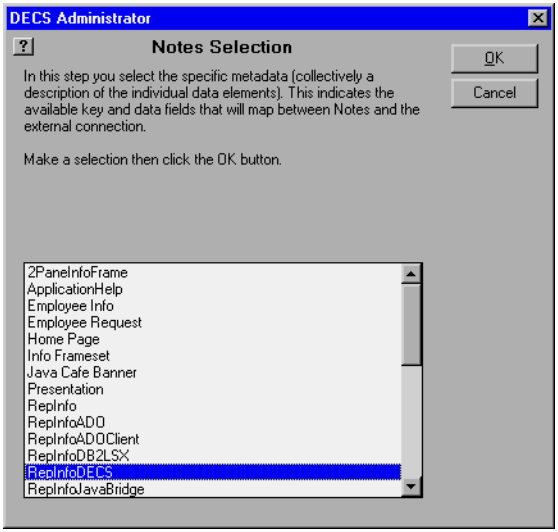
Read the information if you are a first-time user and then click OK. If you require further help, click the ? button in the upper left corner.

4. After you click OK in the dialog box, DECS will connect to your Domino server. Using the Domino Connector for Notes, DECS retrieves a list of databases currently installed on the Domino server, as shown in the following figure.



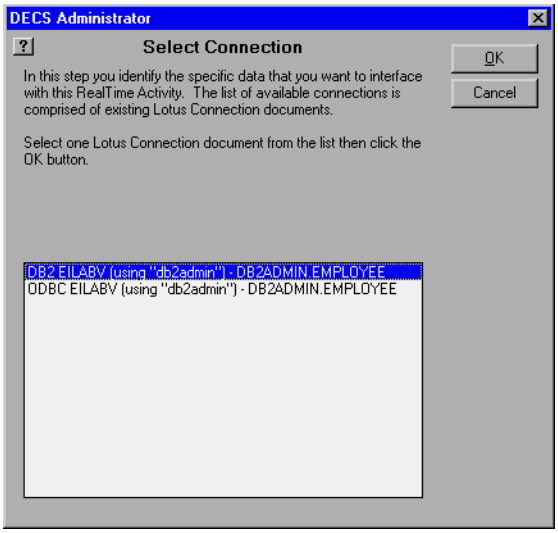
From this list, select the Domino database you want DECS to connect to. To follow the Millennium Cafe example, the line “DECS\cafeR5.nsf” will be selected. Clicking the ? gives you further information about this section of the wizard and it also informs you where to modify any wizard settings, should the computed settings be incorrect. Click OK when you have selected your database.

5. The wizard proceeds to query the Domino database you selected in the previous step. It will present you with a list of “metadata” that you can map to. At this level, the metadata that maps to the external data source table or view will be a Domino form, shown in the next figure.



Select the form you wish DECS to map to for this activity. Clicking the ? button will provide further information on how you can modify settings made by the wizard. Click OK when you have selected your form.

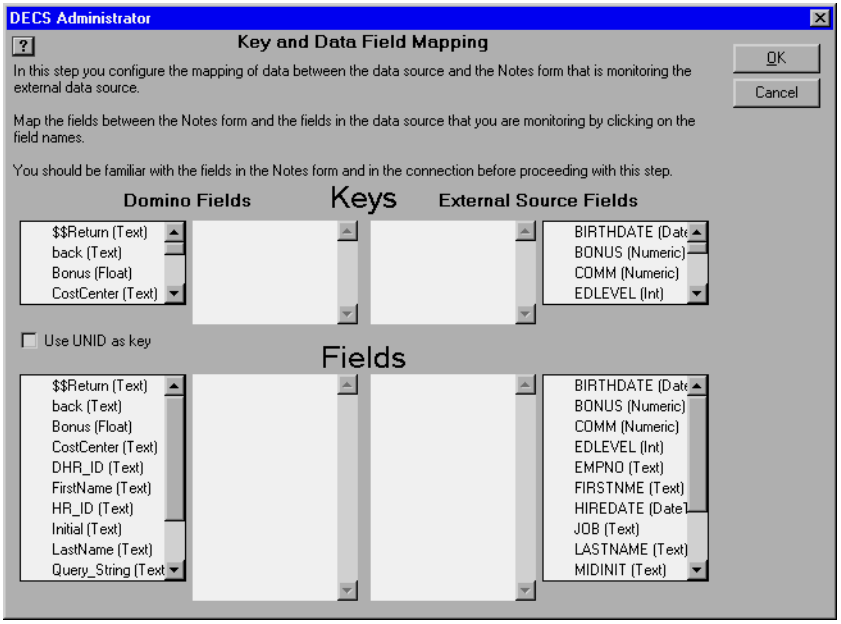
6. At this point the wizard has established the complete Domino connection; now it will ask you to indicate which external data source connection you wish to use.



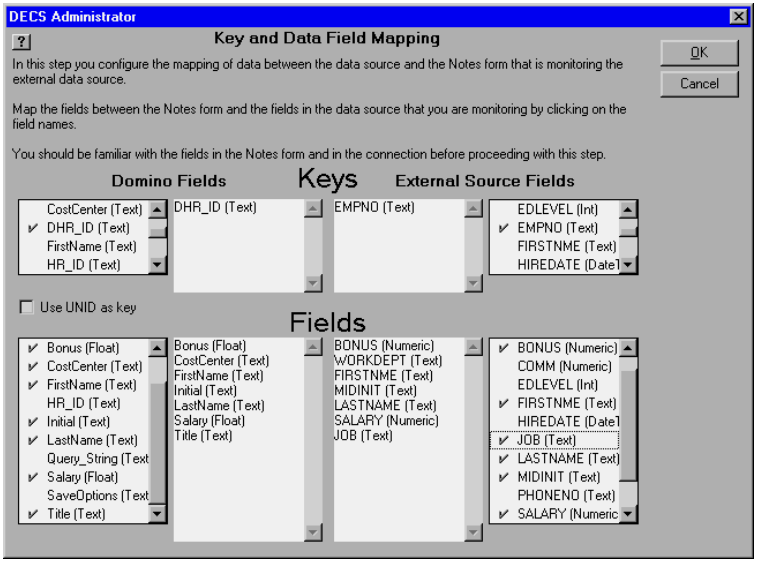
Select the external data source connection for this activity. For extra help, click the ? button. This button explains this part of the wizard and tells you how you can modify any settings that it makes. If the connector you

wish to use has not been created yet, click Cancel to leave this screen, then click New in the Lotus Connection section of the Activity form to create a new connection document. Click OK after you select your external connection to proceed to the next step.

7. The wizard proceeds to query both the Domino connection you supplied and the external data source connection to retrieve a list of fields from Domino and the list of columns from the external data source. The following dialog box is presented:

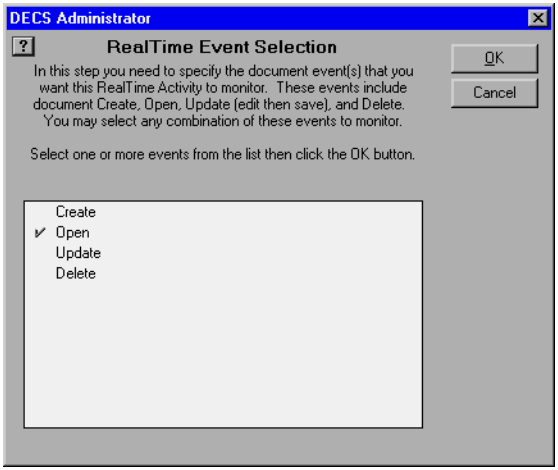


This dialog box shows the Domino fields on the left and the external data columns on the right. Clicking the ? button will provide you with help to perform the mapping. In general, you must select at least one key on either side, and you must map the same number of fields from Domino to the identical number of columns from the external data source. When the process is complete, your dialog box should appear like the following sample:



This dialog box is extremely helpful since it provides you with attribute information on the fields. To continue with the wizard, click OK.

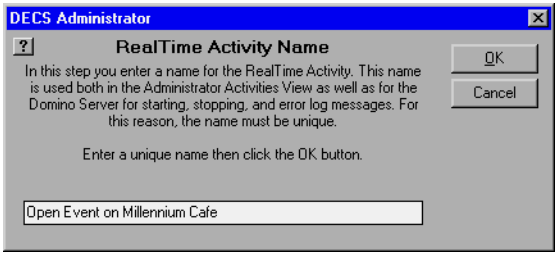
8. You have various degrees of control with the event dialog box.



Clicking the ? explains that certain events require other events to occur. In particular, the Update event must contain an Open event such that the data to be updated can be retrieved. However, if you select to store keys and fields, the Open is not required for an Update event. Refer to the "Activity Options" section in this chapter for further discussion of these topics.

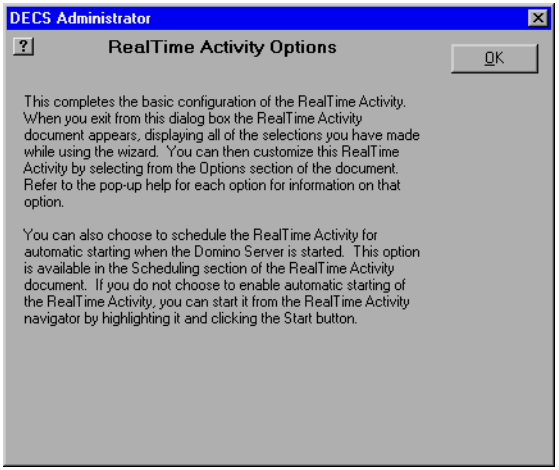
Select the events you wish to capture and then click OK to continue with the wizard.

9. The next dialog box presented by the wizard requests the name of the activity.



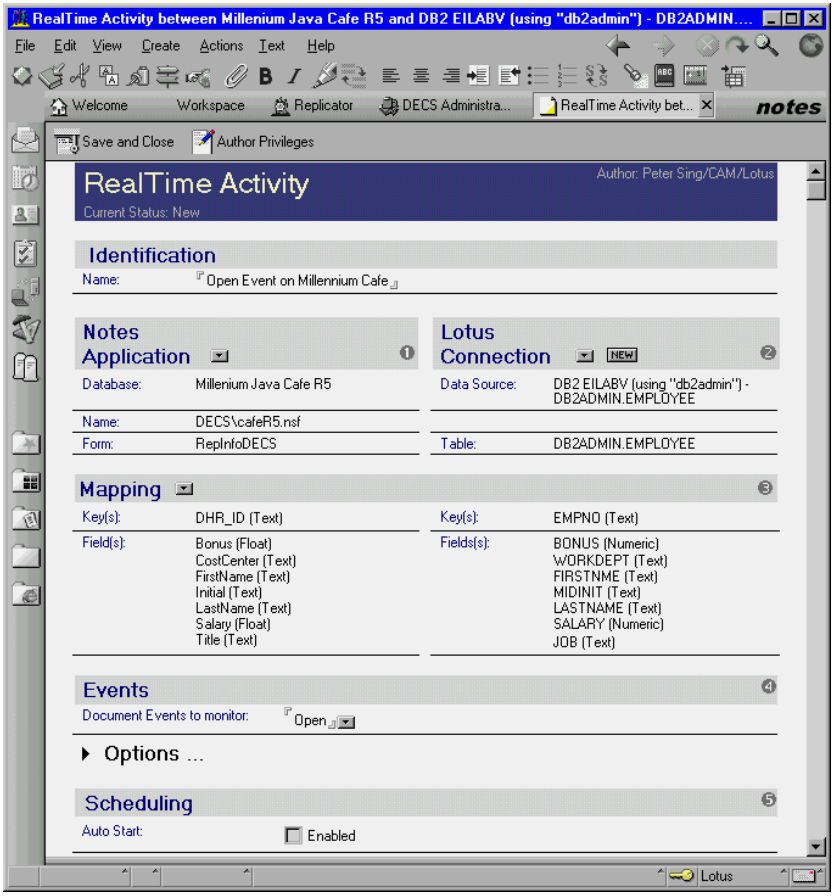
Enter a name that uniquely describes your activity document. Include items like the event types, the Domino database or form (or both), external connection type, and anything else that you feel will separate this activity from the rest. Click the ? button for further help when naming your activity. Click OK to continue.

10. The wizard presents the following dialog box indicating that it has completed its process.



The wizard reminds you that there are options that can be configured to uniquely control the activity. The default options are sufficient for the activity to run. Click OK to complete the wizard.

11. The wizard is now complete. You will be presented with the completed Activity document, shown in the following figure:



Click Save and Close in the top left corner to exit your activity document.

The next section will go through the same process without using the wizard. If you wish to configure your real-time Activity options, proceed to the Activity Options section.

Creating Activities without the wizard

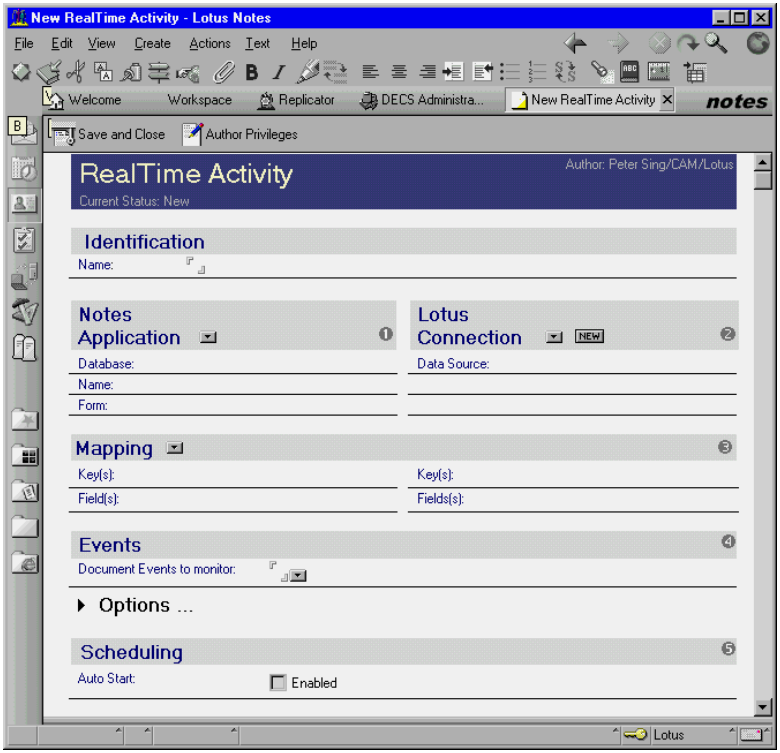
Once you are familiar with the activity creation process, turn the User Assistant off to disable the wizard. The following steps will guide you through the activity documents creation process without the use of the wizard.

1. From your Notes R5.0 client, open your Domino server's DECS Administration database.

2. With the User Assistant (wizard) turned off, click the Create Activity icon.

Note The “blue 2” will not appear when the wizard is off.

3. You will be presented with a new real-time Activity document form.

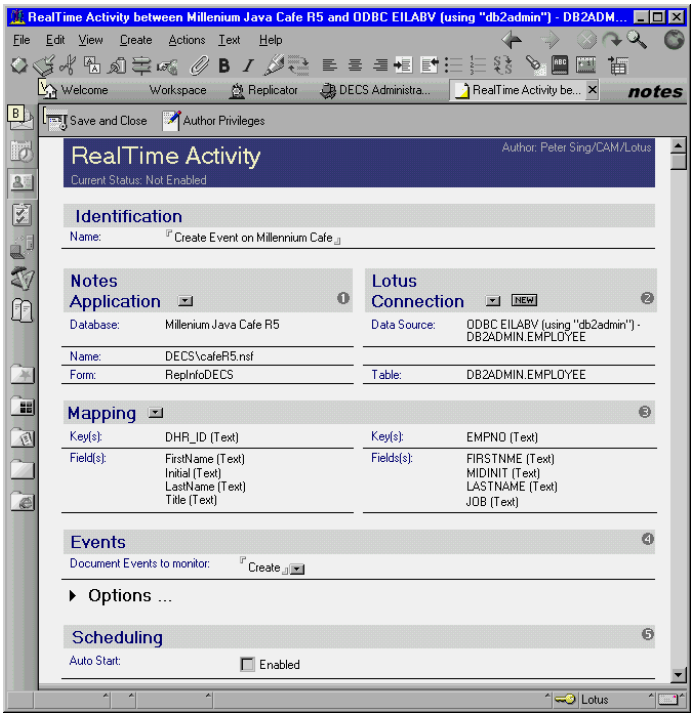


Within the Name field of the Identification section, type the name of your activity.

4. In the Notes Application section, click the drop-down button. This will invoke a component of the wizard that will ask you to select the Domino database with which you will be integrating DECS. See step 4 in the previous section for further detail.

5. After you select your database and click OK, the wizard remains invoked and displays the metadata dialog box showing you a list of forms within your selected database. See step 5 in the previous section for further detail. Click OK after selecting your form from the dialog box.

6. Click the drop-down button of the Lotus Connection section of your activity document. Once again, a component of the wizard is invoked that asks you to select the connection that this activity should use. See step 6 in the previous section. Click OK in the connection dialog box to continue. If the available connections are not suitable for your activity, click New in the Lotus Connection section to create a new connection.
7. Click the drop-down button in the Mapping section of your activity document. The Key and Data Field Mapping dialog box is displayed. Select your key mappings and field mappings as outlined in step 7 in the previous section. Click OK to continue.
8. Click the drop-down button in the Events section to select one or more document events from which to trigger. This field is a multi-value field allowing for more than one type of event to monitor.
9. The remaining sections in the activity document are discussed in the "Activity Options" section of this chapter. Your completed activity would appear similar to the following figure:



10. Click Save and Close to save your activity. It is now ready to use.

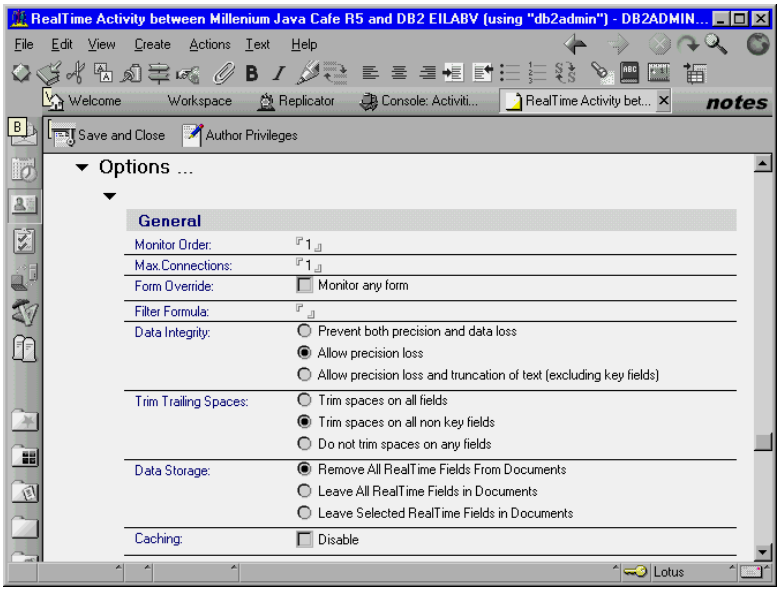
The Options section of an Activity document gives you further control of your activity at a general level and for each event type that you selected. The next section will highlight these options.

Activity Options

In this section, the Millennium Cafe activities have been modified to show the different options available for each event and the general options.

General Options

The General Options available are shown on the following screen.



Each general option is discussed in turn. Business scenarios for their use are explained as well.

General Option	*Description*	*Business Application*
Monitor Order	The Monitor Order option is used if you use more than one real-time Activity for a single Domino form. You may specify the order in which the Real time Activities will intercept the document's events.	Monitor Order acts like an SQL Join. The monitor order option should be used when the external data sources you are accessing are not from the same relational environment. If you have employee data on DB2 and work information on Oracle, you can specify order 1 for the DB2 activity. That activity will extract a department number which is used by the Oracle activity to extract department data using order 2. **Note** You must specify keeping selected real-time fields or all real-time fields so that follow-on activities can process.

continued

General Option	*Description*	*Business Application*
Maximum Connections	Real-time Activities can open more than one connection to the external data source. The higher the number you enter here, the better your performance will be when a very large number of users access documents at the same time. **Note** During the reading or editing of a document, a connection is idle and available to service other users. This means that two users can use the same connection provided either of them is idle at any one time. This is also described as pooled connections. Once a connection has been established it will remain active until its associated activity is stopped.	Any Domino database you have that has a high usage rate and is monitored by DECS would qualify to have its maximum connections option increased. This option would be useful in centralized applications where one Domino database using DECS to access external data would be used by many people. Information such as a company directory would fall into this category. We recommend using a value of 2 or 3 in this field to start. **Note** A high number of connections for any application may require a communication server installed with Domino instead of a communication client.
Form Override	Selecting this option causes the real-time Activity to process the selected events for all the documents in the Domino database, regardless of the form.	This option is useful for databases that contain forms that use a common set of fields or a common subset of fields, provided all the forms have the mapped key fields. This option is used for views that use form formulas or documents that change the form field. This option can also be used along with the Filter Formula option.

continued

General Option	*Description*	*Business Application*
Filter Formula	This option allows you to create a formula that defines the documents that the real-time Activity will monitor. For all events, you can only reference fields stored in the Domino documents.	Combining a filter formula with the form override allows you to restrict which forms should be processed by your activity. Click Form Override to on, and add a Filter Formula of Form = "Memo" | Form = "Reply" which forces DECS to process only the Memo and Reply forms. Another useful application for Filter Formulas is to control submission of data to the external system. If you want to only create or update external data that is flagged as public and not in process in your Domino database, your Filter Formula would be Status = "public."
Data Integrity: There are three levels of data integrity. The level affects what happens when data is sent between the Domino document and the external data source.	Prevent both precision and data loss: This option will write an error to the Log for any data loss resulting from a data type conversion. Allow precision loss: This option will not report loss of numerical or datetime precision as a result of a data type conversion. Allow precision loss and truncation of text (excluding key fields): In addition to precision loss, this option will truncate text data when necessary to conform to field lengths in the external data source. Key fields will not be truncated even with this option selected.	Any application of these settings must be set so that they conform with your integrity requirements. When you use the first option, you run the risk of creating many log entries for your activity. This may be an acceptable start for a new DECS process; however, the errors must be handled, rules must be established, and then you can change to the option to Allow precision loss. Tight control of integrity may require the use of a data dictionary to control Domino field lengths and dynamic validation formulas that use a dictionary to perform length checks. This degree of control would be necessary for Create and Update events. Should you miss any validations, you must decide if the Domino data can be truncated or if you will risk precision loss (option three versus option two).

continued

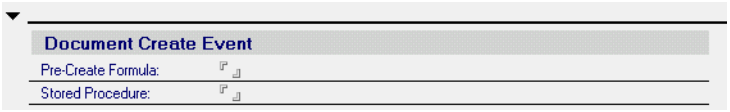
General Option	*Description*	*Business Application*
Trim Trailing Spaces	This option affects any trailing spaces that exist in the Text fields of the external data source. There are three choices: Trim spaces on all fields: Select this setting to trim trailing spaces from all text fields. Trim spaces on all non key fields: Select this setting to trim trailing spaces only from data fields, not from the key fields. Do not trim spaces on any fields: Select this option to leave trailing spaces in all fields.	External data sources will pad blanks or other unreadable data at the end of their fixed length text fields. When these fields are read by DECS, the unreadable data is converted to spaces and placed in Domino. This will cause odd presentation of retrieved data. Domino data will wrap to the next line in a form. Domino views that categorize on retrieved data will react oddly to column formulas. If you match any data between Domino and the external data source, do not trim the spaces. If you find matching is still not correct, consider changing the fixed text format in your external data source to variable length (that is, from CHAR to VARCHAR). **Note** Trimming only occurs when reading data from the external source.
Caching	This option is used to disable caching in the HTTP server for monitored documents. When a document is retrieved, the HTTP server in Domino may cache it to avoid disk access for the next retrieval. For rarely changing external data source records, caching may be fine. For a Real-Time situation with changing data, caching should be disabled.	An unchecked caching option will enable it. Use this option for reference material available over the Internet. Employee handbook information, company policies and product White Papers are just a few applications where caching should be enabled. Order entry or order status systems are more dynamic and would require caching to be disabled. **Note** Keep in mind that this is a Disabling option, not an Enabling option. This option is only effective in Domino's HTTP server.

continued

General Option	*Description*	*Business Application*
Data Storage	Remove All Real time Fields from Documents: Use this option if you want to remove all the data fields mapped in the Activity from the Domino document before it is saved to disk. This is the default. Leave All Real time Fields in Documents: Use this option if you want to leave all the data fields in the Domino document, rather than removing them after updating the external source (see above). This option only takes effect when creating or updating a document when the Activity is active. Opening the document and making no changes does not trigger this option. Leave Selected Real time Fields in Documents: Use this option if you want to leave selected data fields in the Domino document. The Save Fields button that appears when you enable this option lists the Domino data fields from which you can select.	Selecting either of these options may not be a preference for you to decide. Look at your integration requirements to determine if any of these options must be set. When using the Open event only, the default option is enough. You may feel that keeping all the fields on the form will improve performance. It is not so. The Open event will always retrieve data, whether you store the data or not. In some cases, the default performs better. The first option is best for reference- or inquiry-based Domino databases that do not update the external source. Standard examples like a company directory are quite possible, provided the Domino database is not used for input. The next two options have more unique uses. If you will be using the Update event, you must store any fields identified as key fields in other activities (see Monitor Order). This means that the key field is always stored no matter what you select here. If you use the Open event and the Monitor Order option, fields required by the follow-on activities must also be stored. Using the company directory as an example, assume your Domino database has a directory view containing some of the data retrieved from the external source. Those fields referenced in the view must be stored in your form. **Note** For option three, you must select at least one data field; otherwise the default option applies.

Create Event Options

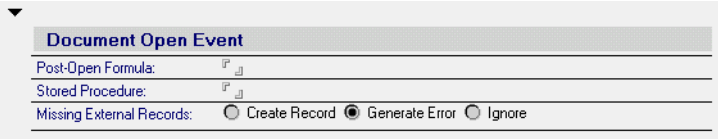
The Create Event Options available are shown in the next figure, and discussed in the following table. Applicable business scenarios are included along with the descriptions to help you use these options.



Create Event Option	*Description*	*Business Application*
Pre-Create Formula	This option runs a Domino formula before the document data is stored in the external data source. This provides an opportunity to compute additional fields or modify existing fields.	You can build a formula using the Domino formula language. Use this to manage fixed lengths, apply default values to blank fields, or compute hidden fields that are mapped. If you have a Call Center application in Domino that stores call information in your external system, you can use a formula to set a random call identifier: FIELD Call_ID := @Unique In this example, Call_ID must exist as a computed field and must be mapped to the external data source.
Stored Procedure	This option runs a stored procedure in the external data source system to store data that has been entered in the document.	Your current company directory on your external system uses defined procedures to manage input. These procedures may be defined to distribute results to multiple external tables. Now that you have moved to Domino, you want to integrate the fastest way possible. Enter the name of your stored procedure in this field and press F9. DECS will retrieve the input parameter requirements for that procedure and present them to you. For each parameter, select the Domino field that corresponds to the input parameter and then let DECS integrate with your stored procedures rather than your external tables.

Open Event Options

The Open Event Options available are shown in the next figure.



In this section, each open event option is discussed and applicable business scenarios are listed to help you use these options.

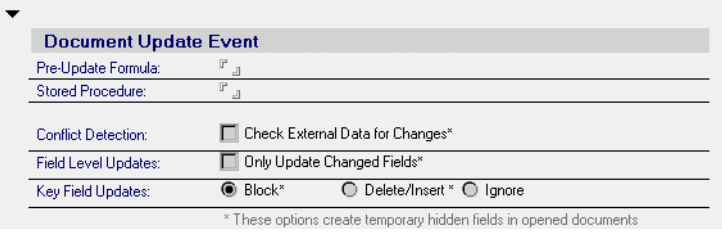
Open Event Option	*Description*	*Business Application*
Post-Open Formula	This option runs a Domino formula after the document data is retrieved from the external data source.	You can use this formula to change retrieved data that can be expanded or translated to take advantage of the Domino interface. You can modify names, coded fields (such as 1 for yes or 0 for no). Note: Do not use a Post-Open formula together with the Conflict Detection item in the Update Event Option as this would continually flag your document as a conflict and never update the external source.
Stored Procedure	This option runs a stored procedure in the external source to determine the data that will be retrieved into the document. The Real time keys are supplied to the stored procedure as input parameters. The stored procedure must produce a result set with both the keys and fields present.	This option can be used for the same reason you would use it in the Create Event options, except in the reverse direction. With your company directory on your external system, Domino has an application that is used for viewing this information. The external system used a stored procedure to do this, now you can use Domino instead. Enter the name of your stored procedure in this field and press F9. DECS will retrieve the input parameter requirements for that procedure and present them to you. For each parameter, select the field that corresponds to the input parameter and then let DECS integrate with your stored procedures rather than your external tables.

continued

Open Event Option	*Description*	*Business Application*
Missing External Records	If no matching record is found in the external data source when opening a document, there is the option to create a new record in the external system, generate an error which will appear as an error box for Notes clients, or ignore the error and allow the document to continue opening but without any external data.	You may find this option an important one to deal with. Issues like this involve ownership and direction of data integration (that is, is the data moving both ways or one way — from Domino or to Domino). The default option to generate an error will notify the client that the external data record is not found. For environments where the external source is the owner or master of the data, use Generate Error or Ignore. If Domino is the Master, use Create Record. If your application, the company directory, allows data entry from both sides, you will have to decide the best option. Note: Deletions from the external source will not be noticed by DECS, therefore making the Create Record choice is dangerous. If the external source is the master — do not use Create Record.

Update Event Options

The Update Event Options available are shown in the following figure.



In this section, each update event option is discussed and applicable business scenarios are listed to help you use these options.

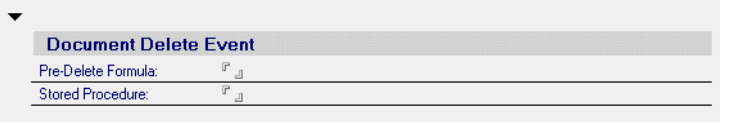
Update Event Option	*Description*	*Business Application*
Pre-Update Formula	When a document is updated (edited then saved) this option runs a Domino formula before the document data is saved in the external source. This gives you the chance to change field values.	This is very similar to the Pre-Create formula and usually they are the same formula. Typically you will want to create or update records in the external source using the same rules. **Note** Be aware of your Conflict Detection option when using a Pre-Update formula. Do not let your choices conflict. The Pre-Update formula is run before conflict detection is determined.
Stored Procedure	When a document is updated, you have the option to run a stored procedure in the external source. This will store the data that has been changed in the document. The Real time keys and fields are supplied to the stored procedure as input parameters.	See the Create Event Option Stored Procedure option. Both the create and update options may have similar stored procedures. If these stored procedures are different, this is a decision made at the external source. Update procedures may generate reports that create statistics unique to updates. Follow the guidelines put forth by your external system.
Conflict Detection	This option ensures that the external data has not changed since the document was opened. If it has changed, the update to the external source will fail.	This option is very useful when many people have access to your Domino database. Team databases, where any one external record can be updated by the team, can make good use of this option. **Note** See the Post-Open formula option for conflict detection restrictions. **Note** Each time you edit and save the Domino document, an update event occurs. With conflict detection on, you cannot edit and save in more than one edit session. Exit the document and reopen each time you want to update data.

continued

Update Event Option	*Description*	*Business Application*
Field Level Updates	This option causes the real-time Activity to not update fields in the external data source unless the same fields in the Domino document have been edited.	External applications that monitor data activity may require the setting of a field to indicate a change has occurred. An interesting example of this is to update a trigger flag in the external source each time the Domino document changes. Have Lotus Enterprise Integrator (LEI) poll the trigger and then use an LEI direct transfer or replication activity to move the change to another external data source. The trigger in Domino can be set using the Pre-Update formula or a QuerySave event.
Key Field Updates	Block: Do not allow updates to key fields in the Domino document or the external data records. Delete/Insert: Updates to key fields will cause the original record in the external source to be deleted and a new record with the new key added. Ignore: Do not update the key fields in the external source. If a key field is edited, the change will be stored in Domino but the key fields in the external source will not change.	Block is the default in this situation and is the better choice. For a Real-Time activity that is only updating, Block should be used. Your external systems may control key creation and deletion and therefore should have their activity simplified by using this. Delete/Insert is the opposite. It allows all key changes to go through. The Ignore option is not common but available. For situations where you want to deposit data but are not aware of the key, DECS will move your update, but its reference will be gone after the key change in Domino. This may be sufficient for survey responses and any application where Domino "Depositor" access is used. Note: DECS uses the Connection ID that you specified in the connection document. This ID has full access to the external source and associated user account privileges defined. To limit who makes updates or changes in general, consider using options like this in combination with strict Access Control List (ACL) settings, document security and roles.

Delete Event Options

The Delete Event Options available are shown in the next figure.



In this section, each delete event option is discussed and applicable business scenarios are listed to help you use these options.

Delete Event Options	*Description*	*Business Application*
Pre-Delete Formula	When a document is deleted, this option runs a Domino formula before the document data is removed from the external source. This provides you with a chance to create or modify fields. Note: The external account ID associated with the connection document must have deletion rights to delete external data.	When you delete an external record, you may want to log the occurrence. You can create a Pre-Delete formula that mails document information to a log database for audit purposes. Note: You should indicate that the fields used in the pre-delete formula be saved on the document, otherwise they will be blank.
Stored Procedure	When a document is deleted, this option executes a stored procedure in the external source to remove the data related to the document. The Real time keys are supplied to the stored procedure as input parameters.	External systems that have stored procedures for deleting data, can be used by DECS. This allows you to easily integrate Domino into your existing processes. If you have a stored procedure that logs deletions on the external system, enter the procedure name, press F9 and you will be presented with the available input parameters.

Logging of Activities

All DECS activities are logged in the Domino server log. You can check the Miscellaneous Events view to see these entries. You can also click the View Log button at the top right portion of your activity document or you can click the Log navigator icon in the Activities view. Both of these will retrieve log entries for your activity since it last started.

Scheduling

For Real time Activities, your scheduling options are minimized as compared to NotesPump and Lotus Enterprise Integrator. Since Real time Activities always run, they do not need a schedule. However, in the event

your system running Domino comes down, during restart you will want your business-critical activities to start automatically. To do this, check the Enabled box in the Scheduling section under Auto Start.

DECS and the Domino Architecture

DECS is included in Domino R4.6.3 and later. Integrating DECS into Domino provides you with the ability to use some of the services in Domino Enterprise Server with DECS. This section discusses DECS together with:

- Clustering
- Partitioning
- HTTP Server Task
- LEI Real time

Clustering

Clustering Domino allows you to maintain high availability of your Domino server. Clustering makes use of the Domino replication technology to ensure one Domino server has all of its Domino functions duplicated on another box. If the first Domino server comes down, the duplicate server will start to run the processes from where the last server left off (to the last point of replication).

DECS supports this process, but with the following limitations:

- Each clustered Domino server requires the communication software needed to connect to your external data source.
- All of your communication configurations must be duplicated. Clustering only duplicates the Domino part of the server. Ensure your communication setup is also duplicated (this includes database alias names).
- Your communication software must allow shared access for the same ID.
- You must use the Conflict Detection option for Update Events.

  DECS automatically detects updates made from Domino Server A to External Connector Source C. As the operation is propagated to Domino Server B, DECS ensures the External Connector Source C does not receive a duplicate update. Under the same circumstance, where a clustered server requested an external update to External Server D from the same DECS enabled form, DECS detects this condition and propagates the update accurately to External Server D.

Partitioning

Creating a partitioned Domino server allows one box to run several Domino Servers. The number of partitions is limited by the platform your Domino server is running on. DECS will only work on one partition. It can be available on multiple partitions but only one DECS task can run at any one time on the computer. That is, if your AIX Server had six Domino partitions running R5.0, each partition would have DECS available to it, but only one of the partitions could actually have DECS loaded at one time.

Note The one partition running DECS must include the extension manager setting in the NOTES.INI file. No other .INI files can have this setting. To allow different partitions to run DECS, the extension manager setting must be invoked for the currently active DECS task and removed for the previously running DECS task.

HTTP Server

The Hypertext Transfer Protocol (HTTP) server task is part of the basic Domino installation. It is mentioned in this section to highlight an important point. All Web-supported Domino functions are available to DECS. The one Web-only related option in DECS is the caching function. See the general options sections for more detail. The DECS Examples sections provide more detail on its function over the Web.

DECS and LEI Real time

As of Domino R5.0, LEI has one solid advantage over DECS, that is, LEI can use metaconnectors in its Real time activity. In a coming release of Domino DECS will support metaconnectors. When this occurs, you have to decide between the two. They are both effective, and each has its own advantages, but there is no simple recommendation as to which way to go. This decision requires investigation regarding the proposed installation.

Some points to consider:

- LEI Real time can handle higher data volumes and can be run as a separate operating system task, allowing it to utilize operating system resources.
- DECS supports clustering and does not require a separate installation for each clustered machine since it is included with Domino.
- DECS does Real time only. If that is all you need, use DECS.
- LEI provides full migration from NotesPump 2.5a Real time to LEI Real time. If your previous Real time setup is large or complex enough to warrant a simple migration path — use LEI.
- DECS Administration is simpler. It appeals to the first-time enterprise integration users.

- DECS uses Domino. A highly active Domino server busy with HTTP, Agent Manager and Database serving tasks may not be ideal for a DECS solution. Gauge your Domino activity and consider distributing the server load to another Domino server or allowing a separate LEI task to work outside of Domino, so as to limit the Domino activity.
- The more active Domino is, the more a powerful machine is recommended. If your infrastructure does not support high-powered hardware, a distributed solution using several Domino servers may be required. You may find it necessary to isolate all of your DECS activity on one server; however, DECS is not recommended for high volume real-time integration. When your volumes increase, switch to LEI.

Weigh the above points carefully. Integration volumes, Domino activity and your infrastructure will suggest whether to use DECS or LEI.

DECS Examples

The Millennium Cafe for Domino R5.0 is used for this section to show you how DECS works in Domino. Client sessions from Notes and a browser are displayed.

Dynamic Queries

DECS functions very well for databases and external data sources whose key values for each document or record already exist. How do you pull new information from the external source without using the Initialize Keys action? What do you do if you know the key, but it is not on the Domino database? These questions are discussed in this section as the Millennium Cafe is used to show them working. The dynamic querying ability is useful when storage of large numbers of keys (stub documents) in Domino is not appropriate, or not required. In general, the dynamic query function will compose a new document. This document will ask you to input the key value and press a button. After pressing the button, the document will be saved and reopened. This reopening will cause DECS to trigger and it will retrieve the data based on the key you typed in.

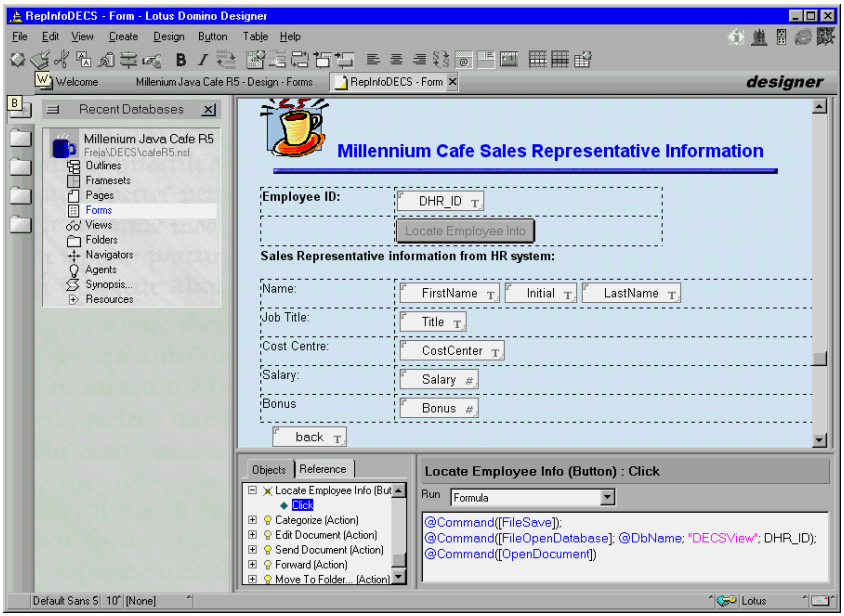
Notes Dynamic Query

We will show how the standard DECS example in Millenium Cafe can be modified to allow dynamic query from a Notes client.

Using the connection and activity definitions discussed in the earlier sections, the activity was modified so that it triggers on Open events only. The dynamic query method works very well with Open events only. Adding an Update trigger to this activity will create a DECS error because an attempt is being made to create a duplicate record (we type in a key that

already exists externally). For dynamic query to work from Notes, perform the following steps:

1. Open your Domino database in Domino Designer. In this example, we open the Millennium Cafe in Designer.
2. Add a Hotspot-Button below the input field for the key as shown in the following figure.



In this example, the Locate Employee Info button was created within a table cell just below the key field "DHR_ID." The button formula is shown in the Programmer's Pane.

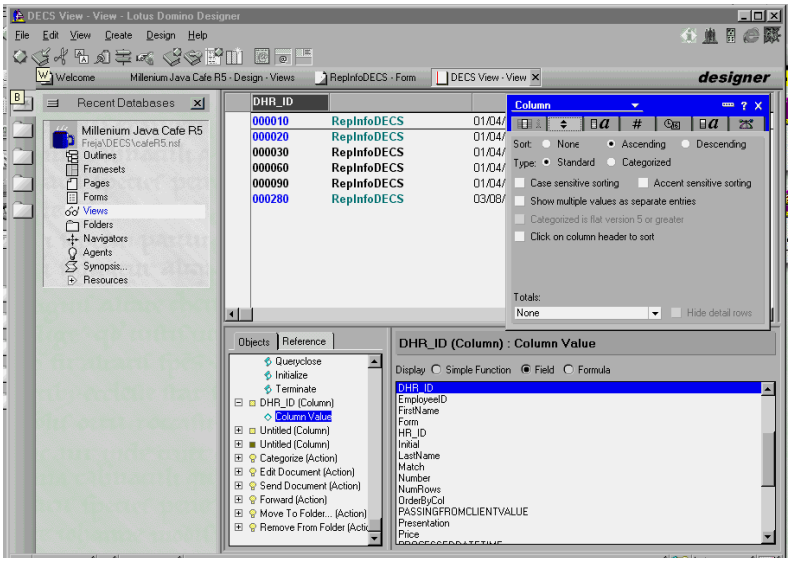
```
@Command([FileSave]);

@Command([FileOpenDatabase]; @DbName;"DecsView";DHR_ID);

@Command([OpenDocument])
```

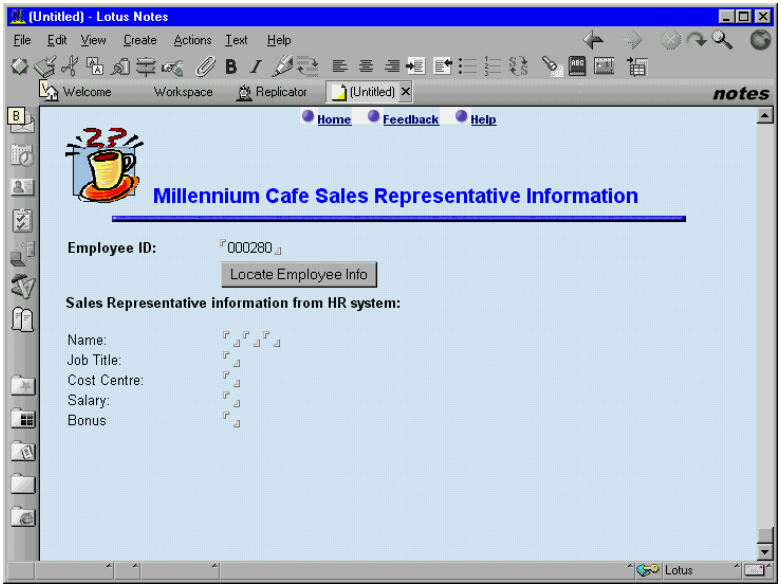
This formula will save the document, open the same database using a particular view and key, and then open the document with the key value to trigger the open event.

3. Save your form changes.
4. From the Designer screen, click the Views icon within the Designer Pane.
5. Highlight the view you want to use for your formula (ensure the view name or view alias is entered in your button formula). If you don't have a view, create one that is sorted by the key field in the first column. Millennium Cafe uses the DECS View, shown in the following figure.



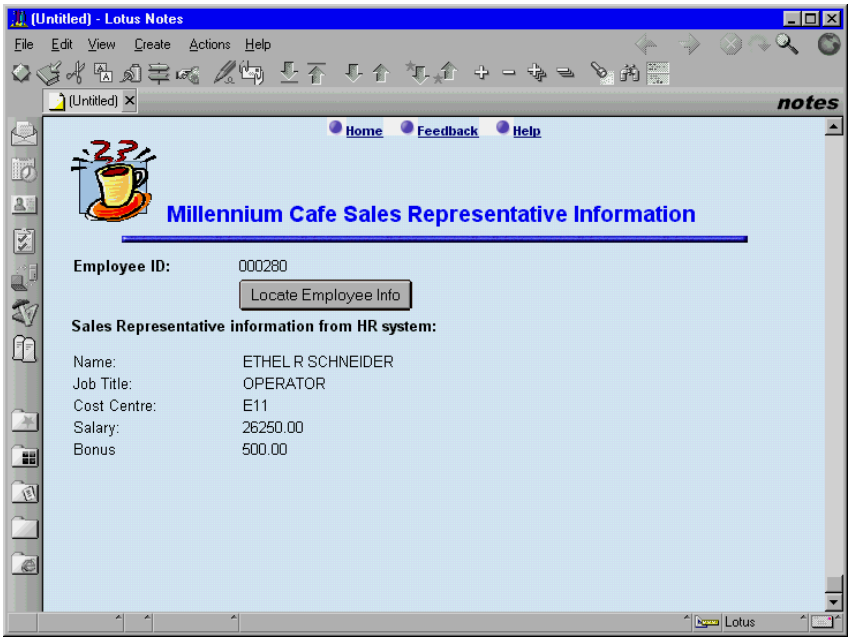
The alias for this view is DECSView. This value was used in the button formula on the form.

6. Save your form and view.
7. Start your Real time activity in the DECS Administration database.
8. Compose a new document in your database and enter the key field value.



The key value 000280 was entered in this example.

9. Click the button you created (the Locate Employee Info button in our example). This action will save your document, reopen the database to your sorted view and open the document with the key value you entered. The opening event will cause DECS to retrieve your data. The following screen shows the result from DECS.

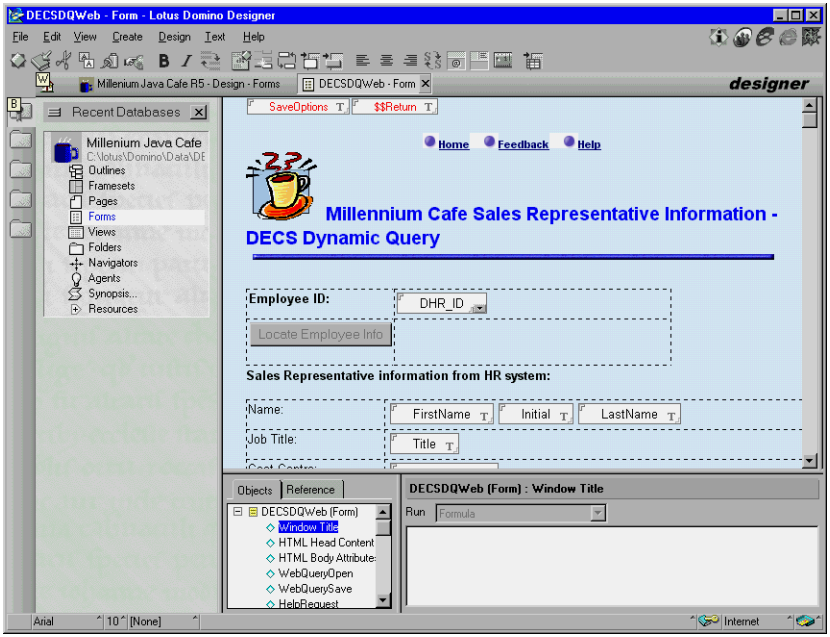


10. You can now go back to your compose window and enter another key field value since the window was never closed.

Web Dynamic Query

Millennium Cafe will be used to show the dynamic query function from the Web. This example uses the form DECSDQWeb and view DECSViewDynamicQuery. The following are required for DECS dynamic query over the Web:

1. Open your database in Domino Designer. In our example, we open the Millennium Cafe.
2. Create a form that defines the format of the data to be returned from DECS. Millennium Cafe uses the DECSDQWeb form shown in the next figure.



This form uses several design elements to allow dynamic Web querying to work. They are:

- $$Return: This field contains the formula used by Domino that executes when a submission occurs. The submission, in this case, will be the Locate Employee Info button. The $$Return field is programmed as follows:

  ```
  dbname := "/" + @Subset(@DbName; -1 );

  "[" + dbname + "/DECSVIEWDynamicQuery/" + DHR_ID +
  "?OpenDocument" + "]"
  ```

  The above formula specifies the URL that the DECSDQWeb form is to return to when the Locate Employee Info button is pressed. This URL tells Domino to open the document keyed with the value in the DHR_ID field and found in the DECSVIEWDynamicQuery view. This view must be sorted by the first column, which must be the key field.

- DHR_ID: This field has a drop-down box that allows you to pick from a list of key values. Millennium Cafe will show a list of keys from the Employee table using the following formula:

  ```
  @DbColumn("ODBC":"NoCache";"EILABV";"db2admin";"password"
  ;"DB2ADMIN.EMPLOYEE";"EMPNO")
  ```

The Domino server running Millennium Cafe has defined the Employee DB2 table in the ODBC Device Manager. The @DBColumn command will return a list containing all of the EMPNO entries in the Employee table. If you refer back to our DECS Real time activity, EMPNO maps to DHR_ID.

- Locate Employee Info: This button contains no code. When pressed from the Web browser, it will perform the submit action (a save). After the submit is performed, the $$Return formula is run.

3. Create a home page that will give you an access point for composing your Web form. Millennium Cafe uses the About document to perform this. The About document should contain an action hotspot using a form compose formula. Millennium Cafe uses the following formula:

   ```
   @Command([Compose];"DECSDQWeb")
   ```

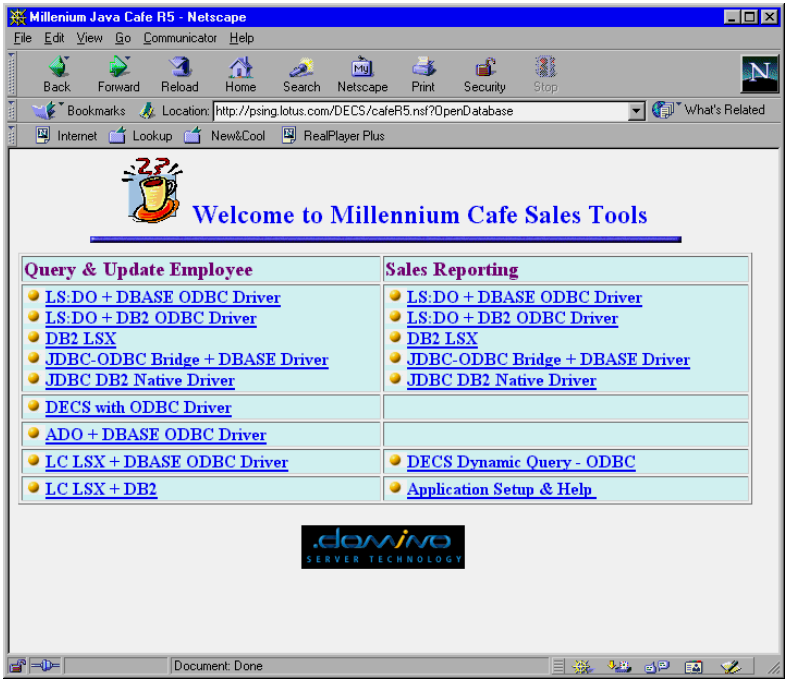
   Set the database Web launch properties such that the About document is always shown.

4. Define a DECS Real time activity to monitor your form. For Millennium Cafe, the existing Real time activity was modified such that the Form Override option was set to Monitor any form and the following Filter Formula is used to only monitor two forms:

   ```
   (Form = "RepInfoDECS") | (Form = "DECSDQWeb")
   ```

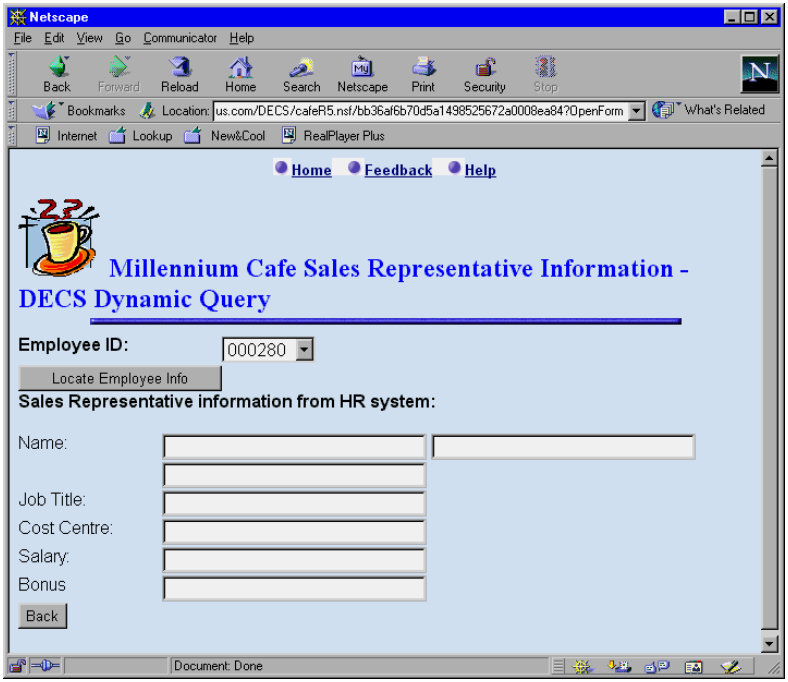
   Once you have completed the changes to the Real time activity, save it, close it, and then start it (you will have to stop the activity in order edit it).

5. Start your browser and access your server. The default access should let you see a database listing. Click on your database to see the About document that you created. Millennium Cafe launches with the About document shown in the following figure.



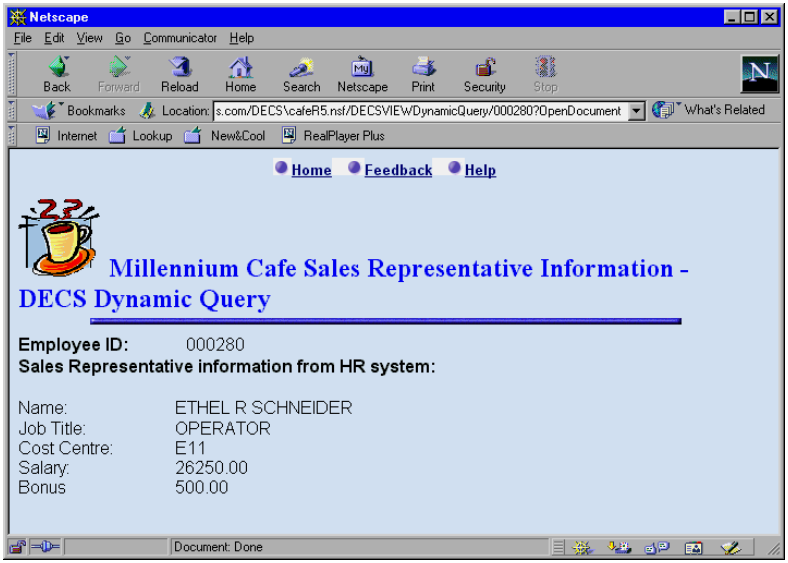
In this screen, the link titled DECS Dynamic Query — ODBC is used for the Millennium Cafe example. This example uses a combination of ODBC and DB2. ODBC is used to generate the DHR_ID field's drop-down list and DB2 is used by DECS to access the external source.

6. Click the link you created to access your form. For Millennium Cafe, clicking the link DECS Dynamic Query — ODBC will produce the following screen.



As with the Notes dynamic query example, employee 000280 is selected from the drop-down list.

7. Click the button you created that will save and re-link to your document. This is the Locate Employee Info button in the above example. At this point, Domino will save your document. As it closes your document the $$Return formula is executed which allows Domino to link to another page. In this case, the same document is requested to be reopened. Millennium Cafe produces the following screen.



As this document is opened, the DECS Real time activity is activated and data for Employee ID 000280 is retrieved from Millennium's EMPLOYEE DB2 table.

Note If you use the WebQueryOpen event for a Web form that is monitored by DECS, this event must be successful for the DECS extension manager to be triggered. If your WebOpenQuery code fails, the DECS Real time activity will not activate. If you must have a WebQueryOpen event, you should consider using the Lotus Domino Connector LSX (LCLSX) instead of DECS.

Running DECS

This section uses Millennium Cafe to show a typical approach to event monitoring. Open events will retrieve data for existing records. The Open event is useful for dynamic queries and for reference databases whose external source is constantly changing information, but not changing keys.

The rest of the monitored events (Create, Update, Delete) are useful for keeping the data between Domino and the external source the same.

Keep in mind, DECS is well suited for managing data with an external data source provided Domino is the source of the data. When Domino is not the source, you must consider ways of informing Domino that external changes have occurred without its knowledge. We will not go into a lot of detail, but briefly mention the following means for synchronizing your data:

- Use a Replication activity from LEI to keep Domino and the external source synchronized.

- Periodically reload your Domino database with the new keys. Customizing the Initialize Keys action into a scheduled agent may be an option.
- Modify the business methods such that Domino is the only source.
- Use LEI Real time in conjunction with Polling Activities and Direct Transfer or Replication Activities so that only one product manages your data.

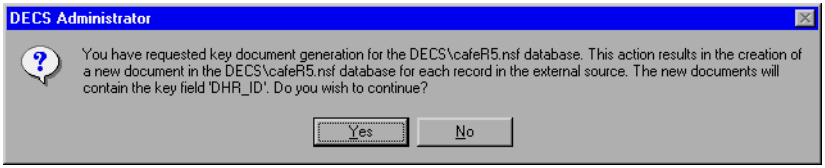
None of these synchronization methods will be shown here. If you need to synchronize your data, your business requirements (in particular, rate of data change on the external system) will determine which method to use.

In the following example, we will initialize Millennium Cafe and Open one of the external records based on the list of returned keys. We will then Create a new document and when we save it, DECS will create the external record. The document will be reopened to trigger the open event, then updated and saved to trigger the update event. Conflict Detection will be shown by re-updating the document and issuing a second save before closing it. Finally, the document will be deleted, triggering the Delete event to remove the external record. The deletion will be verified by making a dynamic query against the key for the document we just deleted.

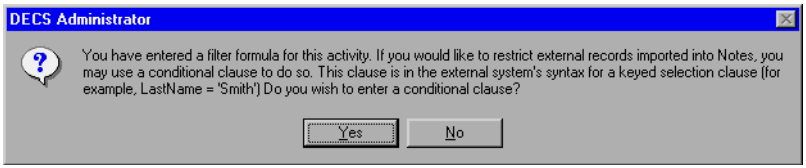
Example Setup and Initialization

1. Start your R5.0 Notes client and access your DECS Administration database.
2. Stop the Real time activity you used for the dynamic query example.
3. Edit your Real time activity and select that it monitor all events (Create, Open, Update and Delete). Save and Close your activity. Do not start it yet.
4. Open the Domino database that you created for your dynamic query example in Notes. For Millennium Cafe, the RepInfoDECS form will be changed such that the Locate Employee Info button is hidden when the document is in read only mode.
5. Save the form and close your database in Domino Designer.
6. Open your database and delete any data documents that you have from previous examples. This will prevent any duplication during the Initialize Keys run. Make sure that the Real time activity is stopped. If the activity is running you will delete the records in the external system as well. Close your database when you have deleted everything.
7. Back in your DECS Administration database, select your Real time activity document and then select Action from the menu bar. In the drop-down menu, select Initialize Keys to begin the key field population process. A sequence of message boxes will appear as the initialization

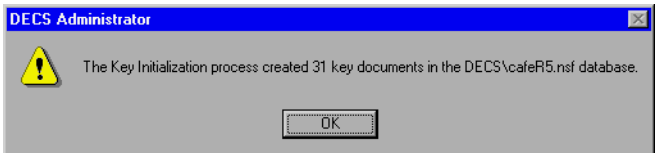
process continues.



This dialog box is presented as the Initialize Keys action is selected. Click Yes to allow the process to continue.



This message box may not appear when you initialize your keys. For the Millennium Cafe example, it does appear due to the filter formula that was entered. DECS will give you the opportunity to filter certain external records if it sees that you are already filtering the events in general. If you wish to enter a conditional formula for the initialization process, click Yes. For this example, No is clicked and the process continues.



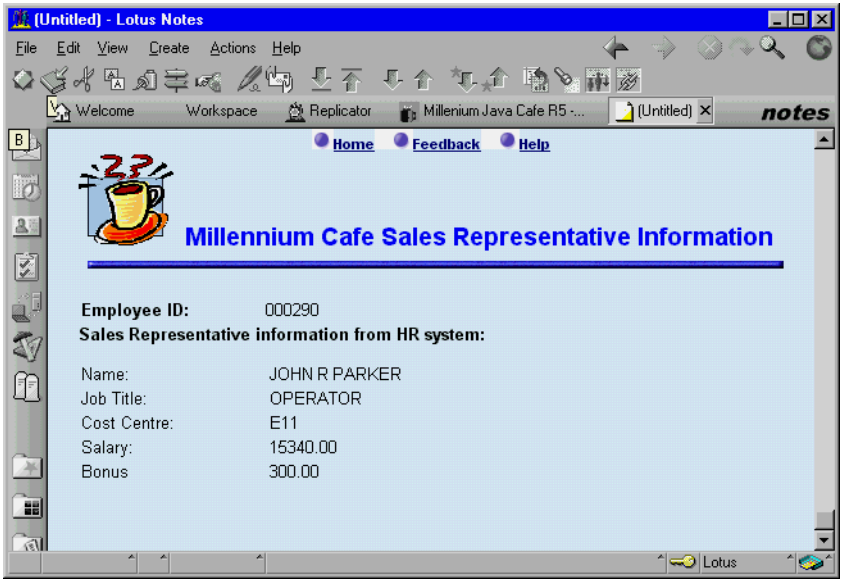
This message box is shown after the initialization process completes.

Note Millennium Cafe uses a filter formula that allows shared use of the DECS Real time activity for both the Notes form and the Web form. When initialization of keys occurs, the form name that is present in the Notes Application section of the activity (RepInfoDECS) will be stored in the Form field for the initialized stub documents.

8. With initialization complete, from your DECS Administration database in the Activities view, click the green start button to start your activity.

Verify the Setup

1. Open your application to a view where you see all of your initialized stub documents.
2. Open one of the documents. Document key 000290 is selected within Millennium Cafe for this example. The DECS activity will retrieve the data from the external source and present it as shown in the following figure.



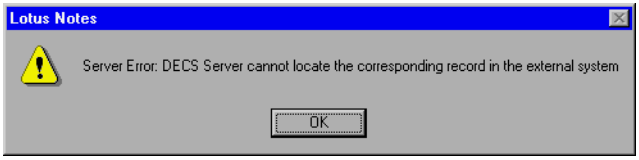
This example verifies that the stub documents were created (as shown in your view) and that the Open event is working against one of those documents.

Create an External Record and Verify

1. Close your window from the above example. You should be back in your stub document view.
2. From the menu bar click Create, and from the drop-down list click the form type for the document you wish to create. The Millennium Cafe example will create a document using the form RepInfoDECS.
3. The Millennium Cafe screen that is shown is the same as the entry form used during the dynamic query example. This example will create record 000280.

   First, we want to show that this record does not already exist. This quick little test requires the DECS Real time activity to be reset as an Open trigger only. If you do this, remember to set it back.

   To show record 000280 is not there, click Locate Employee Info. The following dialog box will appear.



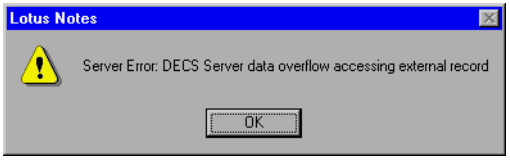
This message will be shown if the record you wish to locate on the external system does not exist. The Millennium Cafe Sales Employee 000280 does not exist. Click OK on the above message box to clear the message. Remember to change your DECS Real-Time activity to act on Create events as well.

4. Back in your Create form or RepInfoDECS for Millennium Cafe, enter the information you want to include for creation to the external system. Enter the following data for employee 000280:

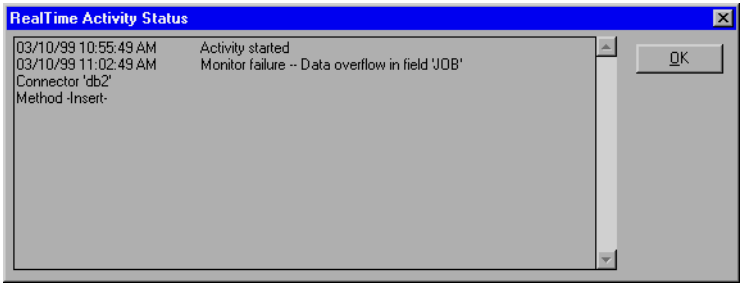
Field	*Data*
Name	Ethel R Schneider
Job Title	Senior Operator (Ethel gets a promotion!)
Cost Center	E11
Salary	30000 (Ethel gets a raise too!)
Bonus	550

Select File-Close from the menu bar (or you can press the Esc key, or press Ctrl + W). Indicate that you wish to save the new document by clicking Yes.

5. As with any true business situation, controlling errors is important. The previous step purposely allowed for an error to be exposed, which produces the following error message.



This message indicates that an overflow occurred, but where? Open the DECS Activity in the DECS Administration database, and click the View Log button. A dialog box shows more information about the actual problem, as shown in the next figure.



This dialog box will also be displayed if you select the Log icon on the DECS Navigator in the Activities view.

Earlier in this chapter, we said that field lengths are fixed in most external systems and the requirement to validate these lengths is recommended in Domino. The content of the field Job Title is too long, and Domino does not check for it. You can edit your application design to check for the length of this field, or you can define a Pre-Create formula that will strip the length of this field. DB2 only allows eight characters for this field; therefore Ethel's job title will be changed to Sr. Oper. Entering the new value and invoking the Save and Close sequence will result in a new external record being created.

6. After the external create DECS event runs, you will return to your database view, or the DECS View in Millennium Cafe. You should see an entry for document key 000280. Double-click this entry to open the document. The open trigger on your DECS Real time activity will run and retrieve the external data into the Domino document for 000280. To prove to yourself that DECS is retrieving data, from the view with the 000280 document selected, right-click the document and select Document Properties.... Under the fields tab (right-angle triangle in R5.0), you will notice that none of the mapped data fields are stored except for the key field.

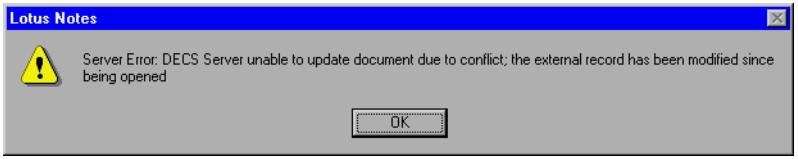
Update and Verify

1. From your document view in Millennium Cafe, reopen record 000280 by double-clicking it.
2. The document will open, triggering DECS to retrieve the mapped data fields.
3. Double-click anywhere on the form or press Ctrl+E to put the document into edit mode.
4. Edit any field you wish. For this example, the cost center will be changed to S12. Save and Close the document, allowing the Update trigger to modify the external record.
5. Reopen the document (double left click) to verify the external update. The reopen will trigger DECS to retrieve the external data and you can check that the cost center has been changed to S12 on the external system.

Conflict Detection

1. From the open document , double-click in the document window or press Ctrl+E to put the document into edit mode.
2. Change the cost center value back to E11 and press Ctrl+S to save the document or, from the menu bar, select File-Save. Do not close the document.

3. Change the cost center value to Z15 and press Esc or Ctrl+W to close the document. Select Yes to save and you should see the following conflict detection error.



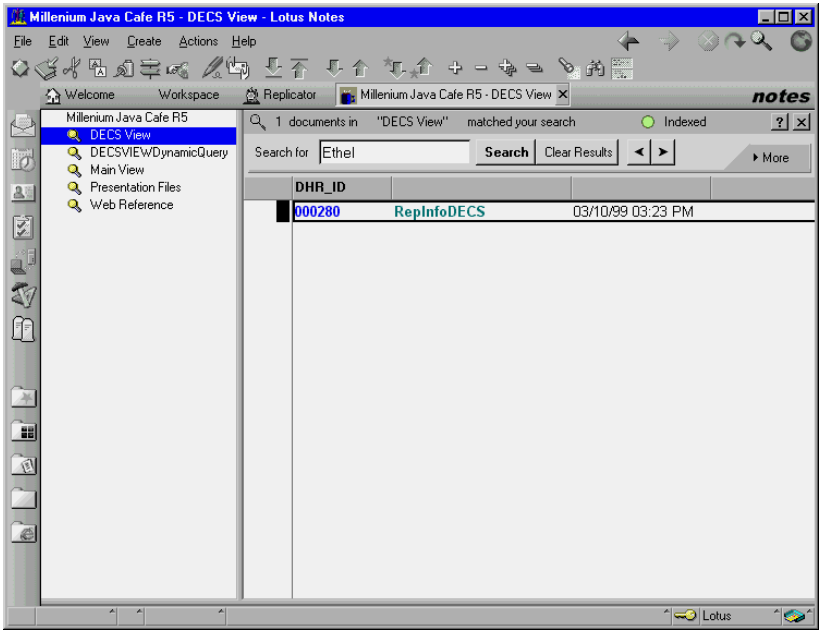
   Even though it was you that modified the document twice in sequence, this conflict is recognized because the internal control fields used to watch for a conflict do not get removed until you close the document. Remember to exit the document after the first save before attempting another change.

4. Click OK to close the error message box. Press Esc to close the document and select No when Domino asks whether you want to save your changes.

Full Text Search

DECS gives you the power to search your external data source by using the Domino full text search capabilities. Now that you have loaded your documents in the previous example, the following steps will index them and then search for Ethel:

1. From your Lotus Notes R5.0 desktop, highlight your database icon by clicking it.
2. Click the right mouse button and select Database. In the database selection menu that is presented, select Properties.
3. Click the Magnifying glass tab to show the index properties. Click the Create Index button and then click OK to accept the defaults.
4. After the index is created, open your database to your defined DECS view. For Millennium Cafe this is the DECS View. You will be presented with the Domino search bar.
5. From previous examples, it was shown that the stub documents in this view do not have any of the mapped data, other than the key value. With this in mind, type Ethel in the search bar (or the value of a non-keyed external field).
6. Click the Search button. Millennium Cafe presents the following result:



As expected, a full text search returns document 000280. Even though the document content does not store “Ethel,” the full text index does.

This is a powerful option for extending search capabilities to Notes and the Web. As documents are found in the result set, they are modified internally to invoke the immediate indexer (if that default option was not changed). This allows the full text index to remain complete.

Delete and Verify

Finally we will delete a document and verify that it no longer exists.

1. From the stub document view, or the DECS View for Millennium Cafe, select document 000280. Press the Delete key (or from the menu bar select Edit - Clear). This will place the little blue wastebasket icon beside your document.
2. Press F9 or select View - Refresh from the menu bar to refresh your view. This will trigger the DECS activity to remove the marked record from the external source.
3. From your document view, select Create and the form name for your document. For Millennium Cafe, select the RepInfoDECS form.
4. Before verifying the deletion, ensure your DECS Real time activity has been modified to trigger on Open only. Within the new form, enter 000280, or whatever the key was for your example document, and click Locate Employee Info.

5. DECS will respond with a message box indicating the record is gone. Refer to the "Create an External Record and Verify" section above in step 3 to see the message box that is returned.

For all of the examples in this section, the definition of DB2 EMPLOYEE table that was used can be referenced in the "DB2 Employee Table Definitions" Appendix of this book.

Summary

In this chapter you were introduced to Domino Enterprise Connection Services. The sections in this chapter included discussions and processes for:

- Installation of DECS during Domino installation.
- Domino configuration related to DECS.
- Starting the DECS server.
- Controlling the DECS server though NOTES.INI settings.
- Introduction to the DECS Administration Database, its navigator and the Connection and Activity views.
- Defining DECS connections with and without the wizard.
- Defining DECS activities with and without the wizard.
- DECS activity options.
- DECS and its relationship to the Domino Architecture components.
- DECS examples focusing on dynamic query.

You can find further detail on DECS in the "Which Tool to Use When" chapter in this book. This chapter enables users to consider their business integration requirements to determine the best tool (or tools) to use.

Chapter 3
Lotus Enterprise Integrator

Lotus Enterprise Integrator (LEI) is a server-based tool that allows movement of data between various data sources. Using the LEI server, applications can be written that read and write between supported data sources, and even access this data in real time. LEI is the latest version of the product previously named NotesPump.

This chapter will focus on the following areas:

- LEI Overview
  - LEI Server
  - LEI Development Client
  - LEI Databases
- LEI Installation
- LEI Administrator Database
- LEI Connectors
- LEI Activities
- LEI Sample Activities

Lotus Enterprise Integrator Overview

In this section we will introduce the LEI server, the LEI Development Client and the databases that are installed together with LEI.

LEI Server

The LEI server runs as a multi-threaded, multi-processing task on server operating systems. It performs the work of transferring data between data sources and destinations. It polls the LEI Administrator Database for instructions which are in the form of Activity Documents in order to determine what kind of data access or data transfer should be performed and when.

LEI Development Client

The LEI Development Client is a version of the LEI Administrator Database and a local LEI server subset program that resides on your development computer. The Development Client enables you to develop, run and test LEI Activities, and provides client product and connector .DLLs enabling developers to use LEI Activity form action buttons for Metadata selection and Field Mapping. It also lets you develop LotusScript agents using LEI classes from Domino Designer for Notes R4.6 or Domino Designer R5.0, resulting in the ability to run custom LEI LotusScript routines from the LEI Servers.

LEI Databases

There are four Domino databases that are copied onto your Domino server during LEI installation. A brief description of each follows.

LEI Administrator Database

The LEI Administrator Database is a Domino application that acts as the control station for all operations performed by an LEI server. It contains forms which define LEI server configuration parameters, connection specifications, and activity details. The Administrator Database will be discussed in detail later in this chapter.

LEI Log Database

The LEI Log Database is a Domino application that logs all LEI server activity. Information included in the database includes server start and stop times, activities executed, records transferred, as well as any errors specific to an activity. Two LEI Activities, Admin-Backup and Admin-Purge are provided for the maintenance of this database.

Script Vault

The Script Vault is an optional database that can be used to store scripted agents, which may be executed using the Scripted Activity. It contains an action called Catalog All Agents, available at the View level, which inventories the agents and creates a *tracking* document for each. This document contains information about each agent, such as name and creator, and allows the user to add additional comments about the agent. Each time the Catalog All Agents button is pressed, the agent tracking document is updated if required, new agents are cataloged, and tracking documents that no longer have associated agents are deleted. We suggest that this database be included in regular backup procedures.

LEI Documentation

The LEI Documentation Database contains much helpful information regarding construction of LEI Connectors and Activities, as well as server administration. You should read it to become more familiar with LEI capabilities.

Installation of Lotus Enterprise Integrator

This section presents the LEI installation requirements and procedures.

Server Installation Requirements

Hardware Requirements:

- 30MB of disk space
- 64MB RAM

Software Requirements:

- Domino Server Release 4.6x or above.
- Notes Client Release 4.5 or above.
- Client connectivity software that matches the data source you will be working with. For example, CAE for DB2, SQL Net for Oracle, and so forth.

LEI runs on Microsoft Windows NT 4.0, OS/2 Warp 3.0 or 4.0, HP-UX 11.0, Sun Solaris 2.5.1, and IBM AIX 4.1.4, AS/400 and will be available on OS/390 later in 1999.

Installing LEI Server

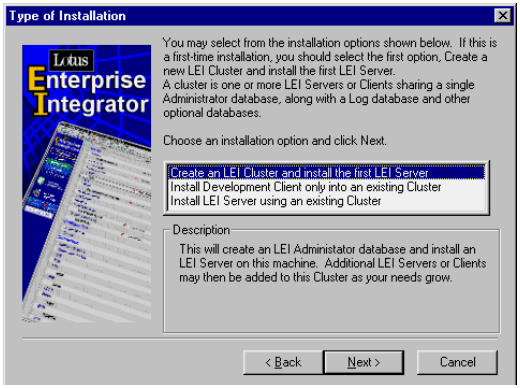
You can download an evaluation copy of LEI from

`http://www.lotus.com/enterpriseintegration`

When you run the executable you downloaded, the setup program will begin.

Note Before starting the setup procedure, make sure the Notes executable path is set in your PATH environment variable, or setup will fail.

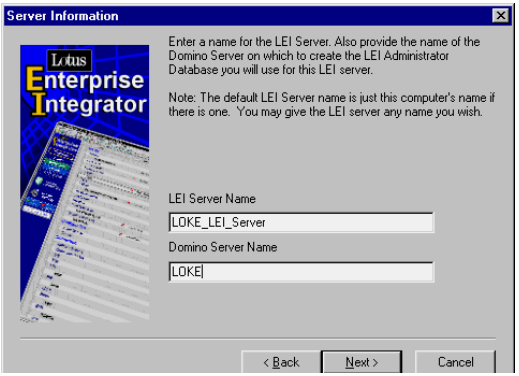
Click Next to continue installing LEI.



Options for installing LEI are displayed. A brief description of each option is displayed when the option is selected.

- Select the first option if this is the first LEI server in your organization. An LEI Cluster will also be created. An LEI Cluster allows you to control several LEI servers from the same administration application.
- Select the second option if you already have an existing cluster and want only to install the LEI Development Client.
- Select the third option if this server will be installed into an existing cluster.

Once you have made your selection, click Next to continue.

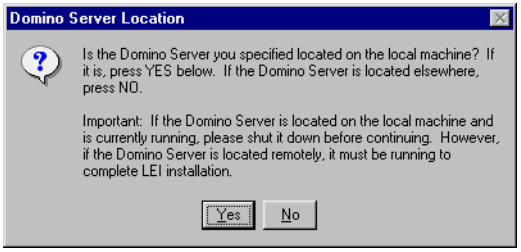


On the Server Information screen, enter the following:

- The name of the LEI server
- The name of the Domino Server where the LEI Administrator Database for this server will be located

Once you have completed this information, click Next to continue.

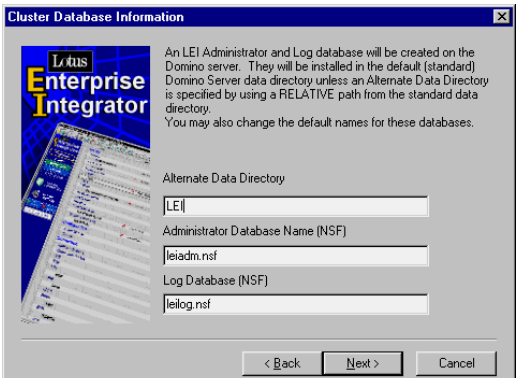
The next screen poses an important question.



If you are installing LEI on the same machine with the Domino server, click Yes. If LEI is being installed on a separate machine, click No.

Note If the you are installing LEI on the same machine with the Domino server, make sure the server is not running during the LEI install. If LEI is installed on a separate machine, then the Domino Server you specified in the previous step must be running to complete the LEI installation.

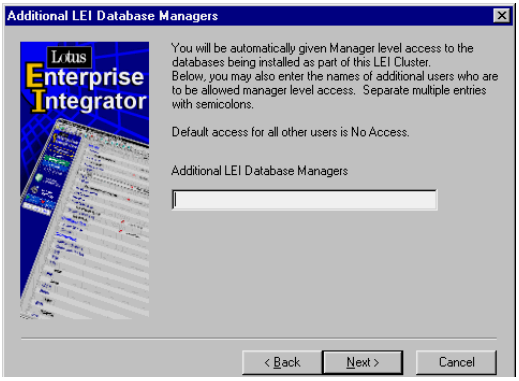
Once you have selected Yes or No, the installation will continue with the Cluster Database Information screen, shown in the next figure.



This screen allows you to select options for the placement and naming of the LEI Administrator and Log databases.

- Alternate Data Directory — If no selection is filled in, the Administrator and Log databases will be placed in the Domino data directory. If a selection is made, a subdirectory will be created in the Domino data directory and the Administrator and Log databases will be placed here.
- Administrator Database Name — The default, leiadm.nsf will appear. You may change the default.
- Log Database — The default, leilog.nsf will appear. You may change the default.

Once you have completed these options, click Next to continue.

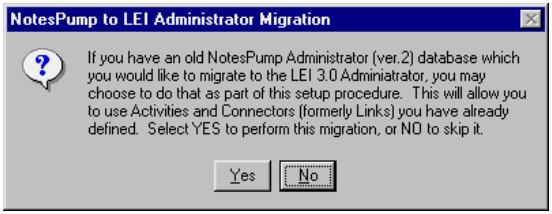


By default, you will automatically be given Manager level access to the LEI Administrator and Log Databases. All other users will have No Access. This screen allows you to enter additional LEI Database Managers by listing their user names. Multiple names should be separated with semicolons.

Remember to add additional names as fully qualified names, including organization (plus country and organizational unit if used.)

Once you have finished listing manager names, click Next to continue.

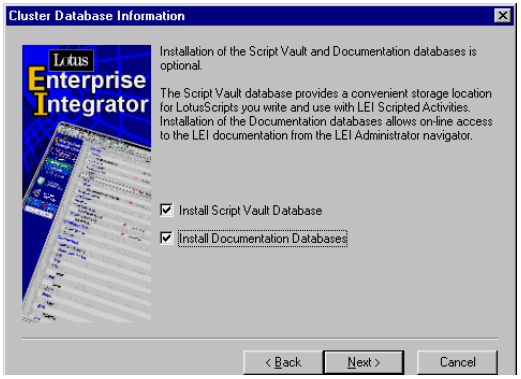
Tip If you don't know exactly who should be given manager access at this time, you can always change it later by manually changing the ACL for the LEI Administrator and Log Databases.



The next screen asks if you have a previous version of NotesPump Administrator that you would like to migrate to the LEI Administrator Database. This will allow you to use Activities and Connectors (formerly Links) you have already defined and used with NotesPump. Select Yes to perform the migration or No if you don't have a previous version of NotesPump Administrator to be migrated.

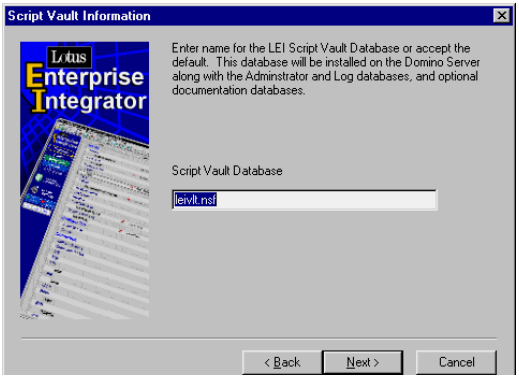
Note Do not copy and paste Activities or Connections from a 2.5a NotesPump Administrator into an LEI 3.0 Administrator. The administrator upgrade aspect of install can take care of this for you.

Continue with the installation.



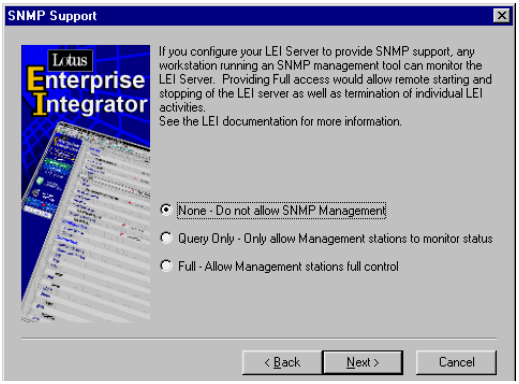
If you wish to use the Script Vault and Documentation Databases, mark the checkboxes next to these items. These are optional and do not affect functionality of LEI.

Click Next to continue with the installation.



If you selected to install the Script Vault Database in the previous screen, this screen prompts you to accept the default name for the database, leivlt.nsf, or specify a different name of your choice.

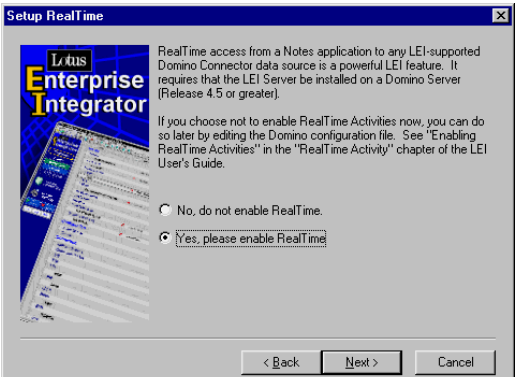
Once you have completed this, click Next to continue.



LEI Servers may be monitored by Simple Network Monitoring Protocol (SNMP). Select the level of support you desire for your server from the following choices:

- None — No SNMP required
- Query Only — Provides the ability to collect information
- Full — Provides the ability to Monitor, Start, Shut down, and Stop LEI server as well as kill Activities

Make your selection and click Next to continue.

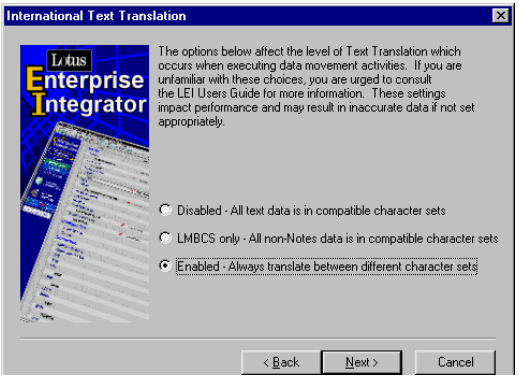


Select if you would like to have RealTime access enabled from the LEI server. If you choose to enable this feature, the LEI server must be installed on the same machine as the Domino Server where the applications that will be accessing the external data are located.

Make your selection and click Next to continue.

The LEI server may be configured to run as an add-in task on your Domino Server, or as a stand-alone server. Running the LEI server as an add-in task will allow you to start the server automatically during Domino startup, or manually from the Domino Server console.

Make your selection and click Next to continue.



All text data within LEI maintains a reference to its own character set. This reference is determined automatically by the connector when the data is fetched from a database. If the data is inserted into a database with a different character set, LEI automatically translates the text data just prior to storing it. This process maintains maximum performance by eliminating any unnecessary translation. Performance may be further enhanced through LEI's three levels of translation support:

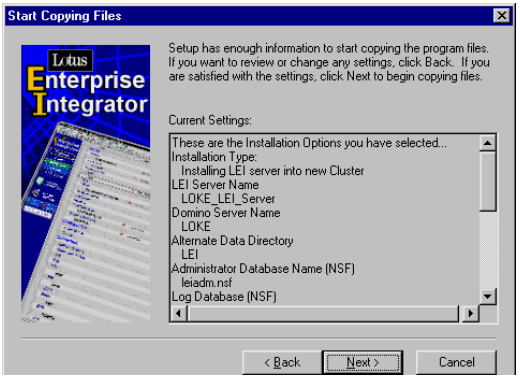
- Disabled — No translation will take place
- LMBCS — Only non-Notes data will be translated
- Enabled — Always translate (default)

If in doubt leave the setting at the default value.

Click Next to continue.

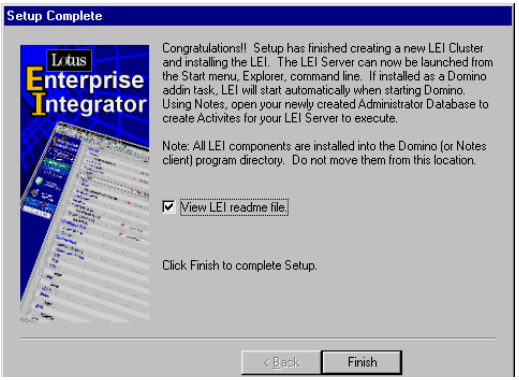
Select the Program Folder in which you want LEI to be installed.

Click Next to continue. The installation options you selected are displayed on the screen shown in the following figure.



Review the settings you selected. If any settings are incorrect, or if you would like to make changes, press the back button to return to the previous screens and make your changes.

Once this is complete, click Next to continue.



Congratulations, you have completed the setup.

All of the LEI executable files and the LEI.INI will be in your Notes/Domino directory. In our example, this is C:\NOTES. They should not be moved from this location.

Verifying Installation and Connectivity

Before proceeding, you should verify that your LEI environment has been properly installed and that client data source access libraries are available and functional on the Domino Server.

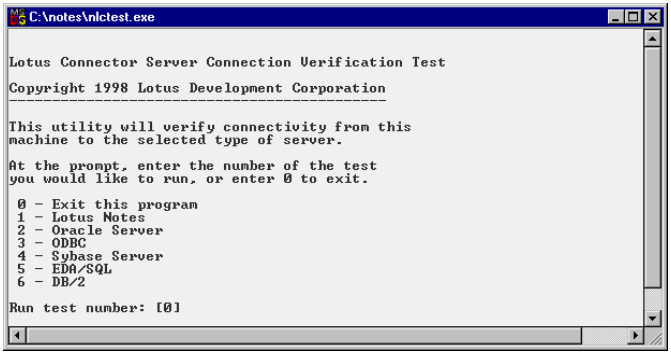
There is a connectivity test program included with LEI that will test this connectivity for you. It will be located in the Notes/Domino directory. To use this utility, follow these steps:

1. Find out the name of the LCTEST executable for your operating system. It has the following names for the following platforms:
   - *nlctest.exe* for Windows 95/98/NT (Win32)
   - *ilctest.exe* for OS/2
   - *alctest.exe* for Windows NT on Alpha
   - *lctest.exe* for AIX/UNIX

   and type it on a command line, like this (for Win32):

   ```
   c:\notes>nlctest
   ```

2. The following screen will appear.



3. Make your selection, based on the data source you will be connecting to.
4. You will be prompted for the information required to log in to that particular data source. This may include server or database name, user ID, and password.
5. If the test fails, check to make sure the external data client located on the LEI server, for example CAE for DB2 or SQL Net for Oracle, is properly configured. Refer to the particular database documentation for specifics.

Once connectivity is established, you should be able to successfully continue running LEI.

Additional information on configuring your system for connectivity can be found in the online documentation database included with LEI, Lotus Connectors Connectivity (LCCON.NSF).

Installing the Enterprise Integrator Development Client

To install the Client, follow the steps described previously to run the LEI Setup program on the Lotus Notes Client computer where you want to install it. The only difference is to select the Install Client option, instead of the other two Install Server options. You will be asked to reference an existing LEI and Domino Server. A Client Configuration document will be added to the LEI Administrator database on the Domino server you specified.

Once the Client is installed, and the appropriate client connectivity software is installed on the computer you use, the features are enabled:

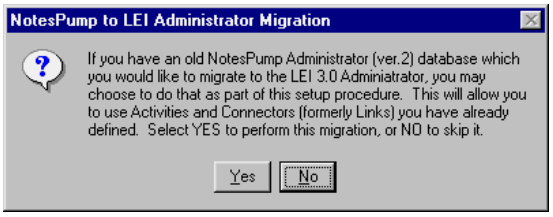
- Metadata selection
- Mapping (browsing) from within an Activity form
- LEI LotusScript development

In addition, scripts can be verified and debugged on the local machine running the Development Client.

Migration from NotesPump to LEI

If you are an existing NotesPump user, you've probably invested significant time and effort into constructing Link and Activity Documents in the NotesPump Administrator Database. You can migrate these documents over to the LEI Administrator Database during the setup process for LEI.

The LEI Setup will detect a previous version of NotesPump present on your system. It will ask you the question shown in the following figure:



Click Yes, so that the contents of your existing NotesPump Administrator will be migrated to the new LEI Administrator.

You will then be prompted for the Domino server name and location of the NotesPump Administrator database. Enter the name of the Domino Server hosting the old NotesPump Administrator database. If it is on the local machine, you must enter "local", and *not* the name of the server, since a local Domino server should not be running during setup. If the Domino Server is located remotely, it must be up and running and accessible. Be sure to include the complete path relative to the Notes/Domino data directory. Continue with the rest of the install, as described previously in this chapter.

Note If you choose to upgrade NotesPump, the new LEI server will use several of the old NotesPump settings for the new LEI server. For example, if the Text Translation setting for your NotesPump Server installation was set to LMBCS only and the Poll Interval was set to 15 seconds, these settings will be preserved in your new LEI server. For this reason, some of the screens asking for these settings will be skipped when upgrading.

There are a few things you should be aware of when migrating from NotesPump to LEI.

- The time required to migrate depends on the size of the NotesPump Administrator database and whether it is local or remote. It may take several minutes to an hour or more if the NotesPump Administrator database is very large and remote.

- Migration does not perform an uninstall of the NotesPump installation. You must manually do this if you no longer will be running NotesPump.
- The old NotesPump Administrator remains intact. Migration does not remove or alter the contents of this database. This means that you will have essentially two copies of every Activity; the one in the NotesPump Administrator and the new copy in the LEI Administrator. If you still have an active NotesPump server running NotesPump Activities, it is possible that an active LEI server could rerun the same Activity from the LEI administrator after the Administrator migration. For this reason, if Activities were previously scheduled in the NotesPump Administrator, they are initially disabled in the LEI Administrator. To re-enable Activities when you are ready to run them with LEI 3.0, select Enable Activities from the Actions menu to enable selected Activities.
- If you are upgrading multiple NotesPump Servers, each Server Upgrade will need to be done individually by running the installation program and selecting the upgrade option. Do not select the option Migrate NotesPump (ver 2.) Administrator Database for these subsequent Server upgrades, as this should have already been done during the first server upgrade.
- The new LEI server will use the same name in the new LEI Administrator database as it did in the NotesPump Administrator database, and you cannot change it.

Starting and Running the LEI Server

When started, the LEI server connects to the LEI Administrator database and immediately runs any overdue activities in the Administrator database. It continues to poll the Administrator database at the interval defined in the Server Configuration document and to run scheduled Activities until shut down.

There are two options for starting LEI.

Running LEI as a Domino add in task

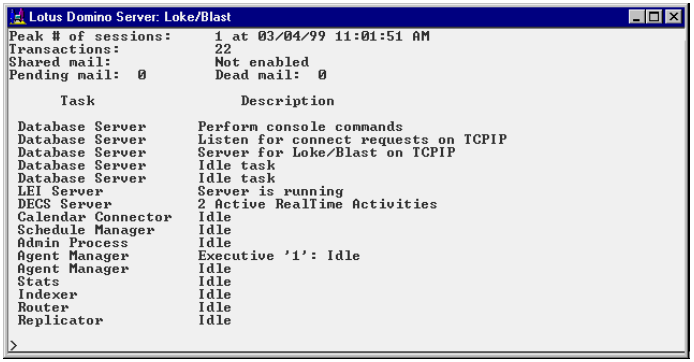
If you have installed LEI as a Domino add in task, it will be included as

```
ServerTasks=...,LEI
```

in the NOTES.INI file. Therefore, it will start when the Domino server is started. You can type

```
sh ta
```

at the Domino Server console to see it running.



The Domino server console will display messages from the LEI server. For example, when running an activity, you will receive start, completion and any error messages for the activity here. To administer the server, use the Administrator Database. This is detailed in the "LEI Administrator Database Commands" section of this chapter.

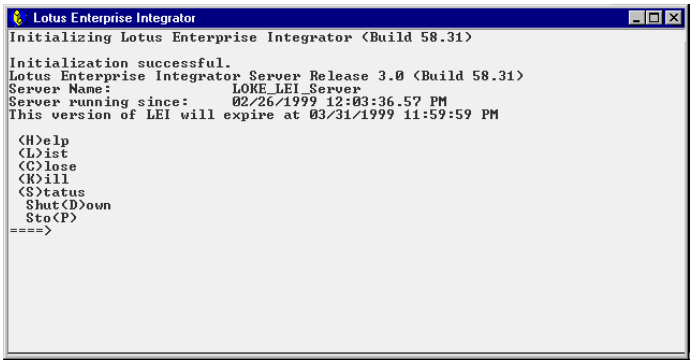
Running LEI as a Standalone LEI Server

If you chose not to run LEI as a Domino server add in task during the install, you must run LEI as a separate server.

From the Notes/Domino executable directory, type the name of the LEI program with the prefix determined by operating system (n for Windows 95/98/NT, i for OS/2, a for Windows NT on Alpha, none for UNIX). For example, for Windows 95/98/NT you must type

```
c:\notes>nlei
```

The LEI server Console will appear, as shown in the following figure.



Use the server console to run server console commands and to receive messages from the LEI server regarding activity status and general server status.

Server Console Commands

When running LEI as a stand-alone server, you have the following server console commands available. The parentheses identify a shortcut character that can be typed instead of the full command. If you type the full command, do not include the parentheses:

- (H)elp — Provides command usage information and a brief explanation of the commands available from the server console.
- (L)ist — Displays a list of activities that are currently being executed, with a number associated with each activity. This number can be used with the Close and Kill commands to terminate a particular activity.
- (C)lose — This is the preferred method to terminate the current activity using the number given by the List command. More than one activity can be terminated at a time by listing the activity numbers separated by a space character.

  A request for orderly termination will be sent to the activity process, which acts on it following the currently occurring database operation by immediately disconnecting from all databases and stopping the specific activities. When an activity is closed, LEI does not consider it an error. On the other hand, the activity is not a completed activity in that dependent activities are not executed.
- (K)ill — The Kill command will terminate one or more activities immediately. More than one activity can be killed at a time by listing the activity numbers separated by a space character. Regardless of the current activity state, it will immediately be terminated. Using this command can leave the system in an unstable state; therefore, this command should be used only when absolutely necessary.
- (S)tatus — Shows the status of the server, including startup information and basic configuration.
- Shut(D)own — Shuts down the server when the last running activity is finished. This prevents the LEI server from starting any new activities and waits for all currently running activities to stop. After all the activities have been shut down, it stops the LEI server program. This is the preferred method of stopping the LEI server.

- Sto(P) — Stop the server immediately. This command prevents the LEI server from starting any new activities and sends a Close command to all the running activities. After a period of time (as specified in the Server Configuration document's Activity Requested Shutdown Timeout field in the Administrator database), any activities which are still running are killed and the LEI server program stops.

LEI Administrator Database Commands

To administer the LEI server when running as a Domino add in task, you must use the following LEI commands from within the LEI Administrator:

- Close Activity
- Kill Activity
- Shutdown Server
- Stop Server

More detail on LEI Administrator Database commands can be found in the "Using the LEI Administrator Database" section of this chapter.

Changing LEI Run Mode

The following procedures outline how to change the mode in which LEI runs.

Changing from Stand-alone Server to Domino add in task

1. Stop LEI server if it is currently running.
2. Stop Domino server.
3. Modify the NOTES.INI file to include

   ```
   ServerTasks=.....,LEI
   ```

4. Restart Domino server and notice the LEI task starting.

If you don't want to modify the NOTES.INI file, you can type

```
load lei
```

at the Domino Server Console, but you will have to do this every time the Domino server is started.

Changing from Domino add in task to Stand-alone Server

1. Stop LEI server if it is currently running.
2. Stop Domino server.
3. Modify the NOTES.INI file to remove LEI from the list of tasks specified after

   ```
   ServerTasks=
   ```

4. Restart Domino server. The LEI task will not be running.
5. Manually start LEI from a DOS prompt by typing

```
c:\notes>nlei.exe
```

Comparing Server Run Modes

The table below compares performing common LEI tasks when running as a stand-alone server and as a Domino add in task.

	LEI as Domino Add In Task	*LEI as Stand-alone Server*
Starting LEI server	1) Automatic on Domino server with ServerTasks=..., LEI set in NOTES.INI 2) LOAD LEI command at Domino Server Console	Run c:\notes\nlei
Administering server	LEI Administration DB Menu Commands	1) LEI server Console Commands 2) LEI Administration DB Menu Commands
Check status of activities	Active view in LEI Administration Database	1) (L)ist command from LEI server Console 2) Active view in LEI Administration Database
Stopping LEI server	1) Use the command TELL LEI QUIT at the Domino Server Console 2) Shutdown Domino 3) LEI Administration DB Menu command	1) Shut(D)own or Sto(P) command from LEI server Console 2) LEI Administration DB Menu command

Using the Development Client

The LEI Administrator Database can be accessed from any Notes Client that has the appropriate access to the database. Once you are connected to the LEI Administrator on the Domino Server, you can construct Connection and Activity documents. However, there are certain advantages to using the LEI Development Client.

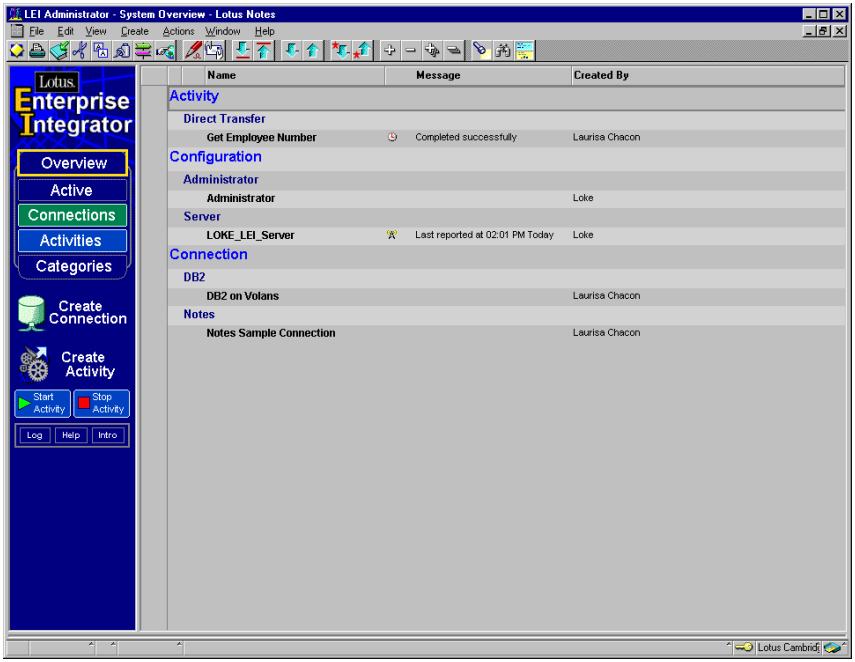
The first advantage is that after the Development Client and the appropriate connectivity software have been installed and configured, the Select Metadata and Mapfields action buttons are functional. This allows you to browse the backend data source you are connecting to. For more information on the capabilities of these action buttons, refer to "LEI Activities — Using Action Buttons to Construct Activities" later in this chapter.

The second advantage is that you will be able to develop and debug scripts using the additional LC LSX classes included with LEI. Note that when running scripts in development (from a Client), the directory containing LEI.INI must be in the path. In addition, use a Notes ID that does not have a password, or that has had Share Password with Add-ins enabled.

Using the LEI Administrator Database

The Administrator database contains the information that LEI servers require for operation. The Administrator database contains all of the configuration, connection, and scheduling elements that LEI servers use to collect, transfer, and write data.

The Administrator interface is shown in the next figure. The Navigator is on the left, and the currently selected view, Overview, shows all existing Activity, Configuration, and Connection documents. The Navigator allows you to select different views.



LEI Administrator Documents

There are three types of documents contained in the LEI Administrator Database.

Activity Documents

An activity document contains instructions for the LEI server and as such it is the central element of LEI. Activity information in the Administrator database tells an LEI server what to do, when to do it, and how. LEI can execute several kinds of activities to move and manage data.

Configuration Documents

Two configuration documents containing LEI server settings are provided by you during the install process. A third configuration document is available for LEI Development Clients. These documents cannot be created manually. However, they can be edited to suit your preferences.

The Administrator Configuration document contains the name and location of the Log database, LEI Help database, and Scripted Agent database, as specified during LEI install.

The Server Configuration document contains defaults such as poll interval, maximum number of activities, maximum duration of activities, maximum consecutive failures, and activity shutdown request timeout. Configuration choices made during install, such as SNMP Support and International Text Translation level, are also located within this document.

The Client Configuration document contains information for any LEI Development Clients that have been installed referencing this Administrator Database on this Domino Server.

Connection Documents

A connection document holds information that defines a connection to the data servers and databases that LEI must access for transferring data. A single connection document can be shared by many activities.

LEI Navigator

The LEI Navigator provides buttons for creating Connections and Activities and view selections for looking at different aspects of your LEI configuration. The menu bar provides access to the actions and views also.

Overview

The Overview view lists all documents: Activity, Configuration, and Connection documents, along with each document author's name.

Configuration View
The Configuration view shows the Administrator Configuration document with its defined Status Interval, and Server Configuration documents with defined Poll Interval, Max Activities, Max Duration, and Max Failures. This view gives you a quick look at your LEI configuration.

Active
The Active view displays currently running activities.

Connections
The Connections view displays, by type, all of the connection documents that have been created on the system.

Activities
The Activities view displays, by type, all of the activities that have been created on the system.

Categories
The Categories view displays all of your connectors and activities according to how they have been categorized.

Create Connection
The Create Connection selecton allows you to create a new connection document. A dialog box appears after you select this option, allowing you to select the type of connection you want to create. The list shows all the connectors that currently are installed with LEI.

Create Activity
The Create Activity selection allows you to create a new activity document. A dialog box appears after you select this option, allowing you to select the type of activity you want to create. See "LEI Activities" later in this chapter for an explanation of the different activities.

Start Activity
Click Start Activity to place the selected activity in the Run ASAP queue. When you select Run ASAP for an activity, it will run the next time the LEI server polls the LEI Administrator for activities to run. You must have selected the activity you want to run before selecting this option.

Stop Activity
Click Stop Activity to stop a currently running activity. You must have selected the activity document before selecting this option.

Log View
The Log view lists all LEI logs, categorized by type (activity, operation, and server). Log documents are further categorized by server and start date.

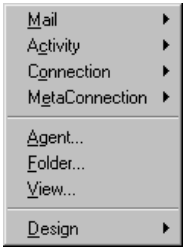
Help
The Help view opens the online version of the LEI documentation.

Intro
The Intro button accesses an online description of the LEI navigator.

LEI Administrator Menu Commands

In addition to the standard Notes commands found in the menus, the LEI Administrator menus provide specific commands for working with LEI. They are described in this section.

Create Menu Commands



The Administrator Create menu includes the following commands.

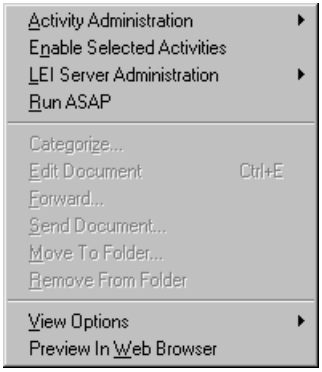
Activity
The Activity submenu provides a list of all LEI activities that can be created. Select the type of activity you want to create from this submenu.

Connection
The Connection submenu provides a list of all connections that can be created. Select the type of connection you want to create from this submenu.

MetaConnection
The MetaConnection submenu provides a list of all MetaConnections that can be created. Select the type of MetaConnection you want to create from this submenu.

Actions Menu Commands

The Administrator Actions menu includes the commands described in the following section.



Activity Administration

The Activity Administration submenu provides commands related to Activities. A description of these commands follows.

- Clear Browsing Cache — The Clear Browsing Cache command deletes the cache for the currently selected or open activity.
- Clear Lock — The Clear Lock agent clears an activity document's Lock field and related fields so that the activity can be run again.
- Close Activity — This command closes a selected activity that is currently running. Close is the preferred method (over the Kill command) of prematurely terminating a currently running activity.
- Kill Activity — This command immediately terminates an activity process. The preferred method of stopping an activity is to use the Close command, which provides an opportunity for the activity to clean up its database connections and log the termination.
- Set Designated Server — This agent is available from the Action List, and may be used to designate a particular LEI server to run a particular defined activity. The LEI Administrator normally assigns individual activities to be run on LEI servers according to available LEI server processing times. Set Designated Server, when used, overrides the default operations and assigns individual activities to be run on specific LEI servers. The agent acts on selected activities and synchronizes the server ID field based on the server name in the designated server field in the activity.

LEI Server Administration
The LEI Server Administration submenu provides commands related to the operation of the LEI server. A description of these commands follows.

- Shutdown Server — This command terminates the LEI server. This command is preferred over the Stop command for terminating the LEI server because it closes the server down in an orderly fashion. It prevents the LEI server from starting any new activities and waits for all currently running activities to shut down. When all activities have shut down, it stops the LEI server program. For a rapid termination of the server, use the Stop command.
- Stop Server — This command quickly terminates the LEI server. It prevents the LEI server from starting any new activities and sends a Close command to all currently running activities. After a period of time (specified in the Server Configuration document's Activity Requested Shutdown Timeout field in the Administrator database), any activities which are still running are Killed (see the Kill command) and the LEI server program stops. For an orderly (and safer) shutdown of the server, use the preferred Shutdown command.

RunASAP
RunASAP causes the currently selected activities to be executed by the LEI server the next time the server polls the LEI Administrator database.

LEI Connector Documents

LEI requires connection documents to allow integration with external data sources. These connections use the Lotus Connector architecture, a common set of connectors available to all Lotus integration products.

A connection document is a defined connection from the LEI server to a specific data source. The connection document provides specific access parameters, such as server names, user IDs and passwords. Connections connect LEI to your system databases and other data sources. Once a connector document to a database has been defined, LEI can access that database for executing activities such as data transfers. The forms used for each connection type (Lotus Notes, Oracle, DB2, etc.) are different.

Connections are the building blocks for activities, which are discussed in a later section. You should construct at least two connections before starting an activity document. This is because when activities are created, connections must be chosen from the existing set of defined connections. After including the information from a defined connection in an activity, you can override individual values, if needed, for that specific activity.

The information required in a connection document may be different for different database products, so the connection documents provide standard Connection Name fields as well as sections and fields specific to the database. Refer to the specific Connection chapters in the LEI online documentation about the type of connection you are creating. If you are using a connector you have purchased separately, refer to the documentation provided with the connector for information specific to it.

Connectors and Supported Data Sources

LEI 3.0 provides the following standard Domino Connectors for the associated data sources:

- DB2
- Domino Directory (NAB)
- EDA/SQL
- File System
- Lightweight Directory Access Protocol (LDAP)
- Notes
- Novell Directory Services (NDS)
- Open Database Connectivity (ODBC)
- Oracle
- Sybase
- ZMerge Text Connector

Note The connector for Domino Directory is called NAB because Domino Directory used to be called Name and Address Book.

Additional Domino Connectors are available and can be purchased separately. These include Domino Connectors for:

- JD Edwards OneWorld
- SAP R/3
- PeopleSoft
- BEA Tuxedo

Constructing Connectors

Before creating any Connections, you should verify that LEI can communicate with your external database sources by running the data source-specific test, LCTEST, as described in the installation section of this chapter.

To create a new connection document, follow these steps:

1. Choose Create - Connection. A pull down menu will appear.
2. Select the database type option from the Notes menu. The connection document will open in edit mode.
3. Enter the required information in the Connection Definition form. Each connector requires different information, depending on the source it will be connecting to. Refer to LEI online documentation for details.
4. Once created, these documents can be displayed through the Connection views in the LEI Administrator database.

A connection can be used by one or more activities.

Testing Connectors

After creating your connections, you should test them using LCTEST. LCTEST is a testing program that uses the native database client access method to test native connectivity. LCTEST, which must be run with a running LEI server, attempts to connect using Connections defined in the currently running LEI Server's Administrator database. LCTEST tests that a connection can be made via the information found in the connection document.

To use LCTEST, use the following procedure:

1. From a command prompt, type:

   ```
   c:\notes>?lctest "Connector Name"
   ```

   where ? is a letter determined by the operating system (n for Windows 95/98/NT, i for OS/2, a for Windows NT on alpha, none for AIX/UNIX)

2. If this is successful, your screen will display:

   ```
   Testing Connector Name

   Connector Name Connection Successful
   ```

3. If you receive an error, make sure you have run LCTEST successfully for the data source you are trying to connect to with the connector being tested.
4. If LCTEST runs successfully, return to the connector document and review it for proper user IDs, passwords, typing errors and so on.

MetaConnectors

A MetaConnector is a special kind of LEI connector that is optional, and provides pre-processing operations on connector data prior to transfer within a defined activity form. Once a MetaConnector is created, the MetaConnection document can be found in the views under Connections since it is a type of connection.

There are four MetaConnectors supported by LEI 3.0. They are described in the following sections.

Collapse/Expand MetaConnector

The Collapse/Expand MetaConnector provides the ability to take multiple records from a data source table, *collapse* them to a single form field, and perform the reverse operation, or *expand* the data into multiple records.

Connection Broker MetaConnector

The Connection Broker MetaConnector allows data password information to be input to a Domino application, then sent as parameters to the back end source application to validate individual user logins to the external system in order to write data. It can also be used to direct data to and from different connections based on contact information that is supplied with each row of data. The additional connection properties are specified through *extra* fields in the fieldlist supplied by the client. The MetaConnector properties allow the client to define the fieldnames that contain the various pieces of contact information.

Metering MetaConnector

Use the Metering MetaConnector to track data flowing to and from another Connection or MetaConnection. The Metering MetaConnector has no effect on any operations; it simply monitors the volume of data transferred.

Results from the metering are logged to a file. The specific information logged is configurable through use of the Metering MetaConnection document.

Order MetaConnector

The Order MetaConnector is useful when ordering data sets from different server sources. For example, a DB2 table on an AS400 system and a Domino database on a Domino server may use different order systems when ordering data. This can result in a data set comparison problem when using an LEI Replication Activity, which requires data sets to be ordered in parallel for data set comparisons. The Order MetaConnector may be applied in this situation to pre-process the data sets to be compared during the activity, ensuring the order pattern will be in parallel for accurate data set comparisons during the Replication Activity.

Constructing MetaConnectors

To construct a MetaConnector:

1. Since a MetaConnector will reference an existing Connector, make sure you have defined the connector you will be working with before attempting to create a MetaConnector.
2. Choose Create - MetaConnection. A pull down menu will appear.
3. Select the MetaConnection type from the menu. The MetaConnection document will open in edit mode.
4. Enter the required information in the MetaConnection form. Each MetaConnector requires different information, depending on the type of preprocessing it will be performing. Refer to LEI online documentation for details.
5. Once created, these documents can be displayed through the Connection views in the LEI Administrator database.

LEI Activities

Activity documents contain the information that instructs the LEI server to execute an event, such as a data transfer. Each activity type has a specific Domino form that can be used to create the activity. Each LEI activity is described briefly in this section. For more information, refer to the LEI online documentation.

Admin-Backup Activity

The Admin-Backup Activity creates a backup copy of your LEI Administrator database (LEIADM.NSF), and, optionally, the Script vault database (LEIVLT.NSF).

Use the Admin-Backup Activity when you want to create safe backup of the LEI administrator and associated documents.

You should use the Admin-Backup Activity at regular intervals to ensure that you always have backup copies of LEI activities, connections, and configurations that have been created. You can also use the Admin-Backup Activity when your Administrator database becomes cluttered with too many documents and you want to clean up the database while at the same time keeping copies of everything already there.

Admin-Purge Log Activity

Use the LEI Admin-Purge Log Activity to purge the LEI log database of documents older than a user-specified number of days.

In order to save any logs, you must copy them to another Domino database before running the Admin-Purge Log Activity.

You should use the Admin-Purge Log Activity for the LEI logs at different times, depending on your situation. For example, if the LEI logs take up too much disk space in your environment, you may want to run this activity on a fairly regular schedule.

Archive Activity

An Archive Activity moves data from one database to another, deleting the original records from the source database as they are moved to the destination database. The source data and target location are indicated by metadata name (table, form, and so on). Selection of documents to archive can be done with either a condition or a relative timestamp.

Some reasons to use the Archive Activity are:

- To protect data on a regular basis
- To archive infrequently accessed data
- When migrating from one database to another
- To create space on a full disk

You can use the Archive Activity in combination with other LEI Activities. For example, you can combine the Archive Activity with a Polling Activity. You could then poll your system databases to determine when data was last accessed, and then archive it based on a specified date of its last access.

Command Activity

A Command Activity executes operating system, database and SQL commands.

In order to execute an operating system command, select No Connector and enter the command as it would be entered at an operating system command prompt.

If you want to execute a database- or connection-specific command, you must select the appropriate connection document.

You should use the Command Activity when you need to execute a specific command either on a connected database or on the operating system on which the LEI server is running.

For example, you can use the Command Activity to:

- Execute an SQL statement on a supported connection
- Execute an operating system command

Direct Transfer Activity

A Direct Transfer moves data from one database to another. The data to be transferred is identified by a statement, for instance, an SQL query or a selection formula. There are three ways to have a Direct Transfer Activity executed:

- Immediate, by clicking the Run ASAP button
- Scheduled, using the Activity scheduling options
- By clicking the Start Activity icon in the Navigator

You can combine a Direct Transfer Activity with a Polling Activity to trigger a data transfer when a specified condition is met.

DPROPR Activity

A DPROPR Activity enables propagation of IBM Data Propagator (DPROPR) Consistent Change Data (CCD) table information to any LEI data source. The CCD table identifies changes that should be applied to target destinations to keep them synchronized with the data source. These changes are Inserts, Updates, or Deletes.

Polling Activity

Polling is a way of causing an activity (such as a Direct Transfer or Replication) to execute when a condition is met. The polling document itself does not contain the instructions to execute. Instead, it triggers another activity or activities that are configured in separate documents.

The Polling Activity does not re-establish a closed connection to a data source. If you are polling a connection, the connection must remain open as long as necessary relative to the polling interval. For example, if you have configured Sybase to time out its connection after 10 minutes of inactivity, but your Polling Activity has a polling interval set to 15 minutes, the Sybase connection may time out and close before the Polling Activity checks the connection. The Polling Activity would fail in this case.

You should use the Polling Activity when you need to execute some action when another action or event occurs. For example, say the Human Resources department uses an Oracle database for employee information, and you need to maintain this same information in a Domino database so the Engineering, Sales, and Marketing departments can access it. You could use a Polling Activity to check the Oracle database for new or changed records,

and when the Polling Activity detects such an event, it would trigger a Replication Activity to copy the changed data to the Domino database.

RealTime Activity

A RealTime Activity provides synchronous access from a Domino application to an LEI supported external data source.

The LEI RealTime Activity intercepts Domino database events and acts on them.

When Domino users open, create, update, or delete Domino documents, these events are intercepted and acted upon, obtaining *real-time* access from the Domino document to backend sources supported by the LEI server. Once a system administrator has created the RealTime Activity, Domino users can open, create, update or delete backend data directly and transparently through their familiar Notes Clients or a Web browser.

The Domino document events are intercepted and handled by the LEI server operating on the Domino Server that is hosting the Domino application.

Replication Activity

LEI Replication synchronizes databases by updating one database from the data in another database, where one of the databases is master and the other non-master. In case of a conflict, the master database will "win" and overwrite the data in the non-master. This is different from Domino replication where a replication conflict document is created if there is a conflict.

Replication involves metadata from two connections (replication is done at the metadata level). There are two types of replication that are available within the Replication Activity form:

- Primary Key Replication
- Primary Key/Timestamp Replication

They will be explained in the next sections. A condition can also be used in the Replication Activity document's Conditional Clause field to further refine the result set to achieve an enhanced, selective replication.

The Replication Activity is recommended for use when you want to:

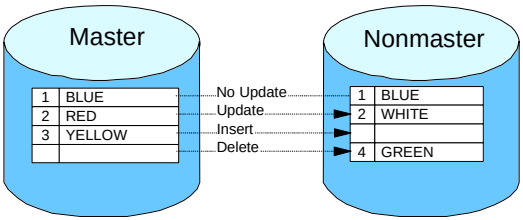
- Synchronize data in two databases
- Merge data from one data source into another data source

Primary Key Replication

Primary Key replication replicates data based on a unique key common to both connections, which can consist of one or more fields in the metadata. The fields must exist in the source and target metadata. The function of the

primary key is to determine whether a mismatch, or conflict of records, requires an update to a record, the insertion of a new record, or a record deletion.

The following figure shows an example of a Primary Key replication between a master and a nonmaster database.



One of the databases identified in the Activity document is the master database. It contains the data that wins any replication conflict.

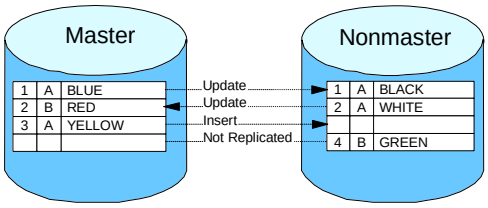
When replication takes place, records are updated according to the following rules:

- Record is identical in both databases: The record is not updated.
- Record does not match but the primary key is identical: The record in the master database updates the record in the nonmaster database.
- A primary key value exists in the master database but not in the nonmaster: The record from the master database is inserted in the nonmaster database.
- A primary key value exists in the nonmaster database but not in the master: The record from the nonmaster database is deleted.

Note that the option of mismatching keys does not exist. Such a case is treated as an insertion and a deletion. Because no information beyond the primary key is available, all mismatches are regarded as conflicts. In the case of conflicts, the master database overwrites the nonmaster database, and the master always prevails in a primary-key-only mismatch.

Primary Key/Timestamp Replication

If the metadata involved in the replication contains time-stamp fields, and those fields are identified in the replication activity, LEI performs Primary Key/Timestamp replication. The following figure shows an example of a Primary Key/Timestamp replication.



First LEI creates two result sets (one for each database) that include only those records that have changed since the last replication based on the time stamp field. LEI then matches primary keys:

- Both master and nonmaster records are found: The master record updates the nonmaster.
- Only one record is found (either master or nonmaster): LEI attempts to locate the primary key in the other database.
  - If the primary key in the other database is found, the changed record updates the record that was not changed.
  - If the primary key in the other database is not found, a copy of the changed record is inserted into the other database.

In Primary Key/Timestamp replication, the nonmaster database can update the master database if a change has occurred only in the nonmaster database record. If a corresponding record is changed in both the master and nonmaster databases, a replication conflict will ensue.

Because LEI looks only at changed records, deletions are not replicated. If a record is deleted from one database, the corresponding record in the other database remains unchanged. To handle issues like this, you sometimes have to schedule a Primary Key replication on a less frequent basis to complement the Primary Key/Timestamp replication.

Primary Key/Timestamp replication looks only at the records where the time-stamp field value is later than the last time the activity was run. Primary Key/Timestamp replication remembers the latest time stamp on both servers, so the servers can have different time and date settings without any problem.

Some database systems use different levels of precision for date and time fields. For example, one system may store date/time values to the nearest second, whereas another system may store date/time values to the nearest millisecond. To carry out time-stamp replication options between these

systems, to produce an accurate comparison LEI may have to reduce the precision of the date and time fields when comparing values.

Java Activity

A Java Activity allows LEI to launch an enterprise integration Java application. Specifically, this would be a Java application which imports and uses the LEI Java Classes shipped with LEI. As with LotusScript and the accompanying Lotus Connector LotusScript Extensions (LSX), the Lotus Connector Java classes expose the entire Lotus Connector Application Programming Interface. This allows you to extend the basic functionality provided by the form-based, or declarative, activities found in the LEI Administrator.

An enterprise integration Java application can be developed in any Java IDE, and must run on a JDK 1.1 compliant JVM. As with any LEI application, LEI and Notes must also be installed on the same machine that will run the application. The Java Activity simply provides the means by which an LEI Java application may be scheduled and executed along with other LEI activities.

Use the Java Activity any time you wish to use LEI to launch a Java application. The Java application can be anything you wish, and in fact need not be restricted to enterprise integration specific applications. Generally, however, the Java application will use the Lotus Connector Java classes shipped with LEI, which expose the entire Lotus Connector Application Programming Interface. If you are already familiar with the Scripted Activity using LotusScript and the Lotus Connector LSX, this is the same concept.

Note Due to current platform limitations, AIX and HP-UX do not support the Lotus Connector Java classes.

Scripted Activity

A Scripted Activity is one that executes LotusScript commands. LEI extends the set of Domino classes available to LotusScript. The Lotus Connector LSX, which comes with Notes/Domino 4.63 and later, is also included with LEI. LEI extends these classes with additional methods to support the LEI Administrator and log databases. Using Lotus Connector LotusScript classes allows you to extend the functionality available from the other LEI Administrator Activity operations (Direct Transfer, Polling, Replication, and so on). You can also construct customized routines in situations where you need more complete control over source and destination data transfers, or wish to instruct the LEI engine to perform data massaging or evaluation during transfer.

Constructing an Activity

To create an activity follow these steps:

1. Open the LEI Administrator Database.
2. Select Create - Activity from the Menu Bar.
3. Choose the activity you would like to construct. The selected activity document will be opened in edit mode.
4. Each type of activity document is different. Refer to the LEI online documentation for specific required fields for the activity document you are constructing.

Using Action Buttons to Construct Activities

Action buttons can be very helpful in the construction of an Activity document.

While in edit mode, an LEI Activity document may include the following action buttons:

- Author Privileges
- Select Metadata
- Map Fields

Note If you receive the error

```
Error loading USE or USELSX module: *leilsx
```

when you click one of the browsing-related buttons (Select Metadata, Map Fields, or Map Timestamp), you either do not have the LEI Client or Server installed on the machine from which you are trying to run the Browsing, or the LEI directory is not in your path.

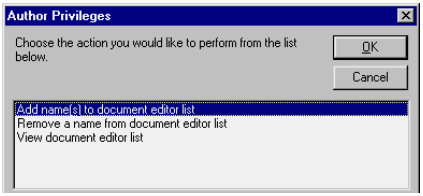
Author Privileges

The Author Privileges action button, shown in the following figure, allows you to change author privileges for the activity currently being edited.

To change Author Privileges for the activity:

1. Select the Author Privileges action button. The Author Privileges dialog box appears.

2. Select the action you want from the options available in this dialog box.



These actions include:

- Add name(s) to document editor list
- Remove a name from document editor list
- View document editor list

Select Metadata

The Select Metadata action button, shown in the next figure, lets you display available metadata objects from which you can pick the Activity's metadata when building Direct Transfer, DPROPR, Polling, and Replication Activities. With Direct Transfer and Polling Activities, the Select Metadata Action lets you select fields from the source metadata for simple command statement building. With DPROPR and Replication Activities, the Select Metadata Action lets you select key fields.

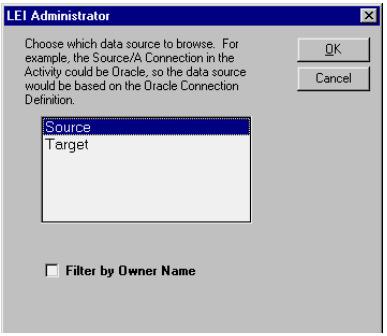
If the Activity has two connections, run the Select Metadata for each connector.

To use the Select Metadata action button:

1. Click the Select Metadata action button. The Connection Selection dialog box appears. You may see a metadata type option depending on the Activity type. The following table shows what you can choose between for the different acticity types:

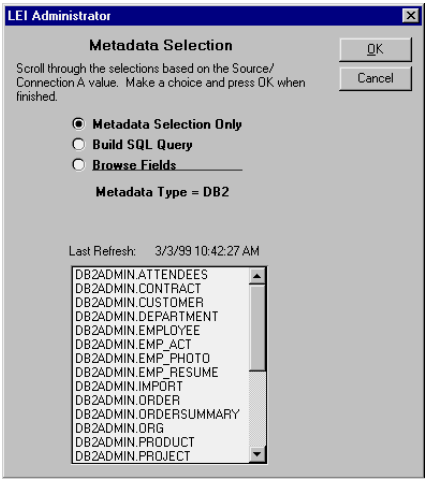
Activity Type	*Click*
Direct Transfer DPROPR	Source or Target depending on which Connection you want to browse
Replication	Connection A or Connection B
Polling	No option. Polling uses a single Connection

For the Direct Transfer acticity the Connection Selection dialog box looks like this:



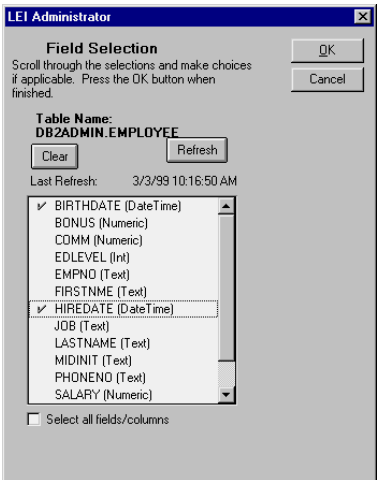
Note More information on the checkbox, Filter by Owner Name can be found in the section "Filter by Owner Name Option" later in this chapter.

2. The Metadata Selection dialog box appears. If you are working with the Source, choose either the Metadata Selection Only (which is the default), Build SQL Query or Browse Fields. A description of each option follows.



- Metadata Selection Only — Only lets you choose from a metadata list based on the Connection selected.
- Build SQL Query — Lets you create a SQL command statement by selecting fields.
- Browse Fields — Will not build or replace any existing command statement. (The Build SQL Query and Browse Fields options are only available in Direct Transfer and Polling Activities).

3. Click OK. If you are working with the Destination, the metadata is entered into the form and the action is completed. If you are working with the source, you will see the field list.
4. The Field Selection dialog is shown in the following figure. The field selection options depend on the kind of Activity that you are building:

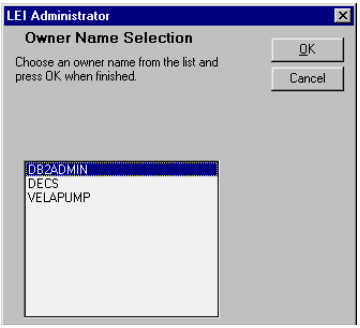


5. Direct Transfer and Polling: If you picked the build option in step 2, choose the fields, or columns, that you want to appear in the command statement. These will be the fields that LEI will *select* for transfer or evaluate when polling. Check Select All if you want to choose all fields in the metadata. LEI does not build a command line through Select Metadata for a Notes Source.
6. For DPROPR and Replication Activities, choose the key field or fields.
7. Click OK to complete the action.
8. Save the activity. The selected metadata and fields are not automatically saved. You must save the document in order to save your metadata and field selections.

Filter by Owner Name Option

This option allows users to view only those tables owned by a certain owner. It only applies to those data sources that support the concept of owner names. This excludes Domino/Notes, text and File System Connectors.

If this option is checked, the next dialog will contain a list of owners found based on the information provided by the connection selected. After an owner is chosen, the list of tables owned by that owner is displayed.



If a data source that does not support owner names is chosen, an informational message to that effect appears.

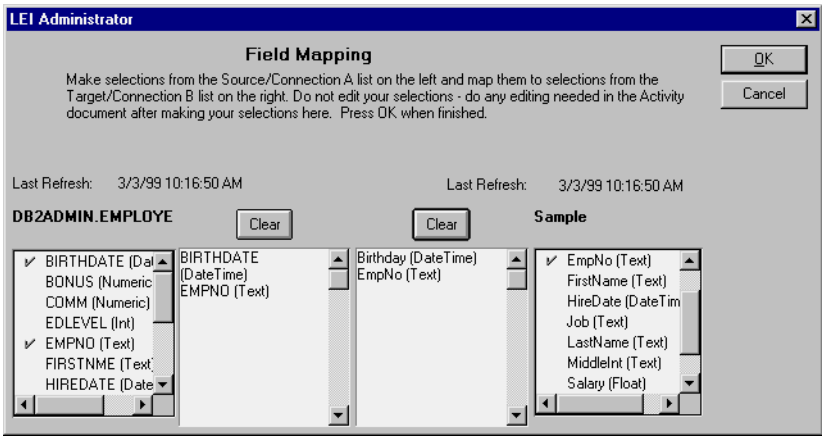
Owner names are cached in the same manner as the lists of tables and fields. The same *refresh* button and last refresh datetime appear in the table and field dialogs. Additionally, the owner name filtering checkbox will reflect the correct setting as cached and used previously. This means that if you chose previously to use owner name filtering on the Source Connection but not on the Target Connection; then as you move between the two Connections in the dialog box the next time, the setting will appear as you last designated — on for Source and off for Target. This is true in all cases, except if owner name filtering was chosen for a connector that doesn't support it. In that case, the setting is turned off for the user since it is invalid.

If a connection previously had owner name filtering turned on, it may be turned off and the entire list of tables for that connection will be displayed. This means that a connection to the data source is made, new table data is gathered and the cache is rewritten.

Note that the menu item Actions — Purge Cache can be applied to destroy the current cache and start over.

Map Fields

The Map Fields action button, shown in the next figure, displays fields in the source and destination metadata. By selecting a list of fields from both metadata objects, you can build a field list that the activity can use to map fields in moving data from one database to the other. Map Fields is available for Direct Transfer, Polling, Replication, Archive, DPropR and RealTime Activities.



To use the Map Fields action button:

1. After specifying both Activity Connections, click the Map Fields button. (If the form contains no metadata or command line, Map Fields steps you through the Select Metadata steps described under the topic "Select Metadata".)
2. In the lists of fields in source and target (or Connection A and B) metadata, make a one-to-one selection of corresponding fields. In other words, the first field on the source selected list will map to the first field on the destination selected list, the second to the second, and so on.
3. If you make an error, click the Clear button and start over, or remove the check from the unwanted item in the master list selection. To update the cache for the selected metadata, click the corresponding refresh button in the dialog box. This will cause LEI to read the data from the defined Connection and update its cached information.
4. When done mapping fields, click OK.
5. If the option Map Fields as User Defines is not selected on the activity form, you will be asked if you want to select it.
6. Save the activity document. The selected fields are not automatically saved. You must save the document in order to save your field mapping selections.

Map Timestamps (Replication Activity)

If data changes both in the external data source and in Domino, and you want to have the changes synchonized both ways, you must select the Timestamp Replication checkbox in edit mode in the Activity document.

This will make the Map Timestamp action button appear.

Clicking the Map Timestamp action button displays fields in the source and destination metadata of a Replication Activity. You can select the field that can be used as the record timestamp. This is the field containing the modification date of the record, which will be used to determine which record takes precedence in Timestamp replication.

To use the Map Timestamp Action Button:

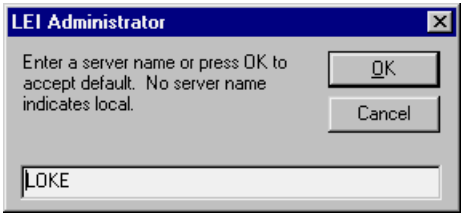
1. After selecting the Activity Connections and metadata, mark the checkbox for Timestamp Replication. The Timestamp Action button will appear. Click the Timestamp Action Button.
2. Select one timestamp field from each list of fields in Connection A and B metadata. If you make an error, just choose a different field.
3. When done, click OK.
4. Save the activity document. The selected fields are not automatically saved. You must save the document in order to save your timestamp selections.

Create Agent (Script Activity)

The Create Agent action button, shown in the next figure, appears in the LEI Administrator document when you select Create - Activity - Scripted from the menu bar.

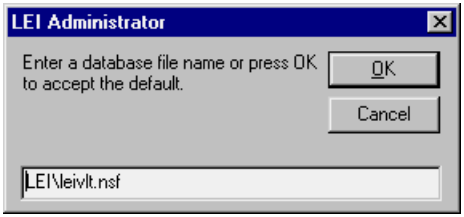
To use the Create Agent Action Button:

1. Click the Create Agent action button. The following screen appears, prompting you to enter a server name for the location of the Scripted Activity.

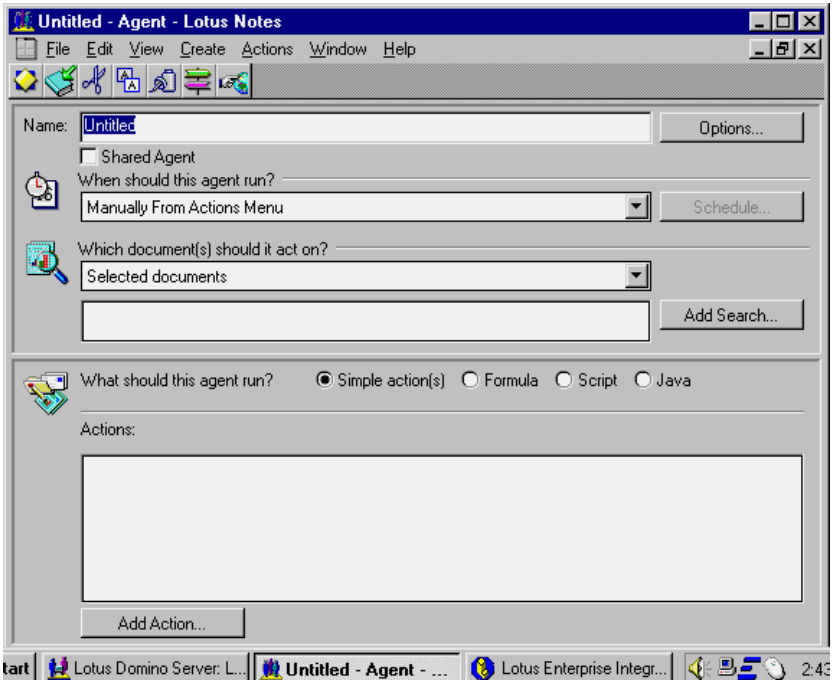


2. Enter a server name for the location of the Scripted Activity. To use the default server, click OK.

3. After specifying the Server or accepting the default server, the following screen appears, prompting you to specify the database containing the Scripted Activity.



4. Specify the database containing the Scripted Activity and click OK.
5. After specifying the database, the LotusScript Agent Development document window appears, as shown in the following figure. Use this document to develop the Scripted Activity.



Running Activities

There are two methods to run an activity. Activities can be run on a schedule, or they can be run on demand using an agent known as Run ASAP!.

Running an Activity with Run ASAP!

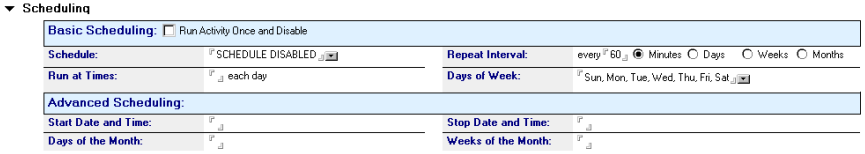
The Run ASAP! button, shown in the next figure , is visible while in view mode of an Activity document. If clicked, it causes the Activity to be run the next time the LEI server polls the LEI Administrator database for pending activities.

Click the Run ASAP! button when you want to run an Activity as soon as possible. It can be used even if an activity is set up to be run on a schedule.

Alternately, the Run ASAP! agent can be invoked by selecting the Activity document from the Overview, Activities, or Categories views in the LEI Administrator database, and selecting Actions - RunASAP! from the menu bar.

Running an Activity on Schedule

Any Activity can be set up to run on a Schedule. There is a Scheduling section in each Activity document, shown in the following figure, that you can fill in to best suit your needs.



Basic scheduling options include:

- Schedule — Enable or Disable Schedule
- Run at Times — A given hour or range of hours at which to run this Activity
- Repeat Interval — How often to repeat this schedule
- Days of Week — Days on which this schedule should run

Advanced scheduling options include:

- Start Date and Time — Activity will NOT be scheduled to run prior to this date and time
- Stop Date and Time — Activity will NOT be scheduled to run after this date and time
- Days of the Month — Numerically specify days of the month to run this schedule
- Weeks of the Month — Numerically specify weeks of the month to run this schedule

LEI LotusScript Extensions and Java Classes

LC LSX Extensions with LEI

The following LC classes have expanded properties when LEI is installed:

- LCConnection Class
- LCFieldlist Class
- LCSession Class

The LCSession class also provides additional methods to support activity level properties and logging.

When an LEI Scripted Activity runs, the script can access the property fields of the activity form. When a Scripted Activity is used in conjunction with the LEI CGI program and an HTML form, the form's field values appear in the session object as read-only properties.

See the chapter "Connector Programming Interface" for additional details about the LCSession Class methods.

Java Classes

The Lotus Domino Connector Java Class Library, available currently with the Lotus NotesPump 2.5 product, provides a set of Java Classes which, in conjunction with the Lotus Connector LSX, can be used to develop a Java Server program to control access to Lotus Connector data sources.

Future versions of the Domino Server (post R5.0) will also include the Lotus Connector Java Class Library. The LC Java Class Library will continue to be provided with the NotesPump product.

LEI Examples

This section presents scenarios in which LEI is used to satisfy various business requirements.

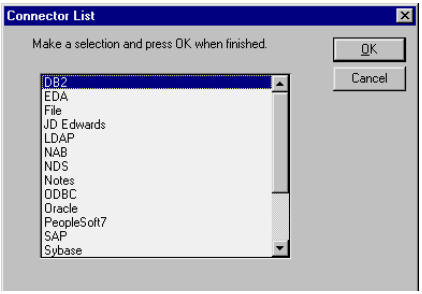
Direct Transfer and RealTime Access to Relational Data

Requirement: Provide real time access to DB2 from a Web browser.

Solution: LEI can be used to access data real time. The steps to complete this activity are:

1. Construct a DB2 Connector.

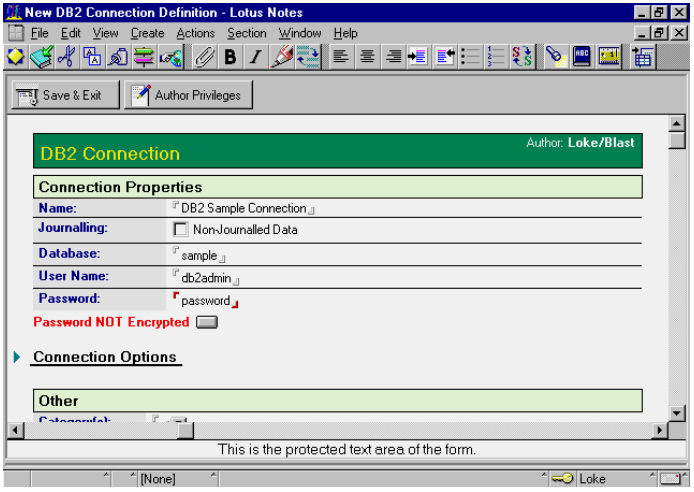
   From the LEI Administrator Database click Create Connection from the LEI Administrator Navigator. The following menu will appear.



   Select DB2 to continue.

   A new DB2 connection document will be opened.

   Choose a name for your connection. Make it something meaningful since this is how you will be referring to it when constructing activities.



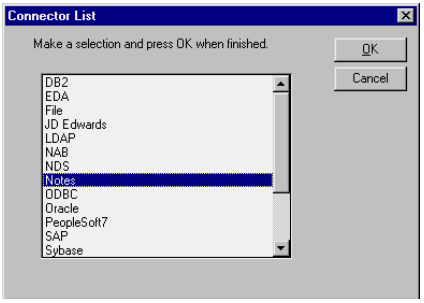
Next fill in the DB2 database name, user name, and password. These are the minimum requirements for a DB2 link. There is an checkbox option available for Non Journalled data. Select this option to enable support for SQL queries against data in non-journalled tables. You can read more about data journaling in “Creating Connections with the wizard” the DECS chapter.

Another optional section is Connection Options, which contains DB2-specific options for this connection. For more information on how to use these options, refer to the LEI Users Guide.
Click the Save & Exit action button when you have completed the connection document.

Note The password you enter in the connections is not encrypted by default. You can click on the button next to the string Password NOT Encrypted on the connection document, if you wish to specify an encryption key to use for encrypting to the password.

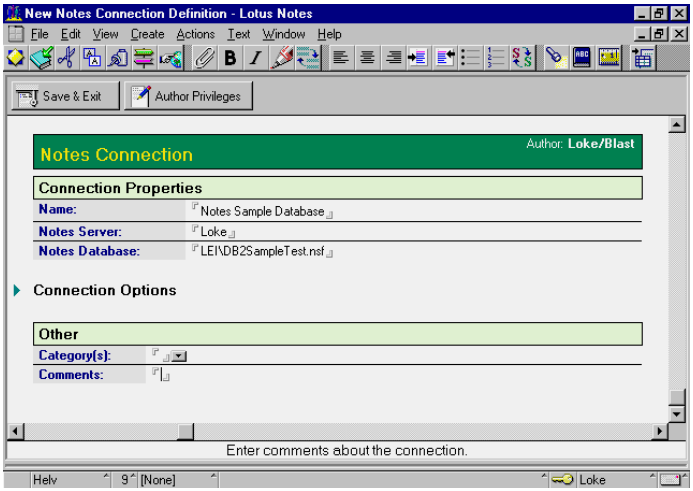
2. Construct a Notes Connector.

   From the LEI Administrator Database, click Create Connection from the LEI Administrator Navigator. The following menu will appear.



   Select Notes to continue.

   A new Notes connection document will be opened. Choose a name for your connection. Make it something meaningful since this is how you will be referring to it when constructing activities.



Type the name of the Notes server where the database you want to connect to is located, and the database name, with the path relative to the Notes data directory. These are the minimum requirements for a Notes link.

There is also an optional section of Connection Options for extending the capabilities of LEI to take advantage of specific Domino data structures and design elements. The options let you specify, for example, view-based selection criteria that manage the hierarchical structure of Notes. Connection Options also permit you to define data translation and manipulation. For more information on how to use these options, refer to the LEI Users Guide.

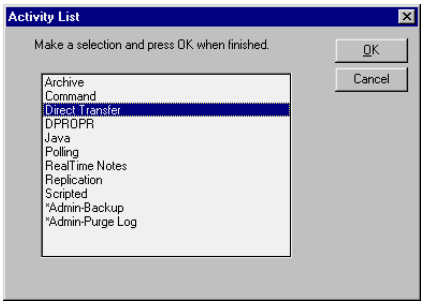
Click the Save & Exit action button when you have completed the connection document.

3. Test both connection documents.

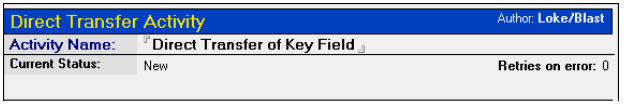
   To avoid problems during the running of an activity, it is a good habit to test your connection documents before continuing. Use the LCTEST program that is included with LEI. It will be in the same subdirectory where the LEI executable is located. See the section “Testing Connectors” earlier in this chapter.

4. Construct Direct Transfer Activity.

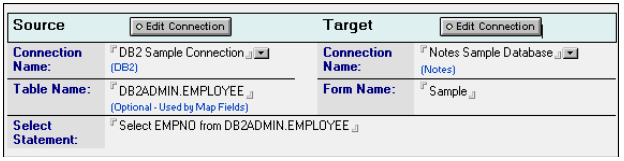
   To complete a Real Time Activity, at least one key field must be resident in Domino. Therefore, you will need to create and run a Direct Transfer Activity to transfer the content of key fields from DB2 to Domino. From the LEI Administrator Navigator, click Create Activity. The following menu will appear.



Select Direct Transfer. A new direct transfer document will be opened. Choose a name for your activity.



Under the Source section, click the pull down box next to Connection Name. You will see a box containing all previously defined connectors. Click the DB2 connector you constructed in step 1. Under the Target Section, click the pull down box next to Connection Name and select the Notes connector you constructed in step 2. If you are working at the server or at an LEI Development Client, you will have access to the Select Metadata and Map Fields action buttons. Details on using these buttons can be found in the section "Using Action Buttons to Create Activities" earlier in this chapter. Use the Metadata buttons to complete the Source and Target sections. If you do not have access to the action buttons, you will need to know beforehand the DB2 table name, target form name, and DB2 select statement for the field you will be transferring.

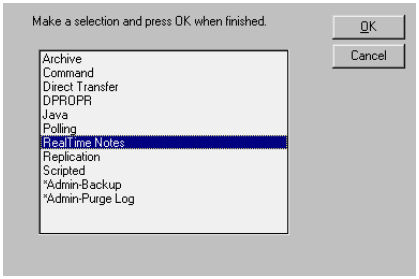


Next, complete the Field Mapping Section. This section gives you the option of selecting Automatic field mapping. Clicking this box will prompt you to choose between mapping by field name or field position. However, for our example we will be using the Map Fields Action button to fill out the Source and Target Field Lists. These fields should contain the names of the DB2 key field and the field where Notes will hold the key field once it is transferred.

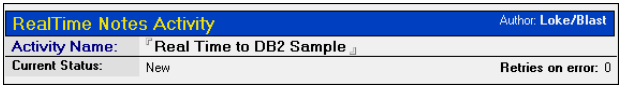
This completes the required information for the transfer of a key field from DB2 to Notes. There are other options that could be used in a Direct Transfer Activity. For more information on these options, see the LEI User's Guide.

5. Construct Real Time Activity.

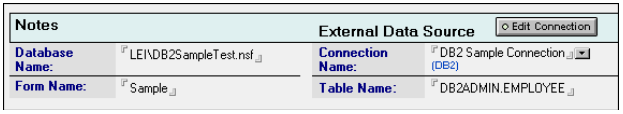
   From the LEI Administrator Navigator, click Create Activity. The following menu will appear.



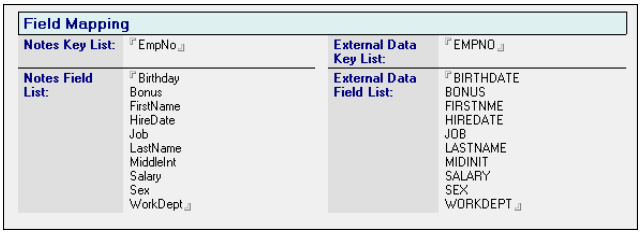
   Select RealTime Notes. You will now be in a new Real Time Activity document. Choose a name for your activity.



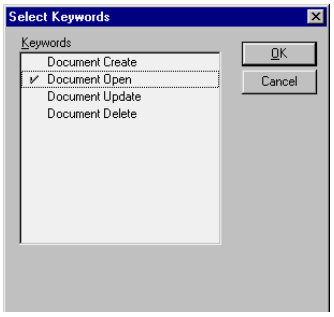
   In the Notes section, use the Select Metadata buttons to fill in the name of the Notes database where users will be accessing the real-time data. It should be the same one that was specified in the Target section of the Direct Transfer Activity Document. The Metadata button will also prompt you to fill in the Notes Form Name and Notes Key List. This should be the field that was transferred in the Direct Transfer. In the External Data Source section, use the pull-down box to select the DB2 connection you previously constructed. You will then need to fill in a Table Name. You can use the Metadata button to select the table and keyfield to be used.

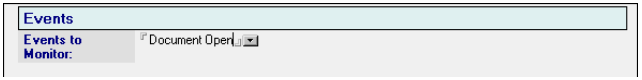


   Continue with the Field Mapping section. The Key List fields have already been filled in with the Metadata button. The Notes Field List and External Data Field List should contain a list of corresponding fields between DB2 and Notes. List all the fields that will be viewed from Notes. Use the Map Fields Action button to complete these fields. If you do not have access to the action buttons for any of these fields, you can manually type in the appropriate information.



Clicking on the pull down box in the Events to Monitor section gives you four options. For our example, we are going to assume the data only has to be viewed via a Web browser. Therefore, we will only select Document Open. More detail on these options can be found in the User's Guide.





This completes the required information for the Real Time Activity document.

6. Run the Direct Transfer activity by selecting it from the Activities view in the LEI Administrator Navigator and then clicking Action - RunASAP from the menu bar. If there are any errors, open the Direct Transfer activity and use the log link to view the error for that activity. Correct the error before continuing.
7. Run the Real Time activity by selecting it from the Activities view in the LEI Administrator Navigator and then clicking Action - RunASAP from the menu bar.
8. Make sure your Domino HTTP task is running. You will now be able to view this DB2 data real-time via a Web browser

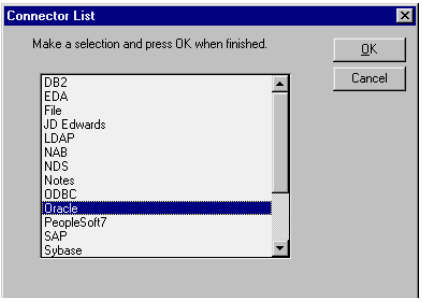
Replication to Relational Data

Requirement: There are two Oracle tables containing employee and customer data. You are required to combine each employee with the data for customers they support. There is a Rep ID in the customer table that corresponds to the Employee ID in the employee table. Combine selected data from these two Oracle tables into a single Domino database. This should be updated hourly during business hours, Monday through Friday. Also, keep the Domino data in synch with the Oracle database. Oracle will be the master database.

Solution: LEI can be set up to perform a scheduled replication hourly from Oracle to Domino. You will need two separate replication activities going to the same Oracle database, but they will both be referencing the same Oracle Connection Document. The steps to complete this activity are:

1. Construct an Oracle connection.

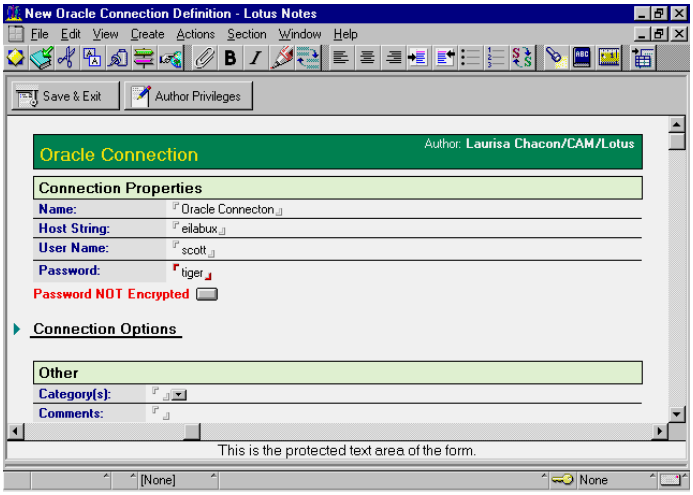
   From the LEI Administrator Database click Create Connection from the LEI Administrator Navigator. The following menu will appear.



   Select Oracle to continue.

   A new Oracle connection document will be opened.

   Choose a name for your connection. Make it something meaningful since this is how you will be referring to it when constructing activities.

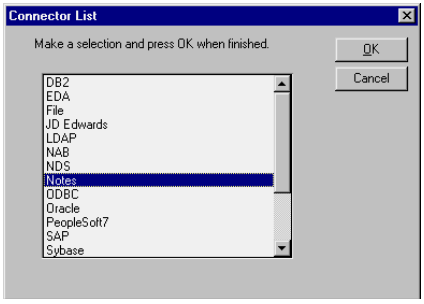


Fill in the host string, user name, and password. You can click the button next to Password NOT Encrypted, if you wish to apply encryption keys to the password. These are the minimum requirements for an Oracle link.

The optional section entitled Connection Options contains Oracle-specific options for this connection. For more information on how to use these options, refer to the LEI User's Guide. Click the Save & Exit action button when you have completed the connection document.

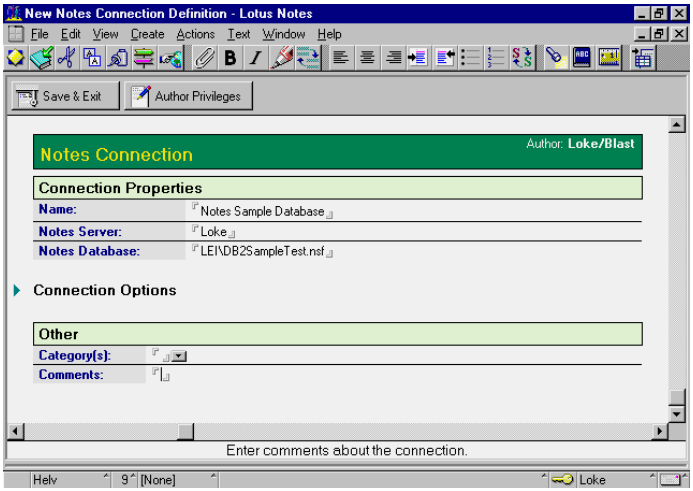
2. Construct a Notes connection.

From the LEI Administrator Database click Create Connection from the LEI Administrator Navigator. The following menu will appear.



Select Notes to continue.

A new Notes connection document will be opened. Choose a name for your connection. Make it something meaningful since this is how you will be referring to it when constructing activities.



Fill in the Domino server where the database you are trying to get at is located, and the database name, with the path relative to the Notes/Domino data directory. These are the minimum requirements for a Notes link.

There is also an optional section of Connection Options for extending the capabilities of LEI to take advantage of specific Domino data structures and design elements. The options let you specify, for example, view-based selection criteria that manage the hierarchical structure of Notes. Connection Options also permit you to define data translation and manipulation. For more information on how to use these options, refer to the LEI User's Guide.

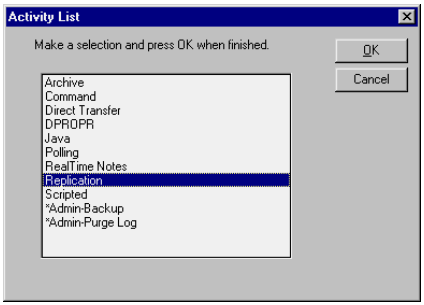
Click the Save & Exit action button when you have completed the connection document.

3. Test both connection documents.

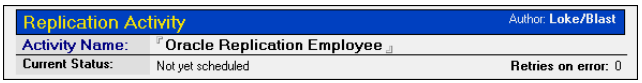
   To avoid problems during the running of an activity, it is a good habit to test your connection documents before continuing. Use the program LCTEST.EXE that is included with LEI. It will be in the same subdirectory where the LEI executable is located. See the section "Testing Connectors" earlier in this chapter.

4. Construct a replication activity to Employee Table.

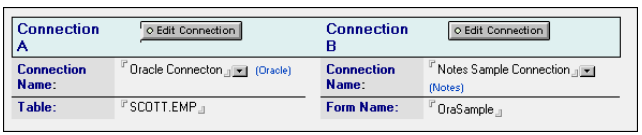
   From the LEI Administrator Navigator, click on Create Activity. The following menu will appear.



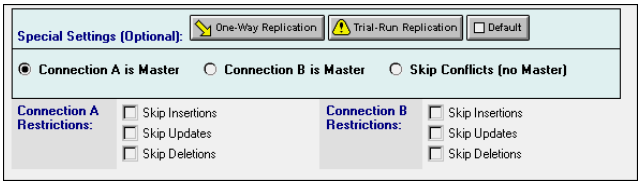
   Select Replication. A new replication activity document will be opened. Choose a name for your activity.



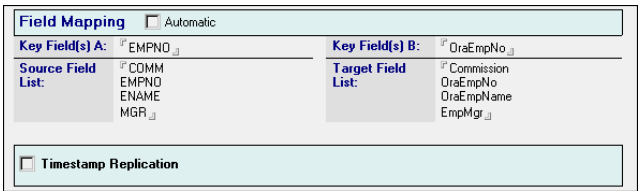
   Next, use the pull down boxes next to the Connection Name fields for Connection A and B. It doesn't matter which connection is A or B since a separate field specifies whether A or B will be the master. For our example, Oracle will be Connection A and Notes will be Connection B. To complete the Connection A table field and the Connection B form name field, use the Select Metadata action button. Since this replication activity is getting the Employee data, we will select SCOTT.EMP. The Domino form to be used is OraSample. You will also be prompted to select a keyfield for each Connection.



   The next section shows optional Special Settings. We won't be using these for this activity. The only selection that must be made is to make sure Connection A, Oracle is selected as the master. The Restriction checkboxes are available if you don't want insertions, updates, or deletions to be performed on the corresponding database. For our example, leave these unmarked.



The Field Mapping section allows you the option of automatic field mapping by field name or field position. Since we don't know the exact order or names of the data in Oracle, we shouldn't select this option. The key fields have already been filled in with the metadata button. This is the key that the activity will use to compare which data should be updated in Domino. To complete the Source Field List and Target Field List fields, use the Map Fields action buttons. The fields must correspond in the appropriate order or you may end up with data in the wrong fields.

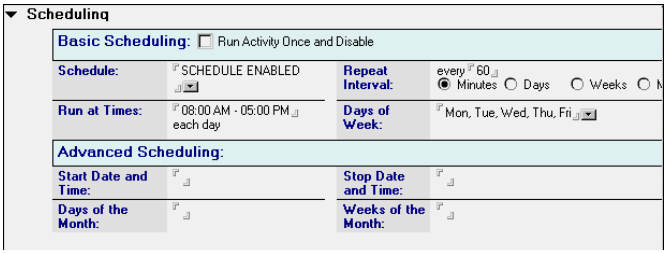


Since this activity must be run hourly from 8:00 AM to 5:00 PM, Monday through Friday, we will set up the scheduling. There are a few guidelines for how LEI evaluates schedules.

- The first parameter of scheduling is the "DISABLED/ENABLED/ RESTRICT" setting. When an activity is disabled, all of the other scheduling settings are ignored. When enabled, the activity will be scheduled to run according to the setting of the remaining parameters. If the activity should not run except at the scheduled times, the "RESTRICT TO SCHEDULE" should be used. Our activity will be set to SCHEDULE ENABLED.
- The next parameters to affect the scheduled run time are Start Date and Time and Stop Date and Time. These parameters create boundaries for the run time. Scheduling will not begin before the start date and will not continue after the stop date. These parameters don't apply to our activity since we want to enable replication immediately and let it run for an indefinite period of time.
- Once the boundaries of scheduling are established, the Days of Week, Weeks of the Month, and Days of the Month parameters are used together to determine which dates in the month are acceptable. Any dates which do not fall on the selected Days of the Week, Days of the Month, or Weeks of the Month are ignored. This feature is helpful for

scheduling activities to run on the last day of the month or even the last Friday of the month. Since our activity should run Monday through Friday, we will select only these days in the Days of the Week field.

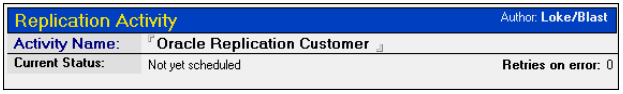
- The final parameters for scheduling are Repeat Interval and Run at Times. When specified, any times which fall outside the specified Run at Times values are ignored. The values may be any combination of distinct times such as "8:00AM, 11:30PM" and time ranges such as "7:00AM – 3:00PM". Once the initial run time has been computed, subsequent run times are scheduled at the repeat interval. Our requirements state the activity should run hourly from 8:00 AM to 5:00 PM. Put this time range in the "Run at Times" field and set the "Repeat Interval" to 60 minutes.



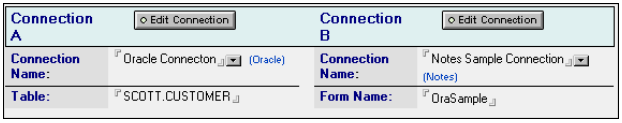
Complete the scheduling section and save and exit the document.

5. Construct a replication activity to Customer Table.

Follow the same steps used in step 3 for constructing the Employee replication activity. There are a few differences to be noted. First of all, the activity will need a different name.

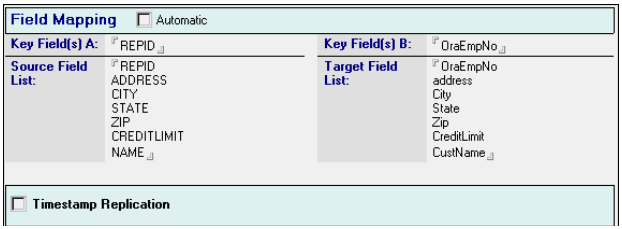


Secondly, since we are going to get data from a different table, we will need to fill in the name of the customer table, SCOTT.CUSTOMER.



Leave Connection A, Oracle as the master.

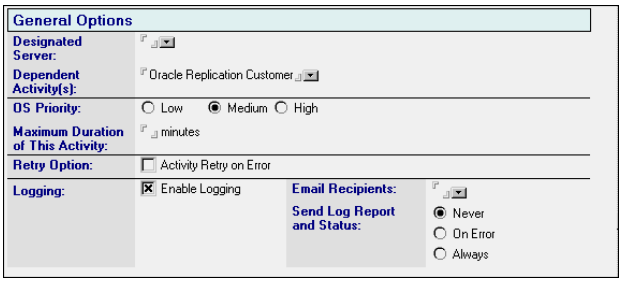
The key field and field mappings will be different, since we are accessing a different table. The field RepId in the customer table will be used as the key field and it will be matched up with OraEmpNo in the Domino database. The rest of the customer data will be mapped over in the Source and Target Field Lists. You can use the Map Fields action button, as previously described.



Once you have completed this information, close and save your document.

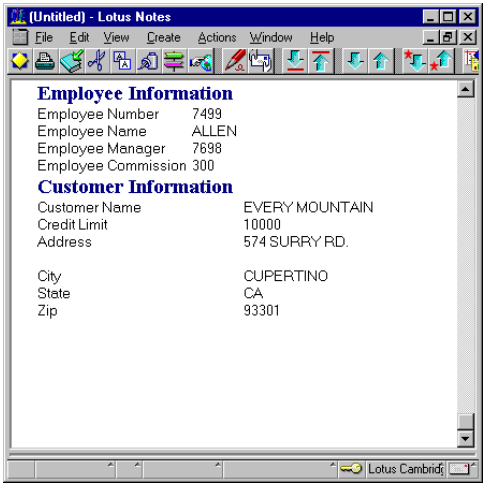
6. Setup Oracle Replicate Customer as a dependent activity.

It is not necessary to set up the schedule for the Customer activity. We will use the schedule set up in the Oracle Replicate Employee Activity to run this activity also, by designating the Customer activity as a dependent activity of the Employee Activity.



To do this, open the Employee Activity for edit. In the section General Options, click the pull down box next to the field Dependent Activity(s). You will see a list of all defined activities in this LEI Administrator Database. Select Oracle Replication Customer. Now, after each time the Employee Activity completes, the Replication Activity will start.

7. After completion of the first scheduled hourly replication, open your target Domino database and verify that the information has been transferred correctly.



Direct Transfer and RealTime Access to SAP

Requirement: Provide real-time access to SAP from a Web browser.

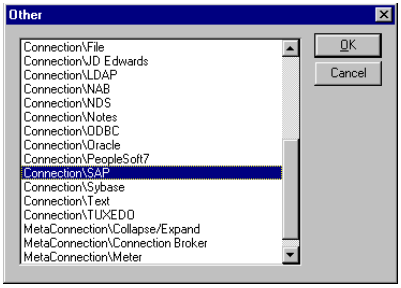
Solution: LEI can be used to access data in real time. The steps to complete this activity are:

1. Construct an SAP Connector.

   Since SAP is an optional connector, it will not appear as a choice from the LEI Navigator. If you do not have the SAP Connector, it can be purchased separately from Lotus. For more information, contact your local Lotus office or see

   `http://www.lotus.com/enterpriseintegration`

   From the LEI Administrator Database select Create - Connection - Other. The following menu will appear.

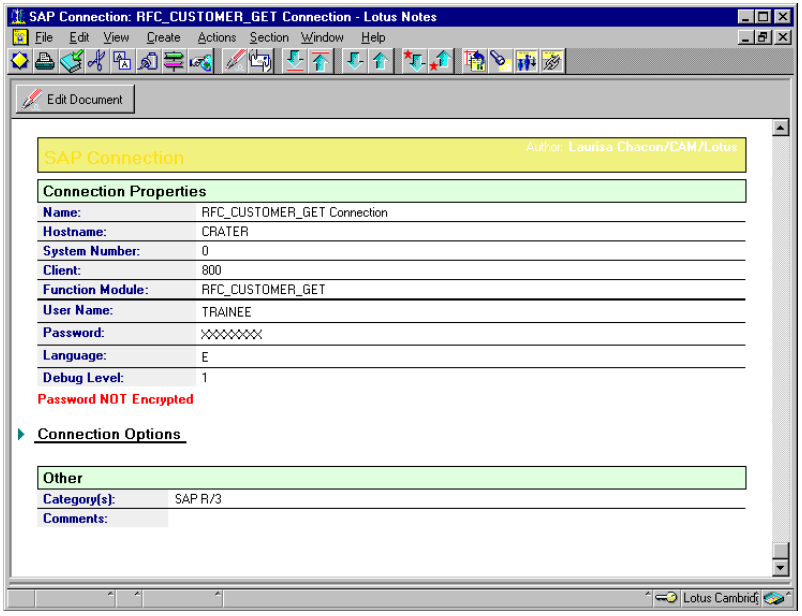


   Select Connection\SAP to continue.

You will enter a new SAP connection document. The following table details the information required in each field.

Field	*Description*
Name	Name to identify this Connection.
Hostname	Name of the SAP application server you want to connect to. This can be a hostname defined in a hosts file, an IP address, or an SAP router address.
System Number	R/3 system number used for log-on, for example, “00”.
Client	Three-digit client number string for system log-on, for example, “800.”
Function Module	Name of the RFC or BAPI that you wish to call.
User Name	R/3 log-on user name.
Password	Password associated with the specified user name.
Language	Language to be used for log on, for example, “E” for English.
Debug Level	Increases the logging in the LEI Log database.
Screen Input Field Definition (Connection Options)	Defines the screens of a transaction. Used only when an R/3 transaction is the target.

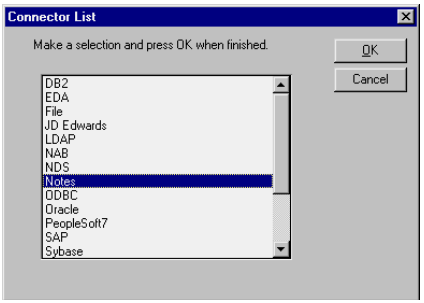
Fill in the appropriate information in your connection document, using the following form as an example.



Save and close your document.

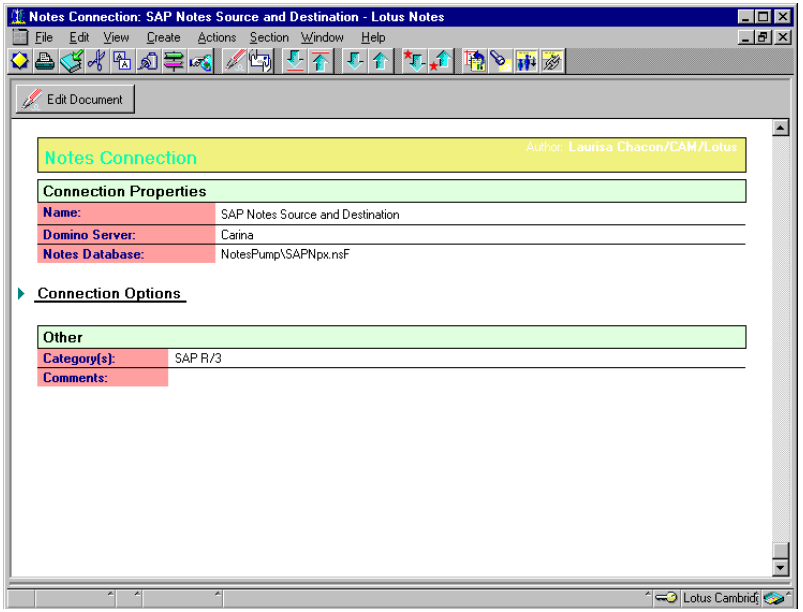
2. Construct a Notes Connector.

   From the LEI Administrator Database click Create Connection from the LEI Administrator Navigator. The following menu will appear:



   Select Notes to continue.

   A new Notes connection document will be opened. Choose a name for your connection. Make it something meaningful since this is how you will be referring to it when constructing activities.



   Fill in the Domino server where the database you are trying to get at is located, and the database name, with the path relative to the Notes/Domino data directory. These are the minimum requirements for a Notes link.

There is also an optional section of Connection Options for extending the capabilities of LEI to take advantage of specific Domino data structures and design elements. The options let you specify, for example, view-based selection criteria that manage the hierarchical structure of Notes. Connection Options also permit you to define data translation and manipulation. For more information on how to use these options, refer to the LEI User's Guide.

Click the Save & Exit action button when you have completed the connection document.

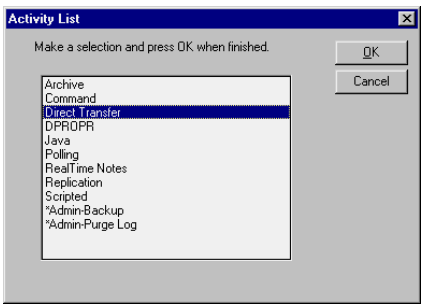
3. Test the Notes connection document.

   To avoid problems during the running of an activity, it is a good habit to test your connection document before continuing. Use the program LCTEST.EXE that is included with LEI. It will be in the same subdirectory where the LEI executable is located. See the section "Testing Connectors" earlier in this chapter.

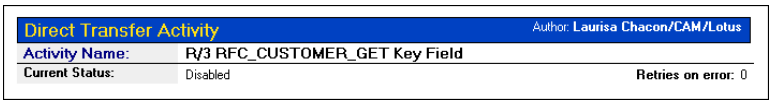
4. Construct Direct Transfer Activity.

   In order to complete a Real Time Activity, at least one key field must be resident in Domino. Therefore, you will need to create and run a Direct Transfer Activity to transfer a key field from SAP to Domino.

   From the LEI Administrator Navigator, click Create Activity. The following menu will appear:

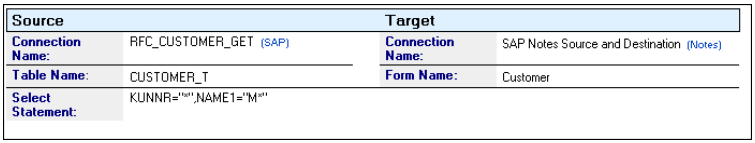


   Select Direct Transfer. A new direct transfer document will be opened. Choose a name for your activity.



   Under the Source section, click the pull down box next to Connection Name. You will see a box containing all previously defined connectors. Click on the SAP connector you constructed in step 1. Under the Target Section, click on the pull down box next to Connection Name and select the Notes connector you constructed in step 2. If you are working at the

server or at an LEI Development Client, you will have access to the Select Metadata and Map Fields action buttons. Details on using these buttons can be found in the section "Using Action Buttons to Create Activities" earlier in this chapter. Use the Metadata buttons to complete the Source and Target sections. If you don't have access to the action buttons, you will need to know beforehand the SAP table name, Notes form name, and SAP select statement for the field you will be transferring.

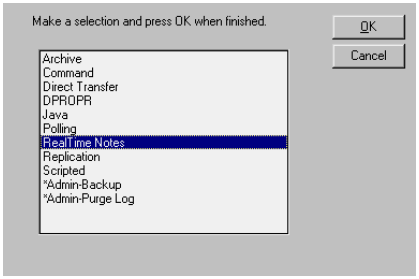


Next, complete the Field Mapping Section. This section gives you the option of selecting Automatic field mapping. Clicking this box will prompt you to choose between mapping by field name or field position. However, for our example we will be using the Map Fields Action button to fill out the Source and Target Field Lists. These fields should contain the names of the SAP key field and the field where Domino will hold the key field once it is transferred.

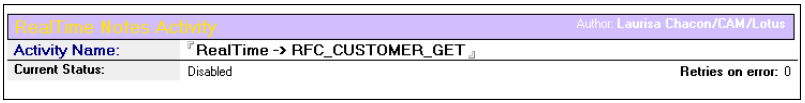
This completes the required information for the transfer of a key field from SAP to Domino. There are other options that could be used in a Direct Transfer Activity. For more information on these, see the LEI User's Guide.

5. Construct Real Time Activity.

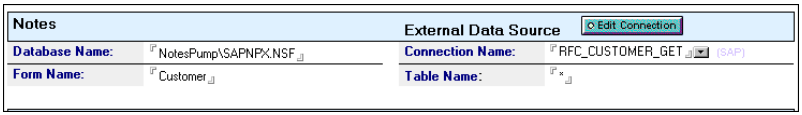
From the LEI Administrator Navigator, click Create Activity. The following menu will appear.



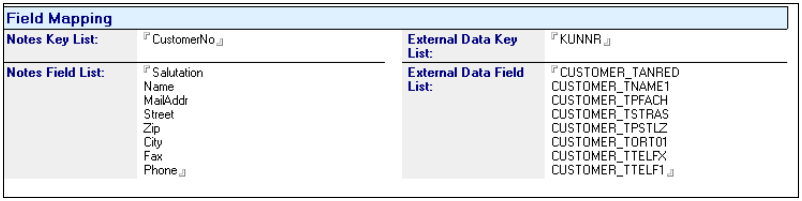
Select RealTime Notes. A new real time activity document will be opened. Choose a name for your activity.



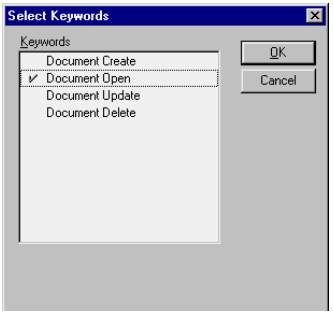
In the Notes section, use the Select Metadata buttons to fill in the name of the Domino database where users will be accessing the real time data. It should be the same one that was specified in the Target section of the Direct Transfer Activity Document. The Metadata button will also prompt you to fill in the Domino Form Name, and Domino Key List. This should be the field that was transferred in the Direct Transfer. In the External Data Source section, use the pull down box to select the SAP connection you previously constructed. You will then fill in a Table Name. You can use the Metadata button to select the table and keyfield to be used.

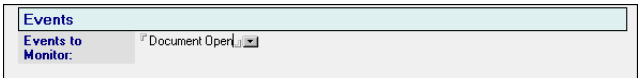


Continue with the Field Mapping section. The Key List fields have already been filled in with the Metadata button. The Domino Field List and External Data Field List should contain a list of corresponding fields between SAP and Domino. List all the fields that will be viewed from Domino. Use the Map Fields Action button to complete these fields. If you do not have access to the action buttons for any of these fields, you can manually type in the appropriate information.



Clicking the pull down box in the Events to Monitor section gives you four options. For our example, we are going to assume the data only has to be viewed via a Web browser. Therefore we will only select Document Open. More detail on these options can be found in the User's Guide.





This completes the required information for the Real Time Activity document.

6. Run the Direct Transfer activity by selecting it from the Activities view in the LEI Administrator Navigator and then clicking Action - RunASAP from the menu bar. If there are any errors, open the Direct Transfer activity and use the log link to view the error for that activity. Correct the error before continuing.
7. Run the Real Time activity by selecting it from the Activities view in the LEI Administrator Navigator and then clicking Action - RunASAP from the menu bar.
8. Make sure your Domino HTTP task is running. You will now be able to view this SAP data real time via a Web browser

Summary

In this chapter we have described the components of Lotus Enterprise Integrator (LEI) 3.0 and explained how to install and run LEI. We have explained how to set up connections and discussed the different activities supported by LEI. Finally, we have shown some examples of solutions with LEI.

Chapter 4
Connector Programming Interfaces

You can program the Lotus connectors using a LotusScript Extension (LSX) and Java classes. There is a Lotus Connector API available from Lotus that permits development of additional Lotus Connectors for Domino.

LotusScript Extension for Lotus Connectors

The LotusScript Extension for Lotus Connectors (LC LSX) extends use of the Lotus Domino Connectors to LotusScript. The programming model is independent of individual connectors, eliminating the need to learn each system, but allowing experienced users to access specific system features. Using this LSX, developers can access external system data natively, using the same object model to make calls syntactically to a variety of back end systems. This reduces the learning curve for developers seeking to integrate Domino with back end data sources.

Through the LC LSX, Notes and Web applications can retrieve and act upon data from agents, during document events, or by clicking a button.

The LC LSX can be used alone or in conjunction with Lotus Enterprise Integrator (LEI). Together these technologies provide programmatic and declarative access to external data for application development.

Software Requirements

Domino R5.0 provides the following standard set of connectors.

- DB2
- ODBC
- Oracle
- Sybase
- Domino Directory
- Lightweight Directory Access Protocol (LDAP)
- Novell Directory Services (NDS)
- File System
- EDA/SQL

- Notes
- Text

Additional connectors can be purchased separately. See the appendix "Lotus Connectors" for a list of available connectors and the supported platforms.

Access to specific connectors may require the installation of additional communication software on the Domino server or the Notes client. For example, for connectivity to Oracle, SQL*net needs to be installed on either Domino or the Notes client.

Terms and Concepts

The following terms have special meaning with respect to connectors.

Metadata

This is a generic term referring to a connector's data definition. The data definition includes the names of data elements and their data types, and implies the order of the elements. For example, Domino uses *forms* to describe both the names of data fields as well as the data type of each field, while Sybase metadata describes a *table*.

Note Do not confuse Metadata with MetaConnectors. Metadata is data about the stored data. MetaConnectors define some action to execute against data being transferred, sometimes resulting in data transformation as with the Collapse/ Expand MetaConnector and sometimes leaving the data unmodified as with the Billing MetaConnector.

Result Set

An information request through a connection returns a result set. Each connection can have a single active result set. You can have multiple simultaneous connections. The LCConnection methods Execute, Select, Call, and Catalog each produce a result set, replacing any existing result set. The result set describes the collection of data from the connection that matches the input criteria. Actual data is not returned until fetched. If specified, these methods build a fieldlist representing the metadata as part of generating the result set.

When using connection methods that read or write the data of a result set, the default order of the metadata may be suspended by using the connection's OrderByName property. Regardless of the order of the data within the metadata, OrderByName matches the names in the metadata to the names in the external system.

LC LSX Classes

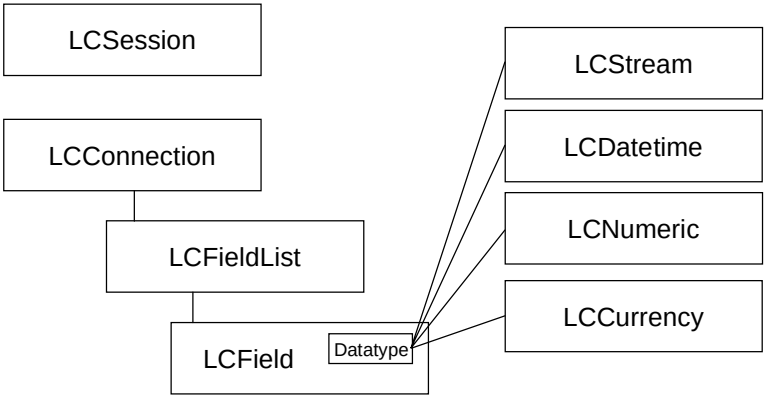
There are 4 LC LSX classes:

- LCSession
- LCConnection
- LCFieldlist
- LCField

In addition to the classes, there are 4 advanced data types:

- LCStream
- LCNumeric
- LCCurrency
- LCDatetime

The following figure shows the relationship between the different LC LSX classes:



LCSession

The LCSession class provides error information useful in error handling. It also provides for query and lookup of available connectors (see the first example). You do not need the LCSession class to connect to a data source, but it is recommended because it gives good error feedback.

The following tables list the LCSession properties and methods. See the LC LSX documentation for syntax and details.

LCSession Property	*Description*
Status	Status of an LCSession. Zero, or LCSUCCESS, indicates no error. A non-zero value (represented by an LCFAIL_XXX constant) indicates an error state.

LCSession Methods	*Description*
ClearStatus	Clears the Session status, putting the Session in a non-error state.
GetStatus	Obtains the current Session status.
GetStatusText	Obtains the error text string corresponding to a status code.
ListConnector	Lists all installed connectors available for LC LSX
ListMetaConnector	Lists all installed meta connectors available for an LC LSX installation.
LookupConnector	Determines if a specified connector is available.
LookupMetaConnector	Determines if a specified metaconnector is available.
Sleep	Suspends script execution for a specified period of time.

LCConnection

The LCConnection class represents an instance of a Lotus connector. This class provides query and data access to the external system. Multiple instances of an LCConnection class can be allocated to a single connector.

LCConnection properties depend on the specific connection. Typical properties include LCConnection.Database, LCConnection.UserID, and LCConnection.Password.

The following table lists the LCConnection methods. See the LC LSX documentation for syntax and details.

LCConnection Methods	*Description*
Action	Performs one of a set of predefined actions, such as reset, truncate, commit, rollback, clear.
Call	Performs a connector-independent procedure call with parameters.
Catalog	Produces a result set containing a metadata catalog.
Connect	Establishes a connection to a data provider.
Copy	Copies an existing connection.
Create	Creates a new metadata object.

continued

LCConnection Methods	*Description*
Disconnect	Disconnects from a data provider.
Drop	Drops an existing metadata object.
Execute	Executes a statement against the data source. The statement is provided in the data source's language.
Fetch	Retrieves records from the current result set.
GetProperty	Gets a property value for a connection.
GetProperty<datatype>	Gets a property as a particular data type.
Insert	Inserts new records into the data provider.
ListProperty	Lists supported properties and values.
LookupProperty	Verifies the support of a property.
Remove	Removes records from the data source. Key values are provided. No keys are required for writeback operations.
Select	Performs a connector-independent selection controlled by various connector properties. Conditional key inequalities, timestamp, and other control is supported.
SetProperty	Sets a property value for a connection.
SetProperty<datatype>	Sets a property as a particular data type.
Update	Updates records in the data source. Key values and update values are provided. No keys are required for writeback operations.

LCFieldlist

The LCFieldlist class is the primary class for manipulating data through a connection. It binds a group of fields together with names and a default order.

When you generate a result set and pass in an empty fieldlist, the connector automatically populates the fieldlist. For each data element of the result set, the fieldlist receives the element's name and an LCField object of the corresponding data type. The result set can be controlled by manually building the fieldlist before generating the result set, or by using the FieldNames property of the connection (see FieldOut in the example on LCFieldList).

The Select and Call connection methods use an optional fieldlist of keys or parameters to restrict the result set. You construct this fieldlist manually and pass it to the Select method. You construct a key or parameter list by appending or inserting names and datatypes to the list. These methods create fields in the fieldlist and return the fields for further manipulation. The fields are then given values and, using field flags, can be given conditions such as greater-than and not-equal (see keyList in the example on LCFieldList).

The following tables list the LCFieldlist properties and methods. See the LC LSX documentation for syntax and details.

LCFieldlist Properties	*Description*
FieldCount	Long. Read-only. Number of fields in a fieldlist.
Fields	Variant. Read-only. Array of LCField objects.
Names	Variant. Array of strings.
RecordCount	Long. Read-only. Number of records per field.
Sequence	Long. Read-only. A fieldlist sequence number.

LCFieldlist Methods	*Description*
Append	Appends a field to an existing fieldlist.
Copy	Creates a duplicate copy of an LCFieldlist and all its data.
CopyField	Copies an existing field within a fieldlist at a specified position.
CopyRef	Creates a new FieldList object as a partial copy of another fieldlist.
GetField	Gets a field reference from a fieldlist result set and places its value into a variable.
GetName	Gets a copy of the name of a field within a fieldlist.
IncludeField	Includes an existing field in a fieldlist
Insert	Inserts a new field into an existing fieldlist at a specified index position.
List	Iterates through fields in a fieldlist, optionally returning information.
Lookup	Locates a field in a fieldlist based on the field name.
Map	Remaps fields in a fieldlist.
MapName	Maps fields with different names.
Merge	Merges two fieldlists, creating a new mapping fieldlist.
MergeVirtual	Merges two fieldlists, creating a new mapping and virtual fieldlist.
Remove	Removes an existing field from a fieldlist.
Replace	Replaces a field within a fieldlist.
SetName	Changes the name of a field within a fieldlist.

LCField

LCField is a storage class that contains one or more data values. The data type of a field applies to all the values and can be one of the four advanced datatypes.

The following tables list the LCField properties and methods. See the LC LSX documentation for syntax and details.

LCField Properties	*Description*
Count	Long. One or greater. A counter for the fieldlist record count.
Datatype	Long. One of the Lotus Connector data types.
Flags	Long. Zero or more field flags.
IsNull	Boolean. True or False.
Text	String. Array of string elements.
Value	Array of LotusScript data types. The values in the array depend on the data type of the field.

LCField Methods	*Description*
ClearVirtualCode	Clears the specified virtual code for the field.
Compare	Compares data values between two fields and returns the relationship.
Convert	Converts data values to the data type of the destination field.
Copy	Creates a copy of a field.
GetCurrency	Retrieves a currency data type from a value in a field.
GetDatetime	Retrieves a datetime data type from a value in a field.
GetFieldList	Retrieves a fieldlist from a field.
GetFloat	Retrieves a LotusScript double data type from a value in a field.
GetFormatNumber	Retrieves the format of a number type field.
GetFormatStream	Retrieves the format of a stream type field.
GetInt	Retrieves an integer value from a value in a field.
GetNumeric	Retrieves a numeric value from a value in a field.
GetStream	Retrieves a stream value from a value in a field.
LookupVirtualCode	Check if a specific virtual code was set for a field.
SetCurrency	Assigns the currency value to the specified index of the field.
SetDatetime	Assigns the datetime value to the specified index of the field.
SetFieldList	Assigns the fieldlist value to the specified index of the field.
SetFloat	Assigns a value to a field of type float.

continued

LCField Methods	*Description*
SetFormatDatetime	Assigns the current format setting for a datetime field.
SetFormatNumber	Assigns the current format setting for a number field.
SetFormatStream	Assigns the current format setting for a stream field.
SetInt	Assigns a value to a field of type long.
SetNumeric	Assigns a value to a field of type LCNumeric.
SetStream	Assigns a value to a field of type LCStream (text or binary).
SetVirtualCode	Adds a virtual code to the list of virtual codes for a field.

LCStream

LCStream is a general purpose text and binary data type. A stream has a format that details the character set of the text and any special attributes of the binary data. Use it to store and manipulate text or binary data in the supported character sets.

The following tables list the LCStream properties. See the LC LSX documentation for syntax and details.

LCStream Properties	*Description*
Flags	Long. The flags for the stream.
Format	Long. The stream data format.
Length	Long. The length of the stream data.
MaxLength	Long. Read-only. The maximum valid data length for the stream. Any value is valid, with a value of zero indicating no maximum length.
RangeCount	Long. Read-only. Range of elements in a stream. Valid only for List<Type> formats.
Text	String. Text representation of the stream.
Value	Variant. An array. If the stream is a string, it is a one element array. If the stream is a number or datetime, it is an array of numbers. It can also be an array of strings for text lists.
ValueCount	Long. Read-only. Number of elements in a stream. Valid only for List <Type> formats.

LCStream Methods	*Description*
Append	Appends one stream to another to yield a third LCStream object containing the data from both.
Clear	Clears a stream value and properties.
Compare	Compares two LCStream objects.
Convert	Converts a stream to a particular stream format.
Copy	Copies one LCStream to another.
DatetimeListGetRange	Gets a range of values in a DatetimeList LCStream object.
DatetimeListGetValue	Retrieves a datetime value from a specified place in a DatetimeList LCStream object.
DatetimeListInsertRange	Inserts a datetime range into a DatetimeList LCStream object.
DatetimeListInsertValue	Inserts a value into a DatetimeList LCStream object.
DatetimeListRemoveRange	Removes a range from a DatetimeList LCStream object.
DatetimeListRemoveValue	Removes a value from a DatetimeList LCStream object.
Extract	Creates a new LCStream object with part of the data from an existing LCStream object.
Merge	Combines one stream into another, creating a new stream.
NumberListGetRange	Selects a particular range from a number list LCStream object.
NumberlistGetValue	Retrieves a specified value from a number list LCStream object.
NumberListInsertRange	Inserts a number range into a number list LCStream object.
NumberListInsertValue	Inserts a value into a number list LCStream object.
NumberListRemoveRange	Removes a range of numbers from a number list LCStream object.
NumberListRemoveValue	Removes a value from a number list LCStream object.
ResetFormat	Resets the format of an LCStream object, without affecting the data or other properties.
SetFormat	Sets the format for an LCStream object.
TextListFetch	Fetches a text list from a text list stream object and assigns it to another stream object.
TextListInsert	Inserts text or a text list into a text list LCStream object.
TextListRemove	Removes a string from a text list LCStream object.
Trim	Trims trailing spaces from a text stream.

LCNumeric

The LCNumeric class is a container for high precision numbers.

The following tables list the LCNumeric methods and properties. See the LC LSX documentation for syntax and details.

LCNumeric Properties	*Description*
Precision	Long. Read-only. Precision and scale are set when the LCNumeric object is constructed.
Scale	Long. Read-only. Precision and scale are set when the LCNumeric object is constructed.
Text	String representation.
Value	Double. Value conversion between Lotus connectors and LotusScript double.

LCNumeric Methods	*Description*
Add	Adds two numeric values, producing the sum.
Compare	Compares two numeric values, returning a value indicating the relationship between them.
Copy	Makes a copy of a numeric value.
Subtract	Subtracts one numeric value from another, producing the result.

LCCurrency

The LCCurrency class is a fixed point decimal data type with 4 decimal places and 19 digits of precision. (This data type is mathematically equivalent to the LotusScript data type and is provided to support connections with a dedicated currency.)

The following tables list the LCCurrency properties and methods. See the LC LSX documentation for syntax and details.

LCCurrency Properties	*Description*
Text	A string representation.
Value	The value of a LotusScript Currency data type.

LCCurrency Methods	*Description*
Add	Adds two currency values, producing the sum.
Compare	Compares two currency values and returns the relationship.
Copy	Makes a copy of an LCCurrency object.
Subtract	Subtracts an LCCurrency value from another, producing the result.

LCDatetime

The LCDatetime class is a date and time data type that is accurate to the hundredth of a second and that specifies time zones and daylight savings time.

The following tables list the LCDatetime methods and properties. See the LC LSX documentation for syntax and details.

LCDatetime Properties	*Description*
Day	Long. Day of month. Valid values 1 to 31.
DST	Boolean. Indicates whether Daylight Savings Time is in effect.
Hour	Long. Hours. Valid values 0 to 23.
Hundredth	Long. Hundredth. Valid values 0 to 99.
Julian	Long. The Julian date.
Minute	Long. Minutes. Valid values 0 to 59.
Month	Long. Month. Valid values 1 to 12.
Second	Long. Seconds. Valid values 0 to 59.
Text	String. A string representation of the date.
Ticks	Long. Tick time representing the hundredths of seconds since midnight. Valid values 0 to 8640000.
Value	Variant. LotusScript Variant containing the datetime.
Weekday	Long. Read-only. Integer indicating the day of the week. Valid values 1 to 7. Sunday = 1.
Year	Long. Integer indicating the year. Valid values 1 to 32767.
Zone	Long. Integer indicating the time zone. Valid values -12 to 12.

LCDatetime Methods	*Description*
Adjust	Alters a datetime by a specified number of units.
Clear	Clears a datetime value.
Compare	Compares two datetime values, returning a value indicating the relationship between them.
Copy	Makes a copy of a LCDatetime object.
GetDiff	Returns the difference between two datetimes in the requested time units.
SetConstant	Produces a special constant Datetime commonly used for comparisons.
SetCurrent	Sets a Datetime value to the current system time.

Working with LC LSX

The typical use of connectors is to gather, create, and modify data in the enterprise system. For example, a Domino application has a number of data fields on a form. Once user input is complete, a button activates a script to update corresponding information in a an external data store. In our example the external data store is a DB2 table. The script is responsible for:

1. Accessing the active Domino form
2. Gathering the input data

   The input data is the key value for the record we want to work with in the enterprise system

1. Creating a connection to the enterprise system
2. Selecting the data based on the input values
3. Loading the data from the enterprise system
4. Transfering the results to the active Domino form

Here is a simple script to accomplish the task. The assumption is that the Domino form has a single text input field called Customer. The script uses the value of the Customer field to locate the corresponding customer information in DB2 and return an address and phone number for the customer, storing the return values in the form using fields called Address, City, State, and Phone.

Note No attempt is made to prescribe a code style. The practice of grouping object declarations together at the beginning of a script versus locating declarations close to the code is a preference and does not affect the execution. In this example, declarations and code are grouped to facilitate explaining the process.

The LC LSX is registered on the machine during installation of the Domino Server or Notes Client. The first step in writing the script is to load the LC LSX. The UseLSX statement in the Options event accomplishes this step:

```
Uselsx "*lsxlc"
```

The remainder of the script is in the Click event of the form's button. Errors are displayed to the user through a simple error handler at the bottom of the example. The LCSession class has a Status property that can be used to determine if the error handler was triggered by an LSX error or a LotusScript error. In all cases where the LSX reports an error, the LotusScript Error function returns error information. However, the LSX has additional error information not available through the LotusScript error statements. Creating and initializing the LCSession status provides this additional information. The creation of an LCSession object is only necessary if you require this additional error reporting.

```
Sub Click (Source as Button)
     On Error Goto Handler
     Dim session as New LCSession
     session.ClearStatus
```

The input values are in the current active document. This information is accessible through a NotesUIDocument object located through the CurrentDocument property of the NotesUIWorkspace class.

```
     Dim wksp As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     Set uidoc = wksp.CurrentDocument
```

The next step establishes a connection to the Lotus connector for DB2. After the connection is created, its properties are accessible and can be set. Common properties include Database and/or Server, UserID, and Password. Properties are not case sensitive. The following code connects to the DB2 system called EILAB as db2admin with the password "password".

```
    Dim src As New LCConnection ("db2")
    src.Database = "EILAB"    src.UserID = "db2admin"
src.Password = "password"    src.Connect
```

Four LCConnection methods perform query operations:

- **Catalog**: Returns metadata information about the enterprise system, for example, the tables of a DB2 database or the columns of a Sybase table. For a complete list of Catalog options, see the Catalog method in the documentation.
- **Execute**: Creates a result set using a connector-specific command statement to determine the contents of the result set. This method is helpful when the enterprise system's command structure is familiar and cross-connector portability is not an issue.
- **Select**: Creates a result set using a combination of key names, values, and condition flags to indicate the desired contents. This method works across connectors and does not require knowledge of the connector's command language.
- **Call**: Creates a result set using procedures and functions. Instead of keys, you specify parameters.

For the example, the DB2 table of interest is the "Customer" table, as specified by the Metadata property of LCConnection. The only record of interest is the customer named by the input value from the Domino form. You accomplish this selection by creating a key list. The default key flag LCFIELDF_KEY requests an exact match. If you want an inequality match such as greater-than or like, combine LCFIELDF_KEY and conditional key constants using OR operations. (In all cases, key fields must have the

LCFIELDF_KEY constant in addition to any optional conditional flag constants.)

```
    Dim keys As New LCFieldList
    Dim field As LCField
    src.Metadata = “Customer”    Set field = keys.Append
(“Name”, LCTYPE_TEXT)
    field.Flags = LCFIELDF_KEY
    field.Text = uidoc.FieldGetText (“Customer”)
```

The Select method creates a result set of all records from the enterprise system that match the keylist. If you substitute the LotusScript keyword Nothing for the key list, all records of the specified metadata are selected. This example just wants the customer record matching the input value from the Domino form. The key list is created to make this restriction.

The fieldlist receiving the result set is currently empty. The selection populates the fieldlist with the fields from the DB2 table. If you don’t need all the metadata fields, you can limit the result set to the fields of interest, either by creating the fieldlist prior to the selection or by setting the FieldNames property of the connection.

```
    src.FieldlNames = “Address, City, State, OfficePhone”
```

The selection returns one of three values: the number of records selected, zero (0) if no matching records are found, or LCCOUNT_UNKNOWN when records are found but the connection does not know the total. Since a return of zero is the only case where data is not found, it is the test case for error handling or branching.

```
    Dim fields As New LCFieldList
    If (src.Select (keys, 1, fields) = 0) Then End
```

The result set does not retrieve data. The Fetch LCConnection method reads the data from the enterprise system into the fieldlist. You access values from a fieldlist using the expanded class properties. For each field in a fieldlist, there is a property with the corresponding name. This property is an array of values using the closest available LotusScript data type to match the LC LSX data types.

```
If (src.Fetch (fields) > 0) Then
    Call uidoc.FieldSetText (“Address”, fields.Address(0))
    Call uidoc.FieldSetText (“City”, fields.City(0))
    Call uidoc.FieldSetText (“State”, fields.State(0))
    Call uidoc.FieldSetText (“Phone”, fields.OfficePhone(0))
End If
```

The data is retrieved from the enterprise system and placed in the Domino document. The final step is to refresh the Domino document to display the new data to the user. The end statement is required to prevent the application from going into the Error handler.

```
uidoc.Refresh
End ‘ Exit Sub will also work here
```

Error Handling

The first line of code in the following example indicates the start of the error handler. Testing for an LSX error first provides additional information in the case of an object creation error. Without the session object and subsequent test in the error handler, failure while creating a connection to DB2 generates the LotusScript message, `Error creating product object.` However, for the same error condition, the LSX reports `Error: Cannot load LSX library ‘db2.’`

```
Handler:
    If (Session.Status <> LCSUCCESS) Then
        Messagebox Session.GetStatusText, 0, _
        “The following Lotus Connector error has occurred”
Else
        Messagebox Error$, 0, _
        “The following LotusScript error has occurred”    End
If
End
End Sub
```

Example Using LCSession to Determine the Installed Connectors

The following example shows how to determine what Lotus Connectors are registered on the system. It will display a message box that will show the installed Connectors.

```
Uselsx “*LSXLC” ‘ This code goes under Options in the form

Sub Initialize  ‘ This sub is in the Initialize event of a
                ‘ button
    Dim session As New LCSession
    Dim conName As String
    Dim text_str As String
  ‘ list the connectors available
  ‘ the parameters for connector code, identity flags, and
  ‘ identity names are optional and omitted in this example
    Call session.ListConnector(LCLIST_FIRST, conName)
    text_str = conName
    While session.ListConnector(LCLIST_NEXT, conName)
    text_str = text_str + “, “ + conName
```

```
    Wend
    Msgbox "The usable Connectors are " & text_str

End Sub
```

Example Using SQL to Generate a Result Set

The next example illustrates how to use the execute method to run a statement against a data source. The statement must be provided in the data source's language. You can use SQL if the data source supports it. This example queries a DB2 database.

```
Uselsx "*LSXLC" ' This code goes under Options in the form

Sub Initialize  ' This sub is in the Initialize event of a
                ' button
On Error Goto Handler
Dim src As LCConnection
Dim FieldOut As New LCFieldList
Dim Session As New LCSession
Dim count As Long
Dim SQLString As String
Dim hr_id As String
Session.ClearStatus
Set src = New LCConnection ("db2") 'Create Connection object
src.Database = "EILAB"src.Metadata = "attendees"src.UserID =
"db2admin"src.Password = "password"src.Connect      'Connect to
db2 data source 'EILLAB'hr_id = Inputbox("Please enter HR
number ","HR name request")
SQLString = "Select Fname, Lname from attendees " & _
"where acakey = '"+ hr_id +"'"count =
src.Execute(SQLString,FieldOut) 'Execute the Query
count = src.Fetch(FieldOut) 'Fetch the unique row
If count = 1 Then 'Display the result
  Msgbox FieldOut.FName(0)+" "+FieldOut.LName(0)+" is a
Member"Else
  Msgbox Cstr(hr_id)+ " is NOT a Member"End If
Exit Sub

Handler:
  'Error Handler
If ( session.status <> LCSUCCESS) Then
    Messagebox session.GetstatusText, 0, _
    + "The following Lotus Connector error has occurred."Else
    Messagebox Error$, 0, _
    + "The following LotusScript error has occurred."End If
End Sub
```

Example Using the Select Method to Generate a Result Set

This example illustrates how to use the Select method to query a data source. You do not use SQL with this method. This example queries a DB2 database. This example produces the same output as the previous example.

```
Uselsx "*LSXLC" ' This code goes under Options in the form

Sub Initialize  ' This sub is in the Initialize event of a
                ' button
On Error Goto Handler
Dim src As LCConnection
Dim keyLst As New LCFieldList
Dim field As LCField
Dim FieldOut As New LCFieldList
Dim count As Long
Dim hr_id As String
Dim Session As New LCSession
session.ClearStatus
Set src = New LCConnection ("db2") 'Create Connection object
src.Database = "EILAB"src.Metadata = "attendees"src.UserID =
"db2admin"src.Password = "password"Src.Connect 'Connect to data
source 'EILAB'hr_id = Inputbox("Please enter HR number ","HR
name request")
'You specify which columns are needed
src.FieldNames = "FNAME, LNAME" _
'Specify the key to access the table
Set field = keyLst.Append("ACAKEY", LCTYPE_TEXT)
field.Flags = LCFIELDF_KEY
field.Value = hr_id 'The value you want to search on
'Execute the Query
count = src.Select(keyLst, 1, FieldOut)
'Fetch the unique row
count = src.Fetch(FieldOut)
'Display the result
If count = 1 Then
  Msgbox FieldOut.FName(0)+" "+FieldOut.LName(0)+" is a
Member"Else
  Msgbox Cstr(hr_id)+ " is not a Member"End If
Exit Sub

Handler:
  'Error Handler
If ( session.status <> LCSUCCESS) Then
  Messagebox session.GetstatusText, 0, _
  + "The following Lotus Connector error has occurred."Else
  Messagebox Error$, 0, _
  + "The following LotusScript error has occurred."End If
End Sub
```

Java Classes for Lotus Connectors

The Java classes for Lotus Connectors (LC Java classes) extend use of the Lotus Connectors to Java. The programming model is independent of individual connectors. This eliminates the need to learn a separate programmatic model to access each individual system, and still allows experienced users to access specific system features.

Through the LC Java classes, Java agents and applications can retrieve and act upon data from enterprise systems.

Software Requirements

Obtain the LC Java software and documentation from the Lotus Enterprise Integration Web site

```
http://www.lotus.com/enterpriseintegration
```

The LC Java software also comes bundled with LEI.

Make the following adjustments to the CLASSPATH and PATH environment variables:

- **CLASSPATH**: add the file lcjava.zip. This file is typically installed in your Domino or LEI Java or program directory.
- **PATH**: add the directory to which LC Java was installed. This is typically your Domino or LEI program directory.

For example:

```
set CLASSPATH=%CLASSPATH%;c:\notes\domino\java\lcjava.zip
set PATH=%PATH%;c:\notes
```

If you are developing Java agents, create a Domino environment variable named JavaUserClasses and add the file lcjava.zip. For example:

```
JavaUserClasses=c:\notes\domino\java\lcjava.zip
```

Put the following import statement in your program:

```
import lotus.lcjava.*;
```

LC Java Classes

The public LC Java classes consist of

- LCSession
- LCException
- LCConnection
- LCFieldlist
- LCField

and the advanced data types

- LCStream
- LCNumeric
- LCCurrency
- LCDatetime
- LCDatetimeParts.

LCSession

You must create an LCSession object before using any other LC Java facilities. The LCSession object contains global state information. To create the object, call the constructor with a value of 0 for the parameter. For example:

```
LCSession session = new LCSession(0);
```

The LCSession constructor and many other LC methods throw LCException, which extends java.lang.Exception. LCException contains the method getLCErrorCode to get an integer error code specific to LC. The LCSession method getStatusText returns the message associated with an error.

The LCSession method "free" disconnects all LC connections and frees all LC objects.

Here is the framework for typical LC Java code:

```
LCSession session = null;

public void getGoing()
{
try
    {
    session = LCSession(0);
    }
catch(LCException e)
    {
    int err = e.getLCErrorCode();
    String errmsg = session.getStatusText(err);
    System.out.println(errmsg);
    }
}

public void allDone()
{
    session.free();
}
```

The following table lists the LCSession methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCSession Method	*Description*
addProperty	Creates a new property and assigns a token.
clearStatus	Clears the current error code.
finalize	Calls free.
free	Disconnects all connections and frees all LC objects.
getProperty	Gets a property value as an LCField, LCCurrency, LCDateTime, Double, Integer, LCNumeric, or LCStream object.
getPropertyBoolean	Gets a property value as a boolean.
getPropertyJavaDate	Gets a property as a Date object.
getPropertyJavaStringBuffer	Gets a property as a StringBuffer object.
getStatusText	Gets the message associated with an error code.
listConnector	Lists the connectors available to the session.
listProperty	Gets property information as an LCStream or StringBuffer object.
log	Adds a log entry for an internal basic error or event.
logEX	Adds a log entry for an internal extended error or event.
logJavaString	Adds a log entry for an external basic error or event.
logJavaStringEx	Adds a log entry for an external extended error or event.
logStream	Adds a log entry for an external basic error or event.
logStreamEx	Adds a log entry for an external extended error or event.
lookupConnector	Gets the virtual code for a connector.
runActivity	Executes LC activities.
setProperty	Sets a property value from an LCField, LCCurrency, LCDateTime, Double, Integer, LCNumeric, or LCStream object.
setPropertyJavaBoolean	Sets a property value from a Boolean object or a boolean.
setPropertyJavaDate	Sets a property value from a Date object.
setPropertyJavaString	Sets a property value from a String object.

LCConnection

The LCConnection class represents an instance of a Lotus connector, providing access capabilities to the enterprise system. Multiple connections can be allocated to a single connector.

The LCConnection constructor identifies the enterprise system by name (first parameter) or session token (second parameter). The connection method establishes a connection. Before establishing the connection, you must set the properties that apply to the connection.

The following code shows the skeleton for establishing and deallocating a typical connection. This example specifies the enterprise system by name in the LCConnection constructor and sets the SERVER, DATABASE, METADATA, UID, and PWD properties with setPropertyJavaString before calling the connection method.

```
LCConnection connection = void;

public void doConnect(String server, String database, String
metadata, String uid, String pwd)
{
try
    {
    connection = new LCConnection(“db2”, 0);
    if(server.length() > 0)
        connection.setPropertyJavaString
        (LCTOKEN.SERVER, server););
    if(database.length() > 0)
       connection.setPropertyJavaString
        (LCTOKEN.DATABASE, database);
    if(metadata.length() > 0)
        connection.setPropertyJavaString
        (LCTOKEN.METADATA, metadata);
    if(uid.length() > 0)
        connection.setPropertyJavaString
        (LCTOKEN.UID, uid);
    if(pwd.length() > 0)
        connection.setPropertyJavaString
        (LCTOKEN.PWD, pwd);
    connection.connection();
    }
catch
    {
    int err = e.getLCErrorCode();
    String errmsg = session.getStatusText(err);
    System.out.println(errmsg);
    }
}

public void undoConnect()
{
    connect.disconnect();
}
```

The following table lists the LCConnection methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCConnection Method	*Description*
action	Performs one of the actions: reset connection, truncate records, commit changes, roll back changes.
catalog	Catalogs metadata information to a fieldlist.
connection	Establishes a connection to a connector.
create	Creates a metadata object.
disconnect	Disconnects a connection.
drop	Destroys a metadata object.
execute	Executes a native connector command and returns the result set to a fieldlist.
fetch	Gets records from the current result set.
free	Deallocates the connection.
getProperty	Gets a property value as an LCField, LCCurrency, LCDateTime, Double, Integer, LCNumeric, or LCStream object.
getPropertyBoolean	Gets a property value as a boolean.
getPropertyJavaDate	Gets a property as a Date object.
getPropertyJavaString Buffer	Gets a property as a StringBuffer object.
insert	Inserts new records into the connection.
listProperty	Gets property information as an LCStream or StringBuffer object.
log	Adds a log entry for an internal basic error or event.
logEx	Adds a log entry for an internal extended error or event.
logJavaString	Adds a log entry for an external basic error or event.
logJavaStringEx	Adds a log entry for an external extended error or event.
logStream	Adds a log entry for an external basic error or event.
logStreamEx	Adds a log entry for an external extended error or event.
remove	Removes the most recently fetched writeback record, or removes all records specified by a key.
select	Selects the current metadata and returns the result set to a fieldlist.
setProperty	Sets a property value from an LCField, LCCurrency, LCDateTime, Double, Integer, LCNumeric, or LCStream object.
setPropertyJavaBoolean	Sets a property value from a Boolean object or a boolean.
setPropertyJavaDate	Sets a property value from a Date object.
setPropertyJavaString	Sets a property value from a String object.
update	Updates the most recently fetched writeback record, or updates all records specified by a key.

LCFieldlist

The LCFieldlist class is the primary class for manipulating data through a connection. It binds a group of fields together with names and an implied order.

When you generate a result set with an empty fieldlist, the connector automatically populates the fieldlist. For each data element of the result set, the fieldlist receives the element's name and an LCField object of the corresponding data type. You can control the result set by manually building the fieldlist first, or by specifying the FIELD_LIST connector property.

The LCFieldlist constructor allocates a fieldlist for a specified number of records.

The following table lists the LCFieldlist methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCFieldlist Method	*Description*
append	Appends a field to the end of a fieldlist.
copy	Copies a fieldlist to a new fieldlist.
copyRef	Copies a fieldlist's metadata to a new fieldlist so that the new fieldlist references the same data as the original fieldlist.
free	Frees a fieldlist, including all data and metadata.
getCount	Returns the number of fields in a fieldlist.
getField	Gets a field from a fieldlist.
getName	Gets the name of a field in a fieldlist.
getNameLength	Returns the total length of all field names in a fieldlist.
getRecordCount	Returns the number of records a fieldlist was allocated with.
getSequence	Returns the sequence number of a fieldlist.
insert	Inserts a field in a fieldlist.
list	Iterates through the fields in a fieldlist and optionally returns information.
listSetup	Prepares for iteration through a fieldlist.
lookup	Locates a field from a fieldlist based on field names.
merge	Merges two fieldlists, creating a new mapping fieldlist.
mergeVirtual	Merges two fieldlists, creating new mapping and virtual fieldlists.
remove	Removes a field from a fieldlist.
replace	Replaces a field in a fieldlist.
setName	Assigns a new name to a field in a fieldlist.

LCField

LCField is a storage class that contains one or more data values. The data type of a field applies to all the value and can be LCStream, LCNumeric, LCCurrency, LCDatetime, long, or double.

The LCField constructor allocates a new field for a specified data type, and a specified number of values for append, insert, and replace operations.

The following table lists the LCField methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCField Method	*Description*
clearVirtualCode	Clears the virtual code for a field.
compare	Compares data values.
convert	Converts data values to the types from a specified field.
copy	Copies a field to a new field.
free	Frees an LCField object, including associated data.
fromJavaInt	Assigns a field value from an int or Integer.
fromJavaString	Assigns a field value from a String object.
getBuffer	Retrieves the data and null indicator pointers for a field.
getCurrency	Retrieves a field value as an LCCurrency object.
getDatetime	Retrieves a field value as an LCDatetime object.
getFlags	Returns the flags for a field.
getFloat	Retrieves a field value as a double object.
getFormatDatetime	Retrieves the format for an LCDatetime field.
getFormatNumber	Retrieves the format for an LCNumber field.
getFormatStream	Retrieves the format for an LCStream field.
getInt	Retrieves a field value as an Integer object.
getJavaStringBuffer	Retrieves a field value as a String object.
getNumeric	Retrieves a field value as an LCNumeric object.
getStream	Retrieves a field value as an LCStream object.
getType	Returns the data type of a field.
getTypeSize	Returns the size of the data value for a field.
getValueCount	Retrieves the number of values for a field.
isNull	Indicates whether the data value for a field is null.
setCurrency	Sets a field value from an LCCurrency object.
setDatetime	Sets a field value from an LCDatetime object.
setFlags	Sets the flags for a field.

continued

LCField Method	*Description*
setFloat	Sets a field value from a double.
setFormatDatetime	Sets the format for an LCDatetime field.
setFormatNumber	Sets the format for an LCNumber field.
setFormatStream	Sets the format for an LCStream field.
setInt	Sets a field value from an int.
setJavaString	Sets a field value from a String object.
setNull	Sets a field value to null.
setNumeric	Sets a field value from an LCNumeric object.
setStream	Sets a field value from an LCStream object.
setVirtualCode	Sets the virtual code for a field.
toJavaInt	Returns a field value as an int.
toJavaString	Returns a field value as a String object.

LCStream

LCStream is a general purpose text and binary data type. A stream has a format that details the character set of the text or any special attributes of the binary data.

Constructors exist to create LCStream objects from double, LCCurrency, LCDatetime, LCNumeric, String, and no values.

The following table lists the LCStream methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCStream Method	*Description*
compare	Compares two LCStream objects.
copy	Copies a stream.
datetimeListGetCount	Gets the number of values and ranges in a datetime list.
datetimeListGetFirst	Gets the first datetime in a datetime list.
datetimeListGetLength	Returns the total length of a datetime list.
datetimeListGetRange	Gets a datetime range.
datetimeListGetValue	Gets a datetime value.
extract	Gets a stream from the contents of another stream.
free	Used by LCActivity for auto-freeing.
fromCurrency	Puts an LCCurrency value into a stream.
fromDatetime	Puts an LCDatetime value into a stream.
fromFloat	Puts a double value into a stream.

continued

LCStream Method	*Description*
fromInt	Puts an int value into a stream.
fromJavaBoolean	Puts a Boolean or boolean value into a stream.
fromJavaDate	Puts a Date value into a stream.
fromJavaDouble	Puts a Double or double value into a stream.
fromJavaInt	Puts an Integer or int value into a stream.
fromJavaString	Puts a String value into a stream.
fromNumeric	Puts an LCNumeric value into a stream.
getFlags	Returns a stream's flags.
getLength	Returns the length of the stream's data.
merge	Creates a new stream from two existing streams.
numberListGetCount	Gets the number of values and ranges in a number list.
numberListGetFirst	Gets the first number in a number list.
numberListGetLength	Returns the total length of a number list.
numberListGetRange	Gets a number range.
numberListGetValue	Gets a number value.
setFlags	Sets a stream's flags.
textListGetCount	Gets the number of entres in a text list.
textListGetLength	Returns the length of a text list.
toJavaBoolean	Returns a boolean representation of a stream.
toJavaDate	Returns a Date representation of a stream.
toJavaDouble	Returns a double representation of a stream.
toJavaInt	Returns an int representation of a stream.
toJavaString	Returns a String equivalent to a stream.
trim	Trims trailing spaces from a stream.

LCNumeric

The LCNumeric class is a container for very high precision numbers. It contains a precision, scale, and variable number of digits.

Constructors exist to create LCNumeric objects from double, LCStream, String, and no values.

The following table lists the LCNumeric methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCNumeric Method	*Description*
add	Adds two numeric values.
compare	Compares two numeric values.
copy	Copies a numeric.
fromFloat	Puts a double value into a numeric.
fromJavaDouble	Puts a Double or double value into a numeric.
fromJavaInt	Puts an Integer or int value into a numeric.
fromJavaString	Puts a String value into a numeric.
fromStream	Puts an LCStream value into a numeric.
getPrecision	Returns the precision of a numeric value.
getScale	Returns the scale of a numeric value.
subtract	Subtracts two numeric values.
toJavaDouble	Returns a double representation of a numeric.
toJavaInt	Returns an int representation of a numeric.
toJavaString	Returns a String equivalent to a numeric.

LCCurrency

The LCCurrency class is a fixed point decimal data type with 4 decimal places and 19 digits of precision. (This data type is mathematically equivalent to the LotusScript data type and is provided to support connections with a dedicated currency.)

Constructors exist to create LCCurrency objects from double, LCStream, String, and no values.

The following table lists the LCCurrency methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCCurrency Method	*Description*
add	Adds two currency values.
compare	Compares two currency values.
copy	Copies a currency.
free	Used by LCActivity for auto-freeing.
fromFloat	Puts a double value into a currency.
fromJavaDouble	Puts a Double or double value into a currency.
fromJavaInt	Puts an Integer or int value into a currency.
fromJavaString	Puts a String value into a currency.
fromStream	Puts an LCStream value into a currency.

continued

LCCurrency Method	*Description*
getPrecision	Returns the precision of a numeric value.
getScale	Returns the scale of a numeric value.
subtract	Subtracts two currency values.
toJavaDouble	Returns a double representation of a currency.
toJavaInt	Returns an int representation of a currency.
toJavaString	Returns a String equivalent to a currency.

LCDatetime

The LCDatetime class is a date and time data type that is accurate to the hundredth of a second and that specifies time zones and daylight savings time.

Constructors exist to create LCDatetime objects from Date, LCStream, String, and no values.

The following table lists the LCDatetime methods. Some of these methods have several signatures. See the documentation for syntax and details.

LCDatetimeMethod	*Description*
adjust	Alters a datetime.
compare	Compares two datetime values.
copy	Copies a datetime.
free	Used by LCActivity for auto-freeing.
fromJavaDate	Puts a Date value into a datetime.
fromJavaString	Puts a String value into a datetime.
fromStream	Puts an LCStream value into a datetime.
getDiff	Returns the difference between two datetime values.
getJulian	Returns the Julian date of a datetime value.
getParts	Gets the LCDatetimeParts values of a datetime.
getTicks	Returns the hundreths of seconds since midnight GMT.
setConstant	Puts a special constant value into a datetime.
setCurrent	Puts the current date and time into a datetime.
setJulian	Puts a Julian date into a datetime.
setParts	Puts LCDatetimeParts values into a datetime.
setTicks	Sets the time to the hundredths of seconds since midnight GMT.
toJavaDate	Returns a Date representation of a datetime.
toJavaString	Returns a String equivalent to a datetime.

The LCDatetimeParts class consists of the following public variables: lcDay (int), lcDSTInEffect (boolean), lcHour (int), lcHundreth (int), lcMinute (int), lcMonth (int), lcSecond (int), lcWeekday (int), lcYear (int), and lcZone (int).

Examples for LC Java Classes

Presented below are two simple examples.

List Connectors

This example lists the available connectors to enterprise systems.

Create an LCSession object. The LCSession.listConnector method returns information on the first or next connection depending on the value of the first parameter. The example uses a loop to return information on all the connectors.

```
import lotus.domino.*;
import lotus.lcjava.*;

public class jctest extends AgentBase
{
    public void NotesMain()
    {
    try
        {
            // Create LC session
            LCSession session = null;
            try {
                session = new LCSession(0);
                System.out.println("Session ready");
            }
            catch (LCException e) {
                int err = e.getLCErrorCode();
                String errmsg = session.GetStatusText(err);
                System.out.println(errmsg);
            }

            // List connectors
            int status = 0;
            int lcList = LCLIST.FIRST;
            LCStream lcConnectorName = new LCStream();
            Integer lcConnectorCode = new Integer(0);
            LCStream lcIdentifyFlagList = new LCStream();
            LCStream lcIdentifyNameList = new LCStream();
            boolean done = false;
            System.out.println("List connectors");
            do
            {
                try {
```

```
                    status = session.listConnector(lcList,
                    lcConnectorName, lcConnectorCode,
                    lcIdentifyFlagList, lcIdentifyNameList);
                }
                catch (LCException e) {
                    int err = e.getLCErrorCode();
                    String errmsg = session.GetStatusText(err);
                    System.out.println(errmsg);
                    done = true;
                }
                String name = lcConnectorName.toJavaString();
                if (name != null)
                    System.out.println("\t" + name);
                lcList = LCLIST.NEXT;
            } while
              (status == LCFAIL.NO_ERROR && done == false);

        }
    catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Query Data Source

This example queries a db2 database. The name for a specified customer number is retrieved.

```
import lotus.notes.*;
import lotus.lcjava.*;

public class jctest2 extends AgentBase
{
    public void NotesMain()
    {
    try
        {
            LCSession session = null;
            LCConnection connection = null;

            try {
```

Create LCSession and LCConnection objects.

```
                // Create LC session
                session = new LCSession(0);
                System.out.println("Session ready");
```

```
// Create connection to db2
connection = new LCConnection("db2", 0);
System.out.println("Connection created");
```

Specify the properties for accessing the data source and establish the connection.

```
// Connect to the data source
connection.setPropertyJavaString(
LCTOKEN.DATABASE,
"EILAB");
connection.setPropertyJavaString(
LCTOKEN.METADATA,
"attendees");
connection.setPropertyJavaString(
LCTOKEN.USERID,
"db2admin");
connection.setPropertyJavaString(
LCTOKEN.PASSWORD,
"password");
connection.connection();
System.out.println
("Connection made to \"attendees\" in EILAB");
```

The following code sets up fieldlists for the query key (the customer number), the result set returned by the query, and the data fetched from the result set.

```
// Get the data
LCFieldlist keyList = new LCFieldlist(1, 0);
LCFieldlist resultList = new LCFieldlist(1, 0);
LCFieldlist fetchList = new LCFieldlist(1, 0);
```

The fieldlist for the key is one field of type LCStream whose value is the customer number. For purposes of the example, the key value is hard coded.

```
// Set up the key
LCField keyField = new LCField();
keyList.append(
"ACAKEY", LCTYPE.TEXT, keyField);
keyField.setFlags(LCFIELDF.KEY);
Integer recCount = new Integer(0);
LCStream hr_id = new LCStream("138703");
keyField.setStream(1, hr_id);
```

The fieldlist for the result set consists of two text (LCStream) fields.

```
// Set up the result set
LCField fnameResult = new LCField();
LCField lnameResult = new LCField();
resultList.append(
"FNAME", LCTYPE.TEXT, fnameResult);
```

```
                resultList.append(
                "LNAME", LCTYPE.TEXT, lnameResult);
```

The FIELD_LIST property specifies the fields to be retrieved from the data source. With no FIELD_LIST specification, all fields are retrieved. The LCConnection.select performs the query and populates the result set. You are expecting exactly one record.

```
                // Get the result set (FNAME and LNAME fields)
                connection.setPropertyJavaString(
                LCTOKEN.FIELD_LIST,
                "FNAME, LNAME");
                connection.select(
                keyList, 1, resultList, recCount);
```

The fieldlist for the fetched data is equivalent to the result set. The two calls to the LCConnection.fetch method retrieves the two fields in the result set.

```
                // Set up and fetch the result
                LCField fnameFetch = new LCField();
                LCField lnameFetch = new LCField();
                fetchList.append(
                "FNAME", LCTYPE.TEXT, fnameFetch);
                fetchList.append(
                "LNAME", LCTYPE.TEXT, lnameFetch);
                connection.fetch(fetchList, 1, 1, recCount);

                // Print results
                System.out.println("Name: " +
                fnameFetch.toJavaString() +
                " " + lnameFetch.toJavaString());
            }
```

The handler for the code involving LC calls catches LCException and gets the LC error code.

```
            catch (LCException e) {
                int err = e.getLCErrorCode();
                String errmsg = session.GetStatusText(err);
                System.out.println(errmsg);
            }

        }
    catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Lotus Connector Toolkit

You can use the Lotus Connector Toolkit API to access connectors and to develop your own connectors. The Toolkit is a C API.

Connectors are of two types:

- Standard connectors are used to access and manipulate data in data sources such as DB2, ODBC, and SAP R/3.
- MetaConnectors transform data between connectors and activities, sometimes altering the behavior of a connector. For example, order MetaConnectors sort the result set of standard connectors and meter MetaConnectors measure the flow of data through a standard connector for billing purposes.

Design Overview

The Lotus Connector Toolkit is designed with the following primary considerations:

- Support for high speed data movement.
- Ability to use product specific capabilities.
- Platform and location independence.
- Access to different types of data stores (for example, relational databases, Lotus Domino, Enterprise Resource Planning systems, Transaction Processing systems, and others).
- Independence from programming language. You can use connectors with languages other than C, like Java and LotusScript.

Installation

You can download the single zipped file for the installation of the LC API from the Lotus Enterprise Integration Web site at

`http://www.lotus.com/enterpriseintegration`

After you download the file, you can create a directory and unzip it into that directory. This will complete the installation. The installation process will create directories and copy files into the directories.

Support Files and Templates

The LC Toolkit documentation is installed as lc30api.nsf in the Domino or Notes doc directory. It provides the information you need to build the connectors.

The installation provides the source code for two connectors that can be used as templates. One template is a connector and the other a MetaConnector.

These templates can be used as a reference, or modified to build your own connectors. The connector template is a simplified ODBC connector. The MetaConnector template examines data that is fetched and inserted for a particular string and counts the occurrences. Also in the template directory are two files with information about the templates: readme.con for the connector template and readme.met for the meta connector.

Summary

In this chapter we have explained how the Lotus Connector programming interface gives you a uniform way to access other datasources from Domino. We have discussed how to access Lotus connectors from LotusScript and Java. Finally, we have explained how you can create your own Connector within the Lotus Enterprise Integration architecture by using the Lotus Connector Toolkit.

Chapter 5
Lotus EI Product Updates

Lotus provides specific integration solutions for ODBC, DB2, MQSeries, CICS, IMS, and SAP R/3. This chapter outlines the associated products with emphasis on recent updates. For detailed information, visit the Lotus Enterprise Integration Web site at

`http://www.lotus.com/enterpriseintegration`

LotusScript Data Object and ODBC

The following tools enable Notes applications to connect to data resources through the Open Database Connectivity (ODBC) technology:

- **LotusScript:Data Object (LS:DO)**: a LotusScript Extension (LSX) that provides additional LotusScript classes for accessing other data resources via ODBC.
- **@DBCommand, @DBLookup, @DBColumn using ODBC**: @functions for ODBC data access. The functions @DBLookup and @DBColumn are used to access Notes databases as well as ODBC-compliant databases.

What is ODBC?

The Open Database Connectivity (ODBC) standard is a set of functions developed by Microsoft to access Relational Database Management Systems (RDBMS) like Oracle, DB2, and others. There are two software components required to use ODBC:

- **ODBC Driver Manager**: a set of APIs in the ODBC dynamic link library. These APIs are called by client programs like LS:DO, NotesSQL, and so on, in order to access an RDBMS via ODBC. You need to install the appropriate ODBC driver manager as required by your operating system and by the applications that use the ODBC features.
- **RDBMS ODBC Driver**: the driver for specific RDBMSs like NotesSQL, DB2, and Oracle. The ODBC driver allows you to issue any SQL statements in Data Definition Language (DDL), Data Control Language (DCL), and Data Manipulation Language (DML) using SQLExecute or SQLExecDirect with the ODBC API. In addition, other ODBC Drivers enable you to get information about column attributes, index, privileges

of column, drivers, foreign keys of tables, and other RDBMS entities. You must install the driver for the system you are accessing.

Before using ODBC, you must register your data sources. In Windows, you use the ODBC Data Source Administrator to register data sources.

LotusScript:DataObject (LS:DO)

LotusScript:Data Object (LS:DO) provides full read and write access to external ODBC data sources using the complete control and flexibility of the LotusScript structured programming language.

LS:DO consists of a set of three classes:

- ODBCConnection
- ODBCQuery
- ODBCResultSet

These classes come complete with a powerful set of properties and methods and full SQL capabilities. At the same time, LS:DO is easy to learn and use because its design is consistent with LotusScript syntax and the LotusScript Notes classes.

Uselsx Statement

Place the following statement in the Options event of your LotusScript program to make the LS:DO classes available:

```
Uselsx "*LSXODBC"
```

Event Handling

If needed, you can create event-handling subroutines for some ODBC methods. An event-handling subroutine you create is called according to the behavior of an appropriate ODBC method, after the On Event statement is issued.

In the following example, an event handler named presub1 is called before the ListDataSources method is called.

1. On Event statement

```
Dim connection As New ODBCConnection
On Event BeforeListDataSources From connection Call presub1
```

2. Event handler

```
Sub presub1(Source As ODBCConnection)
     '** Write your event handling script here
End Sub
```

Your event handler must be in the scope where the event occurs.

ODBCConnection Class

The ODBCConnection class allows you to establish a connection. It also allows you to access some database catalog information, such as data source lists, table lists, procedures lists, and so on.

The following table shows the properties of the ODBCConnection Class:

Property	*Data Type*	*Read/ Write*	*Argument*
AutoCommit	Boolean	R/W	
CommitOnDisconnect	Boolean	R/W	
DataSourceName	String	R	
IsConnected	Boolean	R	
IsSupported(option)	Boolean	R	option: DB_SUPP_ASYNCHRONOUS DB_SUPP_CURSORS DB_SUPP_PROCEDURES DB_SUPP_READONLY DB_SUPP_SILENTMODE DB_SUPP_TRANSACTIONS
SilentMode	Boolean	R/W	

Note Boolean is not a predefined data type in LotusScript. However, you can use a constant value (TRUE and FALSE) as a Boolean data type.

The following table shows the ODBCConnection methods with the corresponding arguments and events.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
CommitTransaction		Boolean	DBstsNCON	BeforeCommit Transaction AfterCommit Transaction
ConnectTo(source$ [, userID$, password$])		Boolean	DBstsCANF DBstsSVRQ DBstsCCON DBstsACCS	BeforeConnect AfterConnect BeforeConnectTo AfterConnectTo
Disconnect		Boolean	DBstsNCON	BeforeDisconnect AfterDisconnect Transaction Update
GetError		Constant *1		

continued

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
GetErrorMessage([error%])	error%: DB_LAST ERROR or Constants *1	String		
GetExtendedError Message([error%])	error%: DB_LAST ERROR or Constants *1	String		
ListDataSources		Array of String		BeforeListData Sources AfterListData Sources
ListFields([tableName$])		Array of String	DBstsNCON DBstsNCOL	BeforeListFields AfterListFields
ListProcedures([source$ [, userID$, password$])		Array of String	DBstsNCON DBstsACCS	BeforeList Procedures AfterList Procedures
ListTables([source$ [, userID$, password$]])		Array of String	DBstsNCON DBstsACCS	BeforeListTables AfterListTables
RollbackTransaction		Boolean	DBstsNCON	BeforeRollback Transaction AfterRollback Transaction

*1 Error number list is shown in the *ODBCResultSet* Class section, later in this chapter.

Following are some sample uses of the ODBCConnection class:

1. To get a data source list registered by the ODBC administrator, use the ListDataSources method of the ODBCConnection class.

```
Dim con As New ODBCConnection
Dim dl As Variant
dl = con.ListDataSources
'** Keywords is a field name in which a data source list is
saved
keyDoc.Keywords = dl
```

2. To get a table list owned by a database, use the ListTables method of the ODBCConnection class.

```
Dim con As New ODBCConnection
Dim tl As Variant
'** sampleDB1 is a database name registered in this example
tl = con.ListTables("sampleDB1")
'** Keywords is a field name in which a table list is saved
keyDoc.Keywords = tl
```

Note When the ListTables method is issued, the SQLConnect ODBC API is called in LS:DO before getting a table list, so you don't need to execute the ConnectTo method.

3. To get a column name list owned by a table, use the ListFields method of the ODBCConnection class.

```
Dim con As New ODBCConnection
Dim Clist As Variant
Dim status As Variant
'** sampleDB1 is a database name registered in this example
status = con.ConnectTo("sampleDB1")
'** courses is a table name in the sampleDB1 database
CList = con.ListFields("courses")
'** Keywords is a field name in which a column list is saved
keyDoc.Keywords = Clist
```

ODBCQuery Class

The ODBCQuery class is used to hold the ODBCConnection object in which a connection is established, and to hold a SQL statement you want to use to perform the inquiry. The SQL statement is parsed through the ODBC driver that your application uses.

The properties of this class are shown below:

Property	*Data Type*	*Read/Write*
Connection	ODBCConnection Object	W
QueryExecuteTimeOut	Integer	R/W
SQL	String	R/W
UseRowID	Boolean	R/W

The ODBCQuery class provides the following methods:

Method	*Argument*	*Return Value*
GetError		Constant
GetErrorMessage([error%])	error%: DB_LASTERROR or Constants *1	String
GetExtendedErrorMessage ([error%])	error%: DB_LASTERROR or Constants *1	String

*1 Error number list is shown in the *ODBCResultSet* Class section, later in this chapter.

This sample shows the execution of an SQL statement using the Execute method in ODBCResultSet. The following steps are performed before carrying out the Execute method. The Connection method and the SQL property in ODBCResultSet class are also used in this example.

```
Dim con As New ODBCConnection
Dim qry As New ODBCQuery
Dim res As New ODBCResultSet
Dim status As Variant
'** sampleDB1 is a database name registered in this example
status = con.ConnectTo("sampleDB1")
Set qry.Connection = con
'** courses is a table name in the sampleDB1 database
qry.SQL = "select * from courses"Set res.Query = qry
```

ODBCResultSet Class

The ODBCResultSet class has many functions that are used to handle the records termed result sets. A result set holds the retrieved records of an SQL query which is specified with the ODBCQuery object.

The following table shows the properties available with the ODBCResultSet Class:

Property	*Data Type*	*Read/Write*
CacheLimit	Integer	R/W
CurrentRow	Integer	R/W
FetchBatchSize	Integer	R/W
HasRowChanged	Boolean	R
IsBeginOfData	Boolean	R
IsEndOfData	Boolean	R
IsResultSetAvailable	Boolean	R
MaxRows	Integer	R/W
NumColumns	Integer	R
NumRows	Integer	R
Query	ODBCQuery Object	W
ReadOnly	Boolean	R/W

The methods of the ODBCResultSet class can be categorized into the following areas:

- **SQL execution and transaction**: These methods are used to issue an SQL statement and to commit or roll back a transaction.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
Close(option)	Option: DB_CLOSE DB_COMMIT DB_ROLLBACK	Boolean		BeforeClose AfterClose
ExecProcedure (name$, arg$)		Boolean	DBstsNCON DBstsODBC	BeforeExecProcedure AfterExecProcedure
Execute ([option])	Option: DB_CANCEL	Boolean	DBstsODBC	BeforeExecute AfterExecute AsynchOperation Complete
Transactions (option)	Option: DB_COMMIT DB_ROLLBACK	Boolean		BeforeTransactions AfterTransactions

- **Result set row navigation and location**: These methods are used to locate a cursor on a result set which is produced by the Execute method.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
FirstRow		Boolean	DBstsINVR	BeforeFirstRow AfterFirstRow BeforeRowPositionChange AfterRowPositionChange
LastRow		Boolean		BeforeLastRow AfterLastRow BeforeRowPositionChange AfterRowPositionChange
LocateRow (column, value$ [, column, value$, ...])	column is Integer or String	Boolean	DBstsCARR DBstsEOFD DBstsNODA	BeforeLocateRow AfterLocateRow BeforeRowPositionChange AfterRowPositionChange
NextRow		Boolean	DBstsINVR DBstsEOFD	BeforeLocateRow AfterLocateRow BeforeRowPositionChange AfterRowPositionChange
PrevRow		Boolean	DBstsINVR DBstsCARR	BeforePrevRow AfterPrevRow BeforeRowPositionChange AfterRowPositionChange

- **Accessing column values**: These methods are used to access specific column values and to check column properties.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
GetValue(column [, variable])	column is Integer or String	Variant	DBstsINVC DBstsNODA DBstsCNVR	BeforeGetValue AfterGetValue
IsValueAltered (column)	column is Integer or String	Boolean	DBstsINVC	
IsValueNull(column)	column is Integer or String	Boolean	DBstsINVC	
SetValue(column, value)	column is Integer or String	Boolean	DBstsRDON DBstsRDEL DBstsINVC DBstsCNVR DBstsNODA	AfterSetValue BeforeSetValue

- **Result set row modification operations**: These methods enable you to dynamically add and delete rows from within the result set. Furthermore, you can retrieve the row status and you can update the altered result set in the database.

Method	*Argu-ment*	*Return Value Type*	*Error Constant*	*Event*
AddRow		Boolean	DBstsAHVR DBstsRDON DBstsNOEX	BeforeAddRow AfterAddRow
DeleteRow (tableName$)		Boolean	DBstsINVR DBstsNUNQ DBstsRCHG DBstsRDON	BeforeDeleteRow AfterDeleteRow RowContentsChanged TransactionsPending
GetRowStatus		DB_UNCHANGED DB_ALTERED DB_UPDATED DB_DELETED DB_NEWROW	DBstsNODA	
UpdateRow		Boolean	DBstsRDON DBstsRDEL DBstsCXIN DBstsNUNQ DBstsRCHG DBstsRUNC DBstsUPDB	BeforeUpdateRow AfterUpdateRow TransactionsPending RowContentsChanged

- **Column attributes operations**: These methods allow you to access information about the column attributes.

Method	*Argument*	*Return Value Type*	*Error Constant*
FieldExpected DataType (column [, dataType])	column is Integer or String. dataType: DB_TYPEUNDEFINED DB_CHAR DB_SHORT DB_LONG DB_DOUBLE DB_DATE DB_TIME DB_BINARY DB_BOOL DB_DATETIME	DB_TYPEUNDEFINED DB_CHAR DB_SHORT DB_LONG DB_DOUBLE DB_DATE DB_TIME DB_BINARY DB_BOOL DB_DATETIME	DBstsINVC
FieldID(column Name$)		Integer	DBstsINVC
FieldInfo(column)	column is Integer or String	Array with elements *1	DBstsINVC
FieldNativeData Type (columnID%)		String *2	DBstsINVC
FieldID(column)	column is Integer or String	Constant	DBstsINVC
FieldName (column)	column is Integer or String	Integer	DBstsINVC

- **SQL parameter operations**: These methods are used to define new SQL parameters and to retrieve the values of those already existing.

Method	*Argument*	*Return Value Type*	*Event*
GetParameter (parameter)	parameter is Integer or String	Variant	BeforeGetParameter AfterGetParameter
GetParameterName (parameterID%)		String	BeforeGetParameterName AfterGetParameterName
NumParameters		Integer	
SetParameter (parameter, value$)	parameter is Integer or String	Boolean	BeforeSetParameter AfterSetParameter

Error Operations
These methods are used to deal with error messages.

Method	*Argument*	*Return Value Type*
GetError		Constant *3
GetErrorMessage([error%])	error%: DB_LASTERROR or Constants *3	String
GetExtendedErrorMessage([error%])	error%: DB_LASTERROR or Constants *3	String

The following three tables contain information referenced in the above ODBCResultSet Class tables about

- Return Value Constants of the FieldInfo Method
- Return Value Constants of the FieldNativeDataType Method
- Error Constants

Reference *1 — Return Value Constants of the FieldInfo Method

DB_INFO_AUTOINCREMENT	DB_INFO_NULLABLE
DB_INFO_CASESENSITIVE	DB_INFO_PRECISION
DB_INFO_COLUMNID	DB_INFO_READONLY
DB_INFO_COLUMNNAME	DB_INFO_SCALE
DB_INFO_COMPUTED	DB_INFO_SEARCHABLE
DB_INFO_DISPLAYSIZE	DB_INFO_SETTABLE
DB_INFO_EXPECTED_DATATYPE	DB_INFO_SQLDATATYPE
DB_INFO_LENGTH	DB_INFO_TABLENAME
DB_INFO_MONEY	DB_INFO_UNSIGNED
DB_INFO_NATIVE_DATATYPE	

Reference *2 — Return Value Constants of the FieldNativeDataType Method

SQL_CHAR	SQL_FLOAT	SQL_TIMESTAMP	SQL_LONGVARBINARY
SQL_NUMERIC	SQL_REAL	SQL_VARCHAR	SQL_BIGINT
SQL_DECIMAL	SQL_DOUBLE	SQL_BINARY	SQL_TINYINT
SQL_INTEGER	SQL_DATE	SQL_VARBINARY	SQL_BIT
SQL_SMALLINT	SQL_TIME	SQL_LONGVARCHAR	

Reference *3 — Error Constants

DBstsSUCCESS	DBstsINVC	DBstsDSTY	DBstsTMPL	DBstsRDON
DBstsFAIL	DBstsNCOL	DBstsDRVN	DBstsBROW	DBstsRCHG
DBstsMEMF	DBstsBADP	DBstsFITY	DBstsCANF	DBstsRUNC
DBstsNCON	DBstsODBC	DBstsFILT	DBstsCNVR	DBstsCXIN
DBstsCCON	DBstsLIBM	DBstsINST	DBstsCNVD	DBstsAHVR
DBstsNOEX	DBstsSNFD	DBstsNODR	DBstsHSTMT	DBstsCPAR
DBstsINVR	DBstsINTR	DBstsNAUT	DBstsSQLP	DBstsNIRC
DBstsCARR	DBstsACCS	DBstsNOSV	DBstsINTE	DBstsRDEL
DBstsNODA	DBstsTYPE	DBstsNAPE	DBstsUPDB	
DBstsEOFD	DBstsENTR	DBstsSVRQ	DBstsNUNQ	

Following is a sample Program using the ODBCResultSet class.

```
Dim con As New ODBCConnection
Dim qry As New ODBCQuery
Dim res As New ODBCResultSet
Dim status As Variant
‘** sampleDB1 is a database name registered in this
example
status = con.ConnectTo(“sampleDB1”)
Set qry.Connection = con
‘** courses is a table name in the sampleDB1 database
qry.SQL = “select * from courses”Set res.Query = qry
Call res.execute
Dim num As Integer
num = 0
Dim vl As Variant
Redim vl(num)
Do
‘** name is a column name in the courses table
     vl(num) = res.GetValue(“name”)
     num = num + 1
     Redim Preserve vl(num)
Loop While res.NextRow
‘** CValue is a field name in which a value list is saved
gDoc.CValue = vl
```

Data Conversion between LotusScript and ODBC SQL

Every time data is moved from an enterprise system to Domino or a Notes client, it is converted from LotusScript to SQL or back. Some data types cannot be converted from one type to another. If you convert data to the wrong type or if the conversion results in truncation, it produces an error. This error can be trapped and handled like any error. This conversion happens when you use the GetValue and SetValue methods. The following

chart shows if a conversion is attempted by GetValue and SetValue. The SQL data types are in the left column and the LotusScript data types are across, each in its own column.

SQL Data Type	*String*	*Boolean*	*Short*	*Long*	*Single*	*Double*	*Currency*	*Date/Time*	*Variant*
CHAR	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
VARCHAR	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
LONGVAR CHAR	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DECIMAL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
NUMERIC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
TINYINT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
SMALLINT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
INTEGER	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
REAL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
FLOAT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
DOUBLE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
BIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
BINARY	Yes	No	No	No	No	No	No	No	Yes
VARBINARY	Yes	No	No	No	No	No	No	No	Yes
LONG VARBINARY	Yes	No	No	No	No	No	No	No	Yes
DATE	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TIME	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TIMESTAMP	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Updates to LSDO in Domino R5.0

Some methods have been added to LSDO in Domino R5.0 and some have been de-activated. Refer to the Domino R5.0 release notes for a full description of the changes. We will mention a few of the new methods here:

CommitTransactions and RollbackTransactions

CommitTransactions and RollbackTransactions are used when AutoCommit (moved from ODBCRecordset to ODBCConnection) is set to false to enter transaction mode. With AutoCommit set to true, the default, it is not necessary to use CommitTransaction after every update or delete (transaction) to make the changes permanent, as all transactions are automatically committed. You can only enter transaction mode when your ODBC driver and data source supports it. When you are in transaction

mode, the changes that your application make will only become permanent after a CommitTransactions method. If you do not want the changes to take effect, the changes can be rolled back with the RollbackTransactions method. In some applications, it is important for a series of transactions to be successful. If one of the transactions fail, all other related transactions must also fail. This is where you can use transaction mode to accomplish your goal.

```
Set Con = New ODBCConnection

' To enter transaction mode

Con.AutoCommit = False

' Your code goes here. If everything is good.

Con.CommitTransactions

' If everything is not good.

Con.RollbackTransactions
```

ExecProcedure
This method was added to ODBCResultset to enable you to execute a stored procedure with parameters. There are two formats for this method and the syntax is as follows.

Status = odbcResultSet.ExecProcedure(ProcedureName$ [,arg1] arg2]...[arg30])

Or

Status = _
odbcResultSet.ExecProcedure(ProcedureName$,DB_PARAM_ARRAY,argArray)

Parameters:
Status
Boolean. Value that will be returned to indicate the success of the operation. 1 indicates that the stored procedure was executed successfully.

procedureName$
String. The name of the SQL procedure you want to execute.

arg1..30
Up to 30 arguments can be passed to a procedure. The arguments can be in any format. An argument can serve as input, output, or both. Argument data types must be consistent with the requirements of the procedure. Separate arguments with commas. Any missing arguments are treated as NULL values. The 30 argument limit is a LotusScript limitation.

To enter more than 30 arguments, you can use the alternate syntax and pass the arguments in an array. In the alternate syntax, the second argument must contain the constant DB_PARAM_ARRAY. The third argument can be an array of any size or type.

Using @DB Functions to Access Other Databases through ODBC

The Domino functions @DBCommand, @DBLookup and @DBColumn enable you to access RDBMSs that use the underlying ODBC interface. The @DB formulas are read-only, but provide an easy to use access to ODBC-compliant databases.

The basic purpose of these functions is to create value lists for keyword fields. @DBLookup and @DBColumn can be used to query a relational database; @DBCommand is only used for executing stored procedures. @DBCommand does not return result sets. If you need a more customized and more complex query, LS:DO is a better option.

Note Some of these functions inherently involve a delay before they complete. To set user expectations, it is sometimes a good idea to code the functions behind a button so that the user expects some delay before the function is completed.

How to Use @DB Functions

The @DB functions are summarized in the following table.

Function	*Description*	*Equivalent SQL*
@DBColumn	Generates a keyword list. Returns a specified column for all rows in the specified table.	SELECT DISTINCT column_name FROM table_name
@DBLookup	Performs a lookup. Returns a specified column value in the row that matches the specified condition.	SELECT column FROM table WHERE condition
@DBCommand	Triggers stored procedures in the external database.	(any SQL statement)

@DBColumn

The @DBColumn syntax is:

```
@DBColumn( "ODBC": Cache ; DataSource ; UserID1 : UserID2 ;
Password1 : Password2 ; TableName ; ColumnName : NullHandling ;
Distinct : Sort )
```

Parameters:

@DBColumn("ODBC":	*Description*	*Choice*	*Optional*
Cache,	Inquiry Cache	"Cache" (Default) "NoCache"	X
DataSource,	Database resource name		
UserID1:UserID2,	User IDs		X
Password1:Password2,	Passwords		X
TableName,	Table Name		
ColumnName:	Column Name		
NULLHandling,	Null Handling	"Fail" "Discard" (Default) "ReplacementValue"	X
Distinct:	Remove duplicate values	"Distinct"	X
Sort)	Sort Direction	"Ascending" "Descending"	X

@DBLookup

The @DBLookup syntax is:

```
@DBLookup( "ODBC": Cache ; DataSource ; UserID1 : UserID2 ;
Password1 : Password2 ; TableName ; ColumnName :
NullHandling ; KeyColumn ; Key ; Distinct : Sort )
```

Parameters:

@DBLookup("ODBC":	*Description*	*Choice*	*Optional*
Cache,	Inquiry Cache	"Cache" (Default) "NoCache"	X
DataSource,	Database resource name		
UserID1:UserID2,	User IDs		X
Password1:Password2,	Passwords		X
TableName,	Table Name		
ColumnName:	Column Name		
NULLHandling,	Null Handling	"Fail" "Discard" (Default) "ReplacementValue"	X

continued

@DBLookup("ODBC":	*Description*	*Choice*	*Optional*
KeyColumn,	Column Name to be looked into		
Key,	Search String in KeyColumn		
Distinct:	Remove duplicate values	"Distinct"	X
Sort)	Sort Direction	"Ascending" "Descending"	X

@DBCommand

The @DBCommand syntax is:

```
@DBCommand( "ODBC": Cache ; DataSource ; UserID1 : UserID2 ;
Password1 : Password2 ; SQL ; NullHandling )
```

Parameters:

@DBCommand("ODBC":	*Description*	*Choice*	*Optional*
Cache,	Inquiry Cache	"Cache" (Default) "NoCache"	X
DataSource,	Database resource name		
UserID1:UserID2,	User IDs		X
Password1:Password2,	Passwords		X
SQL	SQL Statement		
NULLHandling)	Null Handling	"Fail" "Discard" (Default) "ReplacementValue"	X

DB2LSX

The DB2LSX provides native access to DB2 through the DB2 CLI. Based on the same model as the LS:DO, the DB2 LSX is a set of three classes:

- DB2Connection
- DB2Query
- DB2ResultSet

These classes access DB2 data natively, as well as enabling you to use DB2 extended functions, such as support for DB2 user-defined data types and large objects.

The DB2 data access classes provide properties and methods for accessing and updating tables in external databases. These classes are packaged as an LSX file called the DB2LSX. They provide access to DB2 through the DB2 CLI and to non-DB2 databases through the ODBC Version 2.0 standard.

Similar to ODBC, the CLI is a callable API for database access and is an alternative to the embedded SQL API. It is supported by the DB2 family of database servers.

DB2LSX and LS:DO are similar. You can use either to access DB2 databases. There are, however, a few key differences is shown in the following table.

	DB2LSX	*LS:DO*
Data access	Text, number, date, large objects, user-defined data, binary data	Text, number, date
Driver	Accesses DB2 through DB2 CLI	Accesses DB2 through DB2 ODBC
Change and add data	No autocommit by default Use the Transact, Commit, and Rollback method	Autocommit by default when UpdateRow or DeleteRow is executed. Use CommitTransactions and RollbackTransactions methods

You can use DB2LSX to work with DB2 large objects, user-defined data, and binary data. With LS:DO, you access DB2, using ODBC. With DB2LSX, you access DB2, using the DB2 CLI.

By default under DB2LSX, additions and changes are not final in the database until you use the Transact method on the DB2Connection object. Using LS:DO, additions and changes are committed when the UpdateRow or DeleteRow methods are executed; that is, the default for autocommit is OFF for DB2LSX and ON for the LS:DO.

Although DB2 access is native, DB2LSX Version 1.1 requires that you register a DB2 database as an ODBC source. DB2LSX Version 1.1 must know before it makes a connection whether the source is DB2 or not. Putting the DB2 database in the ODBC registry provides that information. If the registry indicates that the source is a DB2 database, the ODBC driver manager is not used. The DB2 CLI is used instead. DB2LSX Version 1.2 does not have this requirement.

DB2 LotusScript Data Objects DB2 LSX update

DB2 LSX release 2 can be downloaded via the Lotus Enterprise Integration Web site

```
http://www.lotus.com/enterpriseintegration
```

This release also includes a samples database. The change and enhancements over the previous release are the same as for LS:DO, described previously, and also include the following features.

AutoCommit

In DB2LSX, AutoCommit defaults to false and must be switched on explicitly; otherwise, the connection will operate in transaction mode and no changes will be made to the database without the Transact method. An example follows.

```
Set Con = New ODBCConnection

‘ To switch from transaction mode to AutoCommit mode

Con.AutoCommit = True

‘ Your code goes here.

‘ No need to use the next 2 commands

‘ Con.Transact(DB_COMMIT)
‘ Con.Transact(DB_ROLLBACK)

‘ All Commands will be automatically committed
```

Multithreading

Support for multithreading was added in release 2.

CLOB data type

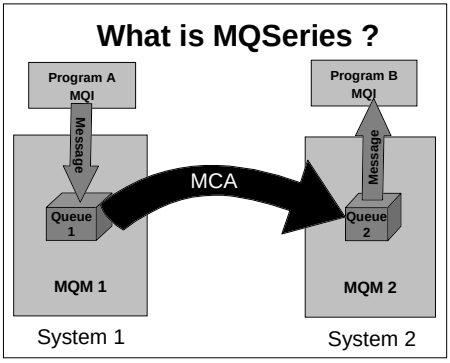
Better support for handling a Character Large Object (CLOB) was added in release 2.

BLOB data type

Better support for handling a Binary Large Object (BLOB) was added in release 2.

MQSeries Integration

MQSeries is IBM's Message Queuing middleware product. It allows two (or more) applications on disparate systems to talk by sending messages.



MQSeries uses a combination of programming (MQSeries API or MQI), queues (MQSeries Queue Manager or MQM) and channels (MQSeries Channel Architecture or MCA) to allow this exchange to occur. The previous figure shows how Program A, written using the MQI, sends a message to a local queue (Queue 1) on the local Queue Manager (MQM 1). The local Queue Manager (MQM 1) will transmit the message across the MCA to the remote queue (Queue 2) on the remote Queue Manager (MQM 2). Program B, also written using the MQI, will retrieve the message and perform any necessary processing. For further details on MQSeries, visit the following IBM Web site:

`http://www.software.ibm.com/ts/mqseries`

This section discusses product updates that have occurred since 1997. The discussion focuses on:

- MQSeries Link LotusScript Extension - MQLSX Version 1.3.2
- MQSeries Trigger Monitor for Lotus Notes Agents - Version 1.1
- MQSeries Enterprise Integrator for Lotus Notes - MQEI - Version 1.0a
- MQLSX versus MQEI

This section assumes that you have previous experience with MQSeries and LotusScript functions and terminology.

MQSeries Link LotusScript Extension (MQLSX)

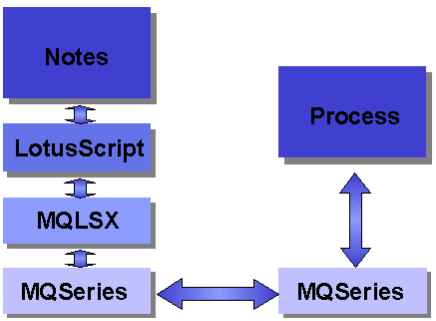
The MQSeries LotusScript Extension allows you to integrate your Domino application with MQSeries. MQSeries must be installed on your Domino server or on your Notes client. The version and type of MQSeries you install is up to you and your installation architecture. The MQLSX currently

supports the full function of MQSeries V2.0, but not the enhanced function of MQSeries V5.0. However, your MQLSX will still work in version 5 of MQSeries. As for the type of MQSeries product to install, this depends on your infrastructure and integrity requirements. You can either install the MQSeries thin-client with Domino or you can install the full MQSeries Server with Domino. There are advantages to either decision and each situation must be explored to weigh those advantages. These considerations are not discussed in this section, as the advantages and disadvantages are dynamic and depend on the systems you are using and your network infrastructure (that is, the topology, architecture and protocol used).

What is the MQSeries LotusScript Extension?

The MQSeries LotusScript Extension (MQLSX) provides a tight link through MQSeries to transaction-based systems, allowing seamless integration of business processes across Domino and transaction systems.

MQLSX enables you to make MQSeries calls directly from your LotusScript code. The following figure shows the architecture of the MQSeries LSX.



With the MQSeries LSX, you can develop applications that integrate the Domino environment with transaction-based applications and legacy system data through MQSeries. This allows you to access enterprise application logic, not just the data, through a variety of platforms since you have direct integration between Domino and MQSeries.

Obtaining MQLSX

MQLSX is a SupportPac available from IBM. It is also available as a component of Lotus' MQSeries and CICS Connections for Domino. From the Lotus Enterprise Integration Web site, chose MQSeries and you will be lead to the download page for MQLSX. You can also try to link directly to the IBM download page:

`http://www.software.ibm.com/ts/mqseries/txppacs/NNNN.html`

where *NNNN* is the SupportPac name (for Windows 32-bit, this is SupportPac MA7D). Download the zip file, unzip the file and follow the directions in the README.TXT file (assuming you downloaded MA7D. Other SupportPacs follow similar guidelines).

Note The specified Web link for download of MQSeries SupportPacs was valid at the time this book was written. It may change in the future. In this case you should go to the IBM Web site and search for the name of the SupportPac.

Installing MQLSX

The provided readme file has all the information required to obtain the installation instructions for MQLSX. You will be directed to the MQLSX User guide for the installation process (for MA7D, see gmqlhelp.pdf or gmqlhelp.nsf). Remember, when installing from MQLSX 1.3 or later, select the update option. Previous releases require that you select the delete and reinstall option.

There have been significant changes from previous releases. Before you upgrade MQLSX, you must adhere to the recommendations in the readme. Notably:

- The GMQ_XLAT_PATH variable is now mandatory.
- Data conversion tables have been changed, which forces you to use the new CCSID.TBL table.
- The GMQ_PATH environment variable has been changed to GMQ_TRACE_PATH.
- You must use the MQQueueManager disconnect method for syncpoint messaging to force the syncpoint.

Using MQLSX.

Once you have installed the new version of the MQLSX, test your existing code to ensure it works. Here are some items to consider when testing and maintaining your program:

- MQLSX 1.3.2 must be used in Lotus Notes R4.5 or higher. For Windows 32 bit, it has been tested against Notes 4.6.1.
- The GMQ_MQ_LIB environment variable is used to override the search order when the shared library for MQLSX is dynamically loaded.
- MQLSX agents can run multi-threaded using the DominoAsynchronizeAgents NOTES.INI parameter (set to 1). Refer to the MQLSX readme file for more detail on multi-threaded support.

- Be careful with other NOTES.INI settings. MQLSX recommends that its agents run with parameters AMgr_DocUpdateMinInterval and AMgr_DocUpdateEventDelay set to 1. These settings create a substantial overhead on Domino. You should optimize these parameters by analyzing your agent load and triggering activity (that is, how many messages are processed per run and how often do requests come in).
- The latest MQLSX causes your LotusScript object code to grow, without any change to your LotusScript. This may cause the error

  ```
  Cannot forward declare CLASS or TYPE.
  ```

  This error occurs for larger MQLSX programs that make effective use of the declaration and initialize events in LotusScript. When the interpreter attempts to load private classes, it is unable to because the Agent Manager limits the executable thread to 64K of object code. If you were to debug the agent on the server in a Designer session, it would work, since the Agent Manager is not used for debugging. Recompiling your LotusScript may solve this problem. If not, start removing unnecessary code (such as, CONST declarations and print statements to the log).

MQLSX Class Overview

There are seven MQSeries LSX classes:

- MQSession
- MQQueueManager
- MQQueue
- MQProcess
- MQMessage
- MQGetMessageOptions
- MQPutMessageOptions

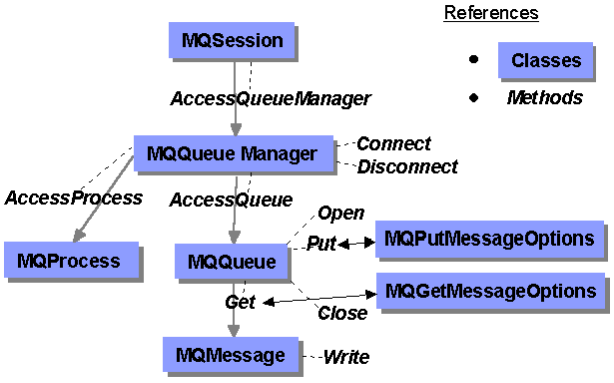
In addition, the MQSeries LSX also provides predefined LotusScript constants, such as MQFMT_NONE, which are used by the classes. The constants are a subset of those defined in the MQSeries C header files, namely CMQC*.H, with some additional IBM MQSeries LSX reason codes.

For these classes, we can categorize them in three groups:

- The first group contains one class which is the MQSession Class. It provides a root object that contains the status of the last action performed on any of the MQSeries LSX objects.
- The second group contains the MQQueueManager, MQQueue and MQProcess classes. They provide access to the underlying IBM MQSeries objects. Methods or properties defined in these classes will, when used, result in calls being made across the MQI.

- The third group contains the MQMessage, MQPutMessageOptions and MQGetMessageOptions classes, which encapsulate the MQMD, MQPMO and MQGMO data structures respectively and are used to help you put messages to queues and retrieve messages from them.

An effective class relationship is presented in the following figure.



In reference to this class relationship diagram, the following list will help identify the components:

- The MQSession class defines an instance of MQSeries access. This class manages all calls and error conditions. Its AccessQueueManager method is used to set the MQQueueManager class object.
- The MQQueueManager class is used to access the MQSeries Queue Manager (MQM). This is performed by the Connect method for access and the Disconnect method to indicate completion.

  Note The Disconnect method must be used in the newer version of the MQLSX for syncpoint control. Please ensure you code this statement.
- The MQProcess class object is set by the AccessProcess method of the MQQueueManager class. The MQProcess class enables you to read the MQSeries process definitions.
- The MQQueue class object is set by the AccessQueue method of the MQQueueManager class. The MQQueue class allows you to manipulate your queues. You must use the Open method to access your queue, and when you are done, use the Close method to close (commit) your work.
- The MQPutMessageOptions class, along with the Put method of the MQQueue class, define the complete put operation to the MQ Queue.
- The MQGetMessageOptions class, along with the Get method of the MQQueue class, define the complete get operation from the MQ Queue.

- The MQMessage class contains the Write and Read methods that enable you to manipulate the message buffer. This buffer is used by the Put and Get operations as the storage area for your message content.

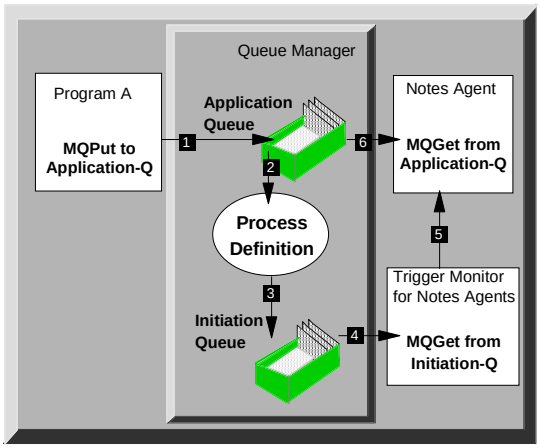
Getting Help for MQLSX

SupportPac help can be obtained through your regular support channels for MQSeries. You can also participate in the MQForum under the Discussions link at Lotus Enterprise Integration Web site:

`http://www.lotus.com/enterpriseintegration`

MQSeries Trigger Monitor (MQTM) for Lotus Notes Agents

The MQSeries Trigger Monitor for Lotus Notes is modified from the standard MQSeries Trigger Monitors such that it allows you to trigger Lotus Notes agents where Domino is running with the MQ Server or MQ Client. The trigger monitor plays a vital role in MQSeries-Domino applications because it enables action by Domino without user intervention. An external system can trigger an activity in the Domino environment.



The previous figure represents how MQSeries triggering works as it applies to the Trigger Monitor for Lotus Notes Agents. The process works as follows:

1. A message is written by Program A to an application queue (either remotely via a remote Queue Manager and Channel, or locally direct to the queue). The application queue must be defined as a trigger queue. The application queue definition includes the trigger type, the initiation queue, the process definition and optional trigger data.
2. The Queue Manager recognizes the application queue as a trigger queue, so it reads the process and initiation queue information on the application queue definition. With this information it reads the Process definition to determine what to include on the initiation message.

3. The Queue Manager writes the initiation message onto the initiation queue. The initiation message contains information regarding the process the trigger monitor is to perform. The process information came from the Process Definition which contains the Notes Database name and Notes Agent name, any optional agent input parameters and an indicator recognized by MQSeries identifying the process as a Notes agent.
4. The trigger monitor is a long-running operating system task that monitors an initiation queue. When the initiation message appears on the initiation queue, the trigger monitor "wakes up" and reads the initiation queue.
5. The process information contained on the initiation message is interpreted by the trigger monitor and it starts the Notes agent defined by the process.
6. The defined Notes agent will run. The agent program will "get" the message from the application queue (step 1) and perform the specified business process.

Obtaining the MQTM for Lotus Notes Agents

MQTM for Lotus Notes Agents is a SupportPac available from IBM. From the Lotus Enterprise Integration Web site, chose MQSeries and you will be led to the download page for MQTM. You can also try to link directly to the IBM download page:

```
http://www.software.ibm.com/ts/mqseries/txppacs/NNNN.html
```

where *NNNN* is the SupportPac name (for Windows NT, this is SupportPac MA7E). Download the zip file, unzip the file and follow the directions in the README.TXT file (assuming you downloaded MA7E. Other SupportPacs follow similar guidelines).

Installing the MQTM for Lotus Notes Agents

The MQTM for Lotus Notes Agents (version 1.1) is installed by following the instructions in the SupportPac readme file.

Note For version 1.1 of the Notes agent trigger monitor, the install instructions in the readme file must be followed. The user guide is still at the version 1.0 level and contains the old install process.

These instructions are summarized as follows:

1. Select START - RUN from your NT server desktop and enter the drive and path where the unzipped contents of your download were stored. Select the SETUP program and click OK.

   Note For operating systems other than NT, refer to the appropriate readme file for instructions.

2. At this point, you can select all the defaults and easily navigate through the screens until the install is complete.

The install process will create a new subdirectory under your existing MQSeries directory (MQM\MQTM). Verify that your path environment variable contains this entry.

Under the MQTM subdirectory, you will find the following files:

Directory	Filename	*Description*
root	README.TXT	Information regarding the product that has become available after the product documentation was created.
docs	gmqltmon.ps	*The MQSeries Trigger Monitor for Lotus Notes Agents User's Guide* as a postscript file.
	gmqltmon.pdf	*The MQSeries Trigger Monitor for Lotus Notes Agents User's Guide* as an Adobe Acrobat PDF file.
samples	gmqlumsg.nsf	A Notes database that contains English (US) messages required by the MQTM for Lotus Notes Agents.
	gmqstnm0.tst	MQSC Command File that provides sample queue, initiation queue and process queue definitions for triggering Lotus Notes Agents. This file is included in the "MQSC Sample File for the MQSeries Trigger Monitor for Lotus Notes Agents" Appendix.
bin	runmqtnm.exe	The executable file for the MQTM for Lotus Notes Agent's task. This program can only run with the MQSeries Server. It will not execute on Windows 95 or 98.
	runmqtnc.exe	The executable file for the MQTM for Lotus Notes Agent's task. This program can only run with the MQSeries thin-client.

The above table defines the MQTM for Lotus Notes Agents package.

Using the MQTM for Lotus Notes Agents

Before running the trigger monitor on your MQSeries system, you must define the following:

- A Notes application that contains a LotusScript agent written using the MQLSX or MQEI, so that the application message can be retrieved to allow for continuous monitoring.
- The application queue, initiation queue, and process queue using MQSC via the runmqsc command. You can use the gmqstnm0.tst file as a sample. The command to create the queues is:

```
runmqsc YourQueueManagerName <gmqstnm0.tst
```

- Any environment variables settings. Refer to the *MQSeries Trigger Monitor for Lotus Notes Agent's User Guide* for further detail.

Starting the MQTM for Lotus Notes Agents

Once the appropriate definitions are in place, you can start the trigger monitor using the following command:

```
runmqtnm -m YourQueueManagerName -q YourInitiationQueueName
```

Note Substitute the names of your queue manager and initiation queue where indicated.

Note If you are running the MQSeries Client, replace runmqtnm with runmqtnc.

Tip Ensure you have a copy of the *MQSeries Trigger Monitor for Lotus Notes Agent's User Guide* at your side when you are starting the trigger monitor for the first time. Once you have your trigger monitor running, consider creating a startup folder task (or equivalent startup task for other operating systems) that will execute the runmqtnm or runmqtnc command whenever the Domino server machine is rebooted.

Stopping the MQTM for Lotus Notes Agents

There are two ways that you can stop the trigger monitor after it begins running:

- Stop your Queue Manager (the drastic but easy approach!)

  Quite simply, for a single queue manager server, enter:

  ```
  ENDMQM
  ```

  For a multiple queue manager server, enter:

  ```
  ENDMQM -i YourQueueManagerName
  ```

  These two commands will quiesce the server until all message handling has stopped. Once the queue manager is idle, it will be stopped by the server. Any tasks attached to the queue manager will be stopped.

- Disable the GET property on your initiation queue (the involved approach!)

  In this case, enter the MQSC environment by entering the following command at a DOS prompt:

  ```
  runmqsc
  ```

  Inside the MQSC environment, enter the following command:

  ```
  ALTER QLOCAL(YourInitiationQueueNameInCaps) GET(DISABLED)
  ```

  Note Your initiation queue must be entered in caps (as indicated) since MQSeries objects must be referenced in upper case.

Once the trigger monitor ends, you must re-enable the GET property before issuing the next **runmqtnm** or **runmqtnc.** This is also done within the MQSC environment by issuing another ALTE**R** command as follows:

```
ALTER QLOCAL(YourInitiationQueueNameInCaps) GET(ENABLED)
```

At this point, you are ready to restart the trigger monitor.

Getting Help for the MQTM for Lotus Notes Agents

SupportPac help can be obtained through your regular support channels for MQSeries. You can also participate in the MQForum under the Discussions link at the Lotus Enterprise Integration Web site. Other options for support of this trigger monitor include the MQSeries User groups and IBM's Talk Link MQSeries discussion.

MQSeries Enterprise Integrator for Lotus Notes (MQEI)

MQSeries Enterprise Integrator (MQEI) is currently at the version 1.0a level. The "a" level provided fixes from the Release 1.0 version and added Double Byte Character Support (DBCS).

This section provides a brief overview of MQEI and its ability to integrate with MQSeries, CICS and IMS. We also discuss a new add-on product feature to the MQEI. It is called MQEIQuick and is an application builder for MQEI that eliminates the need to do any LotusScript programming.

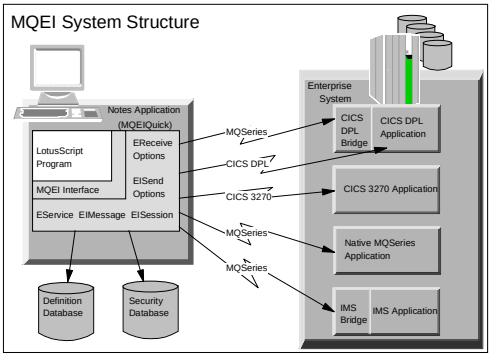
If you require further details on MQEI, refer to the redbook *Lotus Notes and the MQSeries Enterprise Integrator*, Lotus part number 12996, IBM form number SG24-2217.

What is MQEI?

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

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

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



The structure and components of MQEI are shown in the previous figure.

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

Objects from the EIService and EIMessage classes derive their characteristics from tables held in the MQEI Definition database. This database provides a library of predefined objects; the definition for an EIService object contains information such as the type of transaction system on which it runs, and the communications mechanism used to access it.

The LotusScript programmer does not have to be aware of this information, as it does not have to be coded in the LotusScript application itself.

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

The MQEI Definition database also contains definitions for the EIMessage objects required. An EIMessage object consists of a set of typed, named fields. These fields are presented to the LotusScript programmer as properties of the EIMessage object. The LotusScript programmer can read and write fields in the message, using simple assignment statements,

knowing nothing more than the names of the fields in question. This is directly analogous to the interface that Lotus Notes itself provides to the individual fields within a document.

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

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

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

MQEI can access the following enterprise applications:

- Native MQSeries applications. MQSeries is required on both source and target platforms.
- IMS applications. The MQSeries IMS/ESA Bridge is required, as well as MQSeries and IMS/ESA.
- CICS DPL applications via MQSeries. The MQSeries CICS/ESA Bridge is required, as well as MQSeries and CICS. The MQSeries CICS/ESA Bridge now supports both DPL bridging and 3270 bridging; however, MQEI can only integrate to the DPL bridge.
- Native CICS DPL or 3270 applications. A CICS Client is required.

Obtaining MQEI and MQEIQuick.

MQEI is available by purchasing Lotus' *MQSeries and CICS Connections for Domino*. Visit the Lotus Enterprise Integration Web site:

`http://www.lotus.com/enterpriseintegration`

and look under MQSeries for information on how to order MQEI. Evaluation copies are also available.

MQEIQuick is a Support Pac available from IBM. From the Lotus Enterprise Integration Web site, follow the same path as for the MQEI above. MQEIQuick will be available under companion products for MQEI. You can also go directly to to this address:

```
http://www.software.ibm.com/ts/mqseries/txppacs/NNNN.html
```

where *NNNN* is the Support Pac named MA0A. Download the zip file, unzip the file and follow the directions in the README.QIK file.

Installing MQEI and MQEIQuick

MQEI is installed as part of a selected component of Lotus' *MQSeries and CICS Connections for Domino*. Follow the install instructions provided with this package. If you downloaded the evaluation version, follow the installation documentation provided in the MQEI user guide.

Besides the readme file and the license agreement text, MQEIQuick contains one Notes database, MQEIQUIK.NSF, and its template, MQEIQUIK.NTF. There is no platform dependence on these databases.

Note MQEIQuick should work for all supported MQEI platforms; however, it was tested on Windows NT and IBM OS/2 Warp4 only.

There is no formal installation process for MQEIQuick. You simply copy the Notes database and template to your Notes data directory. When you create an instant application, copy the MQEIQUIK.NSF database to your own application database (or use the template to create a new database) and then use your application database to create the instant MQEI application.

If you are installing a new version of MQEIQuick, back up the existing MQEIQUIK.NSF file if you have made direct design changes to it. You will have to apply your custom changes back into the MQEIQUIK.NTF file supplied with the upgrade. Once an upgraded version is installed, select your MQEIQuick application and select to replace its design (File-Database-Replace Design) to take advantage of the new changes. When replacing the design, select the new MQEIQUIK.NTF template as your new database design.

Using MQEI and MQEIQuick

Directions on how to set up and use MQEI and the MQEIQuick database are well documented in the MQEI User Guide (for MQEI) and the About document of the MQEIQUIK.NSF database (for MQEIQuick).

Getting Help for MQEI and MQEIQuick

Help for MQEI and MQEIQuick can be obtained through your Lotus support channels for *MQSeries and CICS Connections for Domino*. You can also participate in the MQForum under the Discussions link at the Lotus Enterprise Integration Web site.

MQLSX or MQEI?

The "Which Tool to Use When" chapter of this redbook presents the overall considerations regarding which tool to use. This section deals specifically with MQLSX and MQEI and provides more detail about them. The following table compares several significant features of these two tools.

Comparable Aspects	MQLSX	*MQEI*
Application Type	- Native MQSeries application - IMS access through the MQSeries-IMS Bridge - CICS DPL access through the MQSeries-CICS/ESA DPL bridge - CICS 3270 access through the MQSeries-CICS/ESA 3270 bridge - SAP R/3 through MQSeries	- Native MQSeries application - IMS access through the MQSeries-IMS bridge - CICS DPL access through the MQSeries-CICS/ESA DPL bridge - CICS DPL direct - CICS 3270 direct
Programming on Lotus Notes	Full LotusScript environment with all the functions of the MQSeries MQI.	Full LotusScript environment with the most commonly used functions of the MQSeries MQI, plus most of the CICS ECI and EPI functions.
Security	Implemented in the LotusScript program.	Implemented through the Security database of MQEI, or in the LotusScript program.
Messages	Implemented in the LotusScript program. Field offsets must be known in the LotusScript program.	Implemented through the Definition database of MQEI. No need for the LotusScript program to know the field offsets (but you still need to know the offsets to define the format in the Definition database).
Abstraction for the enterprise system	You need to understand the enterprise system to which you connect. Bridge products will reduce this complexity.	Provides a higher level of abstraction. It is easier to program with the MQEI.

In general, this table indicates that the MQEI provides an easier interface for message formatting and it provides inherent security. MQEI together with MQEIQuick eliminates the programming element for connecting to MQSeries, IMS, and CICS. MQLSX gives you the flexibility of programming in LotusScript. If MQEI does not support your custom requirements, use MQLSX.

Note Make sure you verify platform availability of the two products, together with their required accompanying products. The MQLSX is available on more platforms and has less dependency on partner products since it deals with MQSeries only.

SAP R/3 Integration

Lotus provides an SAP R/3 connector that can be used by DECS, LEI, LCLSX, and the LC Java classes. The previously available LSX for SAP R/3 that works with SAP Remote Function Calls (RFC) is updated. Lotus provides workflow integration for Domino and SAP R/3. Mail integration is provided through the Domino SMTP Gateway and an MTA for SAP R/3.

SAP R/3 Connector

The Lotus connector for SAP R/3 uses SAP's Remote Function Call (RFC) interface, reading and writing R/3 data through the application, preserving the integrity of the application logic and R/3 processes for accessing underlying database tables. The connector maintains all the business rules provided by RFCs and R/3 transactions, and you can create your own RFC functions.

See the "Lotus Connectors" appendix for a brief description of the connector. See "LEI Documentation" (LEIDOC.NSF) for detailed descriptions of the connector properties and data types.

The SAP R/3 Connector is a separate Lotus product. For product information and technical papers, visit the Lotus Enterprise Integration Web site at

`http://www.lotus.com/enterpriseintegration`

You can use this connector with DECS, LEI, and the LC LotusScript and Java classes.

SAP R/3 2.0 LotusScript Extension

The SAP R/3 LSX, available since 1997 from Lotus and SAP, adopts the SAP RFC object model, providing specific, customized object-model integration with SAP R/3. This LSX wraps the SAP Remote Function Call Software Development Kit (RFCSDK) into LotusScript objects, allowing you to make

any RFC from LotusScript. The LSX works directly with SAP R/3 and does not go through the Lotus connector.

The SAP R/3 LSX, with documentation and examples, is available from Lotus at

```
http://www.lotus.com/enterpriseintegration
```

The LSX is also on the SAP R/3 release CDs. The SAP R/3 2.0 LSX is available for Windows 95, NT, OS/2, Sun Solaris, HP-UX, AIX, and AS/400.

Following is some detail regarding the classes available with the SAP R/3 LSX.

SAP R/3 Classes

Once you have downloaded and installed the LSX for SAP R/3, put the following statement in the Options event of your LotusScript program to access the SAP R/3 classes:

```
Uselsx "*RFC"
```

The following classes become available. For detailed information on the classes, see the SAP R/3 documentation.

- **RfcServer**: The base object required to access the SAP system. This class contains properties that describe the SAP Application Server, and maintains status information about the session.
- **RfcFunction**: Contains everything you need to make an RFC call, including the import and export of tables.
- **RfcParameter**: Edits RfcFunction import and export parameters.
- **RfcStructure**: Provides a more complex environment than RfcFunction for import and export parameters.
- **RfcCollection**: Collects objects of any Rfc type except RfcServer and RfcRow.
- **RfcTable**: The parallel object to a SAP table. It contains columns and rows. Depending on context, a table may serve as input or output to an RFC function.
- **RfcRow**: Lets you access a row in an RfcTable object.
- **RfcRowCollection**: A collection of rows.
- **RfcTransaction**: Supports batch mode transaction processing on SAP R/3.
- **RfcScreen**: Describes one screen within a transaction.
- **RfcField**: Refers to fields in a transaction screen.
- **RfcBO**: New with Release 2.0. See the next section.
- **RfcBOMethod**: New with Release 2.0. See the next section.

Updates for SAP R/3 LSX 2.0

Described here are the updates for the newly released SAP R/3 LSX 2.0.

RfcBO class

Two new classes, RfcBO and RfcBOMethod, provide easier access to R/3 Business Objects, described in the SAP help and the SAP Web page

```
http://www.sap-ag.de
```

Included Business Objects and collections are not expanded during RfcBO object creation; they are not exposed until accessed, for reasons of performance and overhead.

The syntax of the New method for RfcBO follows. The first parameter identifies the RFC server, the second parameter identifies the Business Object by name, and the third parameter identifies an instance of the object.

```
Set bo = New RfcBO(srv As RfcServer, ByVal objName As String,
ByVal Optional key As String)
```

If the Business Object is not found, the object is not fully initialized. Check the Status property.

Here are two examples:

```
Dim server As RfcServer
Dim employee As RfcBO
...
Set employee = New RfcBO(server, “Employee”)
```

```
Dim server As RfcServer
Dim employee As RfcBO
...
Dim employee1000 As New RfcBO(server, _
“Employee”, 00001000)
```

RfcBOMethod class

You can call Business Object methods not implemented as BAPIs using the RfcBOMethod class. See the R/3 browser for method details.

Some methods launch the SAPGUI. The client must have a SAPGUI installed in these cases. Set the Enablesapgui property of RfcServer to True before logging in.

The following example calls the Edit method if the Business Object has this method and the method is of type RfcBOMethod:

```
Forall method In bo.Methods
    If Ucase(method.Name) = “EDIT” Then
        If Ucase(Typename(method)) = “RFCBOMETHOD” Then
            Call method.Call
        End If
```

```
    End If
End Forall
```

Enhanced Memory

Memory management enhancements improve performance, particularly for accessing table objects.

Lotus Domino Integration for SAP R/3 Business Workflow

Domino and SAP R/3 have different models for workflow. Lotus Domino Integration for SAP R/3 Business Workflow is a product that connects the models by rendering R/3 workflow data to Domino mail users, providing a central inbox for users to receive and act upon workflow data. A worklist item processed in R/3 (for example, a credit verification requiring manager approval) can automatically result in a notification to a Domino mail user. Special buttons in Domino mail allow the user to work interactively with the R/3 data, approving, rejecting, or modifying the R/3 worklist item. This eliminates the need to check several inboxes for worklist data requiring user intervention.

Lotus Domino Integration for SAP R/3 Business Workflow consists of two Domino databases: the administration database and the workflow database.

Lotus Domino Integration for SAP R/3 Business Workflow, with documentation and technical papers, is available for free from Lotus at

`http://www.lotus.com/enterpriseintegration`

Administration Database

The *Administration Database* (R3ADMIN.NSF) allows a central Domino administrator to decide who can use the workflow integration product. You send mail from the administration database to the user to control R/3 workitem connections.

Mail Template

The *R/3 Workflow Enabled Mail Template* (R3WORK.NTF) contains code and forms that allow R/3 workitems to be downloaded into a Domino user's mail file. These work items appear as memos from the R/3 system. They appear in Inbox as well as the SAP Workitems view. The Domino user sees them as individual memos that indicate the status of the worklist item.

The user can process workitems from Domino using Action Buttons to invoke the SAPGUI, placing the end user directly in the SAPGUI screens required to further process the item.

Mail Integration

Domino provides an SMTP Gateway for Internet mail and a specific gateway for transfer of SAPOffice R/3 mail to Domino using the Domino MTA for R/3 product.

SMTP MTA

Domino and SAP R/3 both support SMPT. This provides a standards-based approach to exchanging mail between the systems.

Lotus Domino MTA for SAP R/3

Lotus and SAP jointly developed an MTA based on SAP Connect technology. This MTA provides robust logging and tracking, and very good fidelity of rich text attachment transfer support, an advantage over generic mail transfer agent technologies.

The Domino MTA for SAP R/3 is available for free from Lotus at

`http://www.lotus.com/enterpriseintegration`

SAP provides a specific RFC (SAPconnect) for integration with communication systems including mail and facsimile. Lotus uses these RFCs to provide customized integration between SAPOffice and Domino mail for users and applications.

Mail created in SAPOffice goes to the corresponding Domino mail user. Mail created in Domino with an SAP R/3 destination goes to the corresponding R/3 end user. The MTA works on SAP R/3 Release 3.1G and above servers, and Domino Release 4.0 and above servers.

The Domino MTA for SAP R/3 consists of two add-in tasks:

- **Object Client**: Sends mail from Domino to SAP R/3 locations. The mail database SAPConn.nsf collects Domino mail messages to be routed to R/3 servers. A foreign domain for SAP is established in the Domino Directory.
- **Object Server**: Receives mail from R/3 sources. The R/3 mail goes to a special Domino database for conversion to Domino mail formats. The mail is then transferred to a Domino server mailbox and routed to the user or application.

Summary

In this chapter we described some of the enterprise integration tools from Lotus, specifically:

- LotusScript Data Object (LSDO)
- @DBColumn, @DBLookup and @DBCommand
- DB2LSX
- MQSeries Integration (MQLSX, MQTM, MQEI)
- SAP R/3 Integration (SAP R3 LSX, Workflow, MTA)

Chapter 6
Additional Integration Methods

This chapter describes additional integration methods that enable you to connect Domino to enterprise data. These include:

- Domino Driver for JDBC
- Servlets
- CORBA
- ADO and OLEDB
- NotesSQL
- RunOnServer agent method

Java and Domino R5.0

Domino R5.0 contains various Java enhancements. This section will look at Domino Driver for JDBC, and servlets.

For more information on connecting to Domino with Java, please refer to the redbook, *Connecting to Domino to the Enterprise using Java*, Lotus part number CT6EMNA, IBM form number SG24-5425.

What is JDBC?

The JDBC (Java Database Connectivity) API defines Java classes to represent database connections, SQL statements, result sets, database metadata, and so forth. It allows a Java programmer to issue SQL statements and process the results. JDBC is the primary API for database access in Java.

The JDBC API is implemented via a driver manager that can support multiple drivers connecting to different databases. JDBC drivers can either be written entirely in Java so that they can be downloaded as part of an applet, or they can be implemented using native methods to bridge to existing database access libraries.

The JDK 1.1 and the JDK 1.2 contain both JDBC and the JDBC-ODBC bridge. The JDK 1.2 contains the JDBC 2.0 Core API which is the latest version of JDBC. However, there are not any JDBC drivers bundled with the JDK releases other than the JDBC-ODBC bridge. So, developers need to get a

driver and install it before they can connect to a database. JDBC drivers are provided by vendors in order to access their specific product. Every JDBC driver falls into one of the types discussed below.

Types of JDBC drivers

JDBC drivers fit into one of four types:

1. The *JDBC-ODBC bridge* provides JDBC access via most ODBC drivers. Since some ODBC binary code and in many cases additional database client code must be loaded on each client machine that uses this driver, this kind of driver would be most appropriate for an intranet application, or for application server code written in Java in a 3-tier architecture.
2. A *native-API partly-Java* driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Like the type 1 bridge driver, this type of driver also requires that some binary code be loaded on each client machine. The Domino Driver for JDBC is a Type 2 driver.
3. A *net-protocol all-Java* driver translates JDBC calls into a DBMS-independent net protocol, which is then translated to a DBMS protocol by a server. This net server middleware is able to connect its all-Java clients to many different databases. The specific protocol used depends on the vendor. In general, this is the most flexible JDBC alternative. It is likely that all vendors of this solution will provide products suitable for Intranet use. In order for these products to also support Internet access, they must handle the additional requirements that the Web imposes, such as security, access through firewalls, and so forth. Several vendors are adding JDBC drivers to their existing database middleware products.
4. A *native-protocol all-Java* driver converts JDBC calls into the network protocol used by DBMSs. This allows a direct call from the client machine to the DBMS server and is a practical solution for Internet access. Since many of these protocols are proprietary, the database vendors themselves will be the primary source for this style of driver. Several database vendors have these in progress.

Domino Driver for JDBC

The Domino Driver for JDBC enables Java programmers to use any JDBC standard enabled application development tool, such as IBM VisualAge for Java, Borland JBuilders, or Symantec Visual Cafe, to access information stored in Domino databases. The Domino Driver for JDBC makes Domino databases *look* like another relational back-end source to the SQL tool or application interface by producing result sets that mirror the relational

model. An application can also perform an *SQL Join* of data from Domino with data from a relational database such as Oracle, Sybase, or DB/2.

The Domino Driver for JDBC can be downloaded from

```
http://www.lotus-developer.com
```

Note From a functionality and SQL syntax viewpoint, the Domino Driver for JDBC is the same as NotesSQL, which is discussed later in this chapter.

Servlets

A Java servlet is a protocol- and platform-independent server-side software component, written in Java. Servlets run on a Web server machine inside a Java-enabled server, that is, a server that can start the JVM to support the use of Java servlets. They dynamically extend the capabilities of the server because they provide services over the Web, using the request-response paradigm. The servlet technology, created by Sun, is an extension to the standard Java language. If you want to run or create your own servlets, you can download the Java Servlet Development Kit (JSDK) from the JavaSoft Web site

```
http://java.sun.com
```

From a high-level perspective, the servlet process flow is:

1. The client sends a request to the server.
2. The server sends the request information to the servlet.
3. The servlet builds a response and passes it to the server. The response is dynamically built, and the contents of the response usually depend on the client's request.
4. The server sends the response back to the client.

Servlets look like ordinary Java programs. The servlets import particular Java packages that belong to the Java servlet API. Because servlets are object byte codes that can be dynamically loaded off the Web, we could say that servlets are to the server what applets are to the client. However, servlets run inside servers, so they do not need a graphical user interface (GUI). In this sense servlets are also called faceless objects.

Servlets are controlled by the Domino Java Servlet Manager which is part of the HTTP server task. For more information on configuring servlet support for Domino, refer to the redbook, *Connecting to Domino to the Enterprise using Java*, Lotus part number CT6EMNA, IBM form number SG24-5425.

CORBA and Domino

Domino R5.0 unveils support for the Common Object Request Broker Architecture (CORBA), so you can build robust, distributed applications. The Object Request Broker (ORB) technology and Java allow you to create client applications that are dynamically loaded from the server with transparent access to the server-side *Domino Object Model*. While Notes client applications have been able to access Domino objects for quite some time, CORBA and Internet Inter-ORB Protocol (IIOP) support in Domino R5.0 expands this access to Web clients. Your primary access to this ORB is through Java applets or applications. For example, you can place a custom Java applet on a form and have that applet access objects in either the Notes client or a Web browser. For the Notes client, you're actually using the local Java interfaces. For the browser, you're using the CORBA-remote objects — that is, the applet uses IIOP to connect back to the ORB on the server.

CORBA allows you to create client-side objects that *talk* IIOP across the wire to the server-side ORB, which is hard-wired to Domino's back-end classes for better performance. The main purpose is to offload the server by projecting its services to the client. Browser applications can then execute locally with the context of the Domino server, so for instance, you can interact with your server-based mail locally, without involving the server in each user transaction.

With support for CORBA and IIOP, Domino now allows you to create client/server Web applications that take advantage of the Domino objects and application services. In addition, you can now access back-end relational databases for enhanced data integration using the new Domino Enterprise Connection Services. In previous releases, when you designed a Web application, the Domino Object Model (formerly known as the remote back-end classes, or the Notes Object Interfaces or NOI) allowed you to access data that was not on display in the browser. These back-end classes were LotusScript or Java objects. In R5.0, Lotus has made these objects available to the browser using a technology called Common Object Request Broker (CORBA).

This means that now your Web application is similar to your Notes application in terms of programmability. In a Notes application, you could always manipulate data that was on display and data in other databases. In a Web application, you had to wait for a user to open or save a Web page to access this same data. CORBA allows you to access the data without the user opening or saving the document; instead, you can use Java or JavaScript on a W3C event.

You can also embed a CORBA applet in a document or a form using the same procedure as for any other applet. You can use a browser to view embedded CORBA applets on a Domino server. It is no longer necessary to set alternate HTML. A CORBA property box setting tells Domino to provide the HTML source that the applet needs to make an IIOP connection back to the server.

ADO and OLE DB

ActiveX data Objects (ADO)

ADO was designed by Microsoft to access databases of all types. ADO is the data interface to OLE DB. Independent database service providers can write OLE DB providers that can access databases. ADO is a COM-based (Common Object Model) component and any application that can work with COM can use ADO.

ADO is one of the components of Microsoft Universal Data Access. The latest set of components can be downloaded from the Microsoft site at:

```
http://www.microsoft.com/data
```

At the time of the writing of this book, MDAC 2.0 (Microsoft Data Access Components) was available for download. These components must be installed on each computer that needs to use ADO.

In addition to the MDAC software, some versions of Windows 95 will also need DCOM95 1.2 or later installed. You can download DCOM95 1.2 from the Microsoft Web site

```
http://www.microsoft.com/com
```

You must first install DCOM95 if needed and the MDAC software.

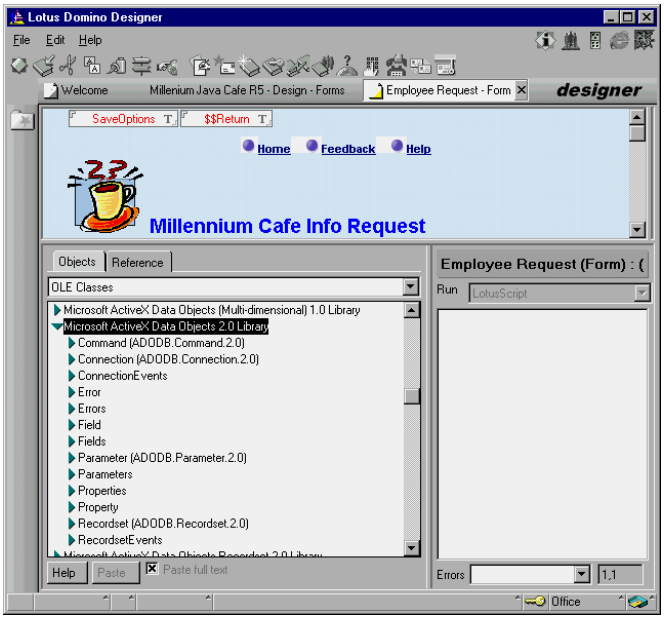
If you are finished installing this software, you can start to use these Classes from Notes.

How to access ADO Classes from Domino

In the design pane in the Domino Designer, where you can use LotusScript, you can access the OLE classes by clicking on the reference pane and selecting OLE Classes from the drop-down list. This is where all the classes that are registered on the system will be displayed. The ADO Objects will show as

```
Microsoft ActiveX Data Objects 2.0 Library
```

This can be used to show all the Objects with their Methods and Properties. There will not be any help if you press the help button on the bottom of the screen. Refer to the following figure for an example of the screen that was just explained.



The ADO Objects

There are 7 objects and 4 collections that you can use with ADO:

- **Objects**: Connection, Command, Parameter, Recordset, Field, Property, and Error.
- **Collections**: Fields, Properties, Parameters, and Errors.

Note Three of the objects are described briefly in this section. For further information, refer to the help documentation that was provided with the installation of MDAC 2.0.

Recordset Object

This object can be used without the other objects to connect to an external database and create a recordset that can be manipulated. To create any OLE object, use the CreateObject function in LotusScript after you creat a variant variable:

```
Dim rs As Variant
Set rs = CreateObject("ADODB.Recordset.2.0")
```

The above code will create a recordset object and assign it to the rs object variable. Next, open the recordset:

```
rs.Open “SELECT Client_Name FROM contract”, “DSN=DBDATA”
```

This will open the recordset and place the result into the recordset object. You can use any SQL that is allowed by the data source. In this case the data source is a dbase file and the only connection information that is needed is the Data Source Name (DSN). Other connection information you may need is the username and password:

```
rs.Open “SELECT client_Name FROM contract”, _
“DSN=DBDATA;UID=db2admin;PWD=password”
```

After this line, you are ready to access the data. Use the following methods to move within the recordset.

```
rs.Movefirst
rs.Movenext
rs.Movelast
rs.Moveprevious
```

To access a field value within the active record:

```
Msgbox rs.Fields.Item(0).value
```

This will display the content of the first field (Client_Name) in the select query in a message box. To display the next field, you can select the next item number. All fields are stored in an array.

```
Msgbox rs.Fields.Item(1).value
Msgbox rs.Fields.Item(2).value
```

Connection Object

If you want more control over the connection and its settings, you can use this object. In the example below, a connection object is used to change time out from a default of 60 seconds to 150. After the connection is opened, the recordset is opened with the active connection as one of its parameters:

```
Dim con as variant
Dim rs as variant
Set Con = CreateObject(“ADODB.Connection.2.0”)
Set rs = CreateObject(“ADODB.Recordset.2.0”)
Con.Open
“DSN=DBDATA;UID=db2admin;PWD=password”Con.CommandTimeout = 150
rs.Open “SELECT client_Name FROM contract“,Con
....
```

Command Object

With the Command Object, you have more control over the SQL or statement that you want to send to the Database. You can optionally send parameters using the parameter object. You can define a specific command

with the Command object and then run it against a data source. You can use a Command object to query a database and return the result in a RecordSet object, or to manipulate the structure of a database. Depending on the provider used and the back-end data source, all Command methods, collections and properties may not be supported and may generate an error.

```
Dim Com as variant
Dim con as variant
Dim rs as variant
Set Com = CreateObject("ADODB.Command.2.0")
Set Con = CreateObject("ADODB.Connection.2.0")
Set rs = CreateObject("ADODB.Recordset.2.0")
Com.CommandText = ""SELECT client_Name FROM contract"Con.Open
"DSN=DBDATA;UID=db2admin;PWD=password"Con.CommandTimeout = 150
rs.Open Com,Con
....
```

Example

The following example, in the exiting event of a field on a Notes form, will query an Oracle database with the value typed into the field (HR_ID) and then update the form in real time. It uses the 3 ADO objects described previously and the default OLE DB provider.

```
Sub Exiting(Source As Field)
' Define the variables
     Dim ws As New Notesuiworkspace
     Dim uidoc As notesuidocument
     Dim Cmd As Variant
     Dim Con As Variant
     Dim Rs As Variant

' Create the Objects
     Set uidoc=ws.currentdocument
     Set Cmd = CreateObject("ADODB.Command.2.0")
     Set Con = CreateObject("ADODB.Connection.2.0")
     Set Rs = CreateObject("ADODB.Recordset.2.0")
     uidoc.refresh

Con.CommandTimeout = 150 'Set connection time out to 150 secs.
Con.Open "DSN=ORACLE;UID=scott;PWD=tiger" ' Open connection
Cmd.CommandTimeout = 0 ' switch command timeout off
Set Cmd.ActiveConnection = Con
'Define the SQL statement
Cmd.CommandText =_
"SELECT FName,Lname,address1,Title,Company,city FROM " & _
     "attendees Where acakey = '" _
 + Trim(uidoc.fieldgettext("HR_ID")) + "'"' run the command and
return the result in a recordset
rs.Open Cmd
' Use rs.movefirst, rs.movenext, re.moveprevious, rs.movelast
```

```
‘ To move between records in a recordset
‘ Update the notes document
Call _
uidoc.Fieldsettext(“FirstName“,Trim(rs.Fields.Item(0).value))
Call _
uidoc.Fieldsettext(“LastName“,Trim(rs.Fields.Item(1).value))
Call _
uidoc.Fieldsettext(“address“,Trim(rs.Fields.Item(2).value))
Call uidoc.Fieldsettext(“Title“,Trim(rs.Fields.Item(3).value))
Call _
uidoc.Fieldsettext(“company“,Trim(rs.Fields.Item(4).value))
Call uidoc.Fieldsettext(“City“,Trim(rs.Fields.Item(5).value))
uidoc.gotofield(“HR_ID”)
Rs.Close ‘ close the recordset to free memory
Con.Close ‘ close the connection free memory
End Sub
```

NotesSQL

NotesSQL is the Lotus Notes ODBC driver for Windows, allowing you to connect to and query a Notes database (an NSF file) through the Open Database Connectivity (ODBC) interface. NotesSQL makes Domino look like another relational back-end source to the SQL tool or application by producing result sets that mirror the relational model. An application can also perform an SQL Join of data from Domino with data from a relational database such as Oracle, Sybase, or DB/2.

The current release is NotesSQL 2.05. This release should be used with Domino R5.0.

This section outlines NotesSQL. See the documentation for details.

ODBC Conformance

Programs using ODBC are database independent. The ODBC Driver Manager takes library calls from your program and passes them in appropriate format to the database. The ODBC library contains calls to connect to a database, execute SQL statements, and retrieve results.

NotesSQL supports the three ODBC conformance levels with exceptions as noted:

- Core Level, except for transaction control
- Extension Level I
- Extension Level II except for primary key and foreign key, table and column privilege control, stored procedure, and cursor control

NotesSQL supports the minimum SQL grammar conformance level and some additional grammar. You can check the grammar conformance level with the SQLGetInfo function.

Downloading and Installing NotesSQL

You can download NotesSQL from the Lotus Developer Web site

`http://www.lotus-developer.com`

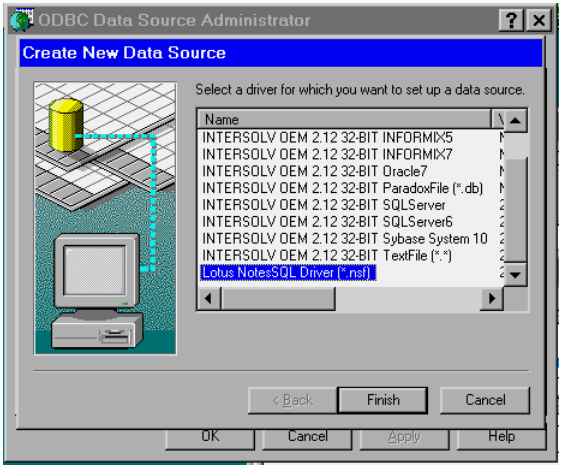
We recommended that you also download the samples.

Download the NotesSQL software to a temporary directory. Run the application and follow the installation instructions. When the installation completes, the ODBC Data Source Administrator opens, allowing you to enter a Domino data source or close it.

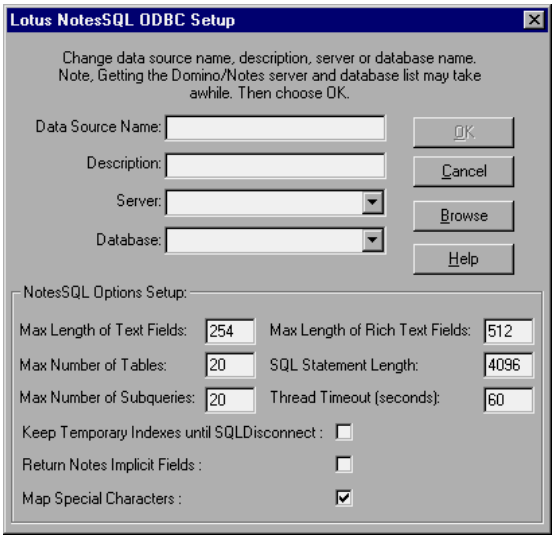
The installation program installs two versions of the documentation: HTML and a Domino database (NSF). You can use the Domino version only from a Notes client.

Setting up a NotesSQL Data source

Before you can access a Notes database with NotesSQL, you must configure a connection to it with the ODBC Data Source Administrator. Start the ODBC Data Source Administrator and add a User DSN or System DSN selecting Lotus NotesSQL as the ODBC driver.



In the Lotus NotesSQL ODBC Setup screen, enter a name and description for the data source. Identify the source by server (blank for local) and database. The database identification is its file name, which is relative to the Domino data directory for a server (for example, personnel\employee.nsf) and absolute for a local file (for example, c:\notes\data\personnel\employee.nsf).



Connecting to a Domino Data Source

NotesSQL prompts for a password when you first access a database from a Domino server if your Domino user ID is password protected. NotesSQL supports the following general keywords for the SQLDriverConnect call:

- **DSN**: Name of the data source.
- **Database**: Name of the Domino database, with a path if necessary.
- **Server**: Name of the Notes server where the database is located. If the database is local, do not specify.
- **DRIVER**: Name of the ODBC driver.

Mapping SQL Components to Domino

SQL components map to Domino as follows:

- **Table**: Maps to either a Domino form or view. However, a Domino database contains only one real table, referred to as the Universal Relation. This table has the same name as the database.
- **Column**: Maps to a field in a form or a column in a view. When creating a NotesSQL Table or View, you should avoid the use of column names that are ODBC or SQL reserved words or that contain characters other than letters, numbers, and underscores.
- **Index**: Maps to a view in which all sorted columns refer directly to fields in a single form, and that selects documents from only that form.

- **View**: Maps to a view that selects documents from one form in which all columns are calculable from the form. When you create a view using SQL, a view is created in Domino that selects from one form. Except for private views, all Domino views are reported as SQL views.

Example Using Personal Address Book

The Domino Personal Address Book (NAMES.NSF) is a good database to illustrate the use of forms and views in a database. The Personal Address Book database includes:

- A form called Person
- A view called People with a sort key on LastName

The following statement is the most efficient way to find people in the Personal Address Book sorted by LastName:

```
SELECT LastName
FROM People
ORDER BY LastName
```

This query is efficient because NotesSQL can use the index already associated with the People view that lists LastName in the right order. Now assume you want to list people sorted by their mailing addresses. You could use the following statement:

```
SELECT LastName, Mail_Address
FROM People
ORDER BY Mail_Address
```

Since the People view is not sorted on Mail_Address, NotesSQL uses the People index, generates a temporary table, and creates a temporary index on Mail_Address. This results in slower performance.

A more efficient way to achieve the same result is to issue the following statement:

```
SELECT LastName, Mail_Address
FROM Person
ORDER BY Mail_Address
```

Person is a Domino form. If there is no index on Mail_Address, NotesSQL generates a temporary index on Mail_Address but does not need to generate a temporary table. This statement is faster than the previous statement, which used ORDER BY on a view-based table. This statement can be executed even faster if the user creates an index in Domino or uses the CREATE INDEX statement in NotesSQL.

Summary of Supported SQL Grammar

The supported SQL statements are:

- ALTER TABLE statement
- CREATE INDEX statement
- CREATE TABLE statement
- CREATE VIEW statement
- DELETE (positioned) statement
- DELETE (searched) statement
- DROP INDEX statement
- DROP TABLE statement
- DROP VIEW statement
- INSERT statement
- SELECT statement
- FOR UPDATE clause
- FROM clause
- GROUP BY clause
- HAVING clause
- ORDER BY clause
- UNION clause
- UPDATE (positioned) statement
- UPDATE (searched) statement
- WHERE clause

The supported operators are:

- Numeric operators: addition (+), subtraction (-), multiplication (*), division (/)
- Predicate operators: less than (<), greater than (>), less than or equal (<=), greater than or equal (>=), equal (=), not equal (<>), BETWEEN, IN, LIKE, NOT, ANY, SOME, ALL, EXISTS
- Column functions: AVG, COUNT, MAX, MIN, SUM

Grammar exceptions are:

- NULL: on a query, NULL means @IsAvailable is false. NULL on an update removes fields.
- ORDER BY clause: NotesSQL supports ordering by expressions that aren't in the project list. This is not standard SQL but many applications use it.

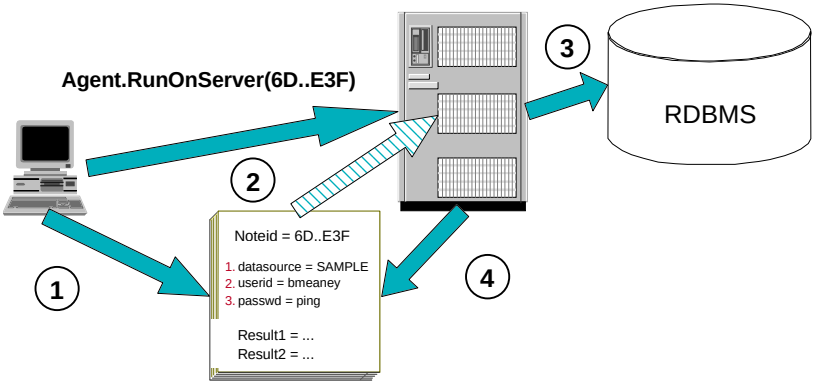
- WHERE clause and TIMESTAMP fields: Timestamp fields need to be fully padded when used in WHERE clauses if the time component is not used. That is, if time is not used, the following format is required: "1995-12-31 00:00:00"
- ALTER TABLE: The following keywords are not supported: NULL, NOT NULL.
- CREATE INDEX: The UNIQUE keyword is not supported.
- CREATE TABLE: The following keywords are not supported: NULL, NOT NULL, UNIQUE, PRIMARY KEY, REFERENCES. No constraint definition.
- PARAMETER RESTRICTIONS: Parameters are supported in INSERT, DELETE, and SELECT SQL statements. They cannot be used with CREATE TABLE, CREATE VIEW, and other statements that manipulate the structure of tables and views (DDL). Arrays of parameters are not supported.
- RESTRICT and CASCADE: Dependencies are only recognized if they were originally using the driver. DROP TABLE CASCADE only removes dependent views created using the driver. DROP RESTRICT only prevents the removal of a table if a dependent view was created using the driver.
- GRANT and REVOKE: Not supported. All access control is handled implicitly by Domino.

Using the RunOnServer to Connect to Enterprise Data

In Domino R4.6.1 a method called RunOnServer was added to the Agent class in the Domino Object Model and in Domino R5.02 it will be enhanced to accept a noteid as a parameter. With this new method it is now possible for users to access enterprise data using the LSX or Java classes for the Lotus Connectors (or one of the system-specific LSX') without having any communication software installed on their client.

Note At the time of writing Domino R5.02 was not released. The behavior of the RunOnServer method described here is based description of the planned implementation. The behavior and the release it is being implemented in may change.

We will here show the principle in using RunOnServer to access enterprise data. See the following figure



The user want to read data from a relational database residing on another system using the LC LSX. The user has no software for communicating with that relational database installed on the client. However, the Domino server has the software required to communicate with the relational database.

1. The user enters logon information and selection criteria for the desired data and the application stores it in a document in a database that resides in the Domino server
2. The agent that uses the LC LSX to connect to the relational database is started using the RunOnServer method with the noteid of the document that stores logon information and selection criteria. The server agent reads the information from the specified document.
3. The agent connect to the relational database and gets the requested data.
4. The result data is written to the document and the agents exits.

   The client application continue processing after the agent running on the server is completed. It reads the returned data from the document and make the data available to the user.

RunOnServer allows you to utilize the flexibility of programmatic access to enterprise data without the requirement for communication software on each client that accesses the enterprise data.

Note You can also create similar solutions with earlier versions of Domino using runOnServer without the noteid argument where you always use the same document to store the data being passed to and from the server agent. However, you must make sure that several users cannot run the same agent concurrently, for example by assigning different agents and documents to different users.

Summary

In this chapter we discussed additional ways to access external data from Domino using:

- Servlets
- CORBA
- ADO and OLE DB

We have also described ways to access data that resides in Domino from other systems, specifically:

- Domino Driver for JDBC
- NotesSQL

Finally, we have described a technique for accessing external data by programming without the requirement for communication software on the client.

- RunOnServer

Chapter 7
Which Tool to Use When

You can access enterprise data from Domino using a variety of declarative and programmatic options: DECS, LEI, or any of the programming tools discussed in this redbook. This chapter helps you decide which tool is most appropriate. Included in this chapter are charts that help you choose the one (and sometimes more) tools that are appropriate given your requirements. Requirement scenarios are included to provide examples on how to use the charts.

There are many choices to get to the same data sources. The first question to consider when deciding which tool to use is to determine which type of data source you are integrating with. There are three basic types of data sources:

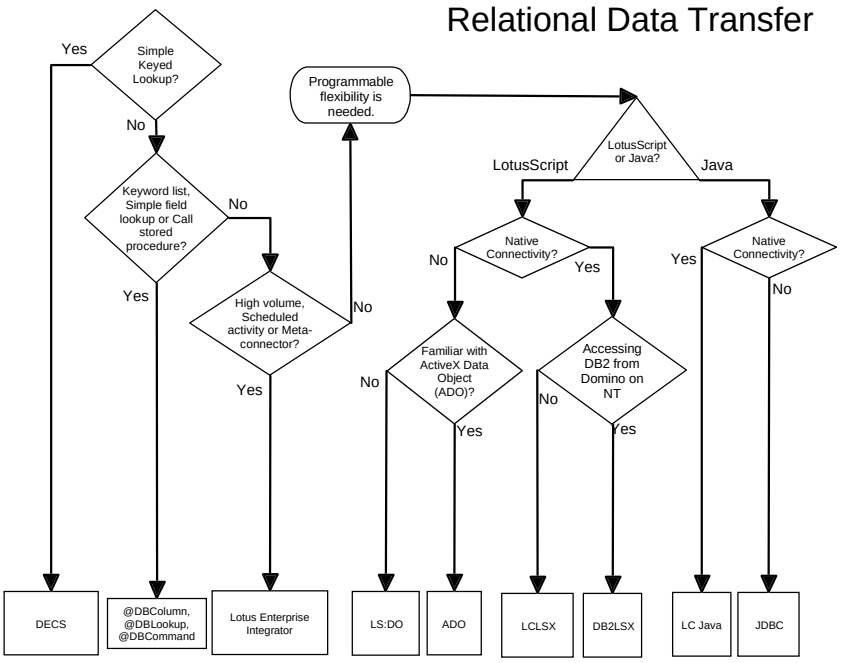
- Relational Database Management Systems
- Transaction Processing Systems
- ERP Systems

This section details the questions each of these decision charts ask, and provides guidelines for answering them. Some products do not require complex decision charts and these are discussed following the four main sections. Also, a section with sample requirements that uses the main decision charts to select a tool is included. Finally, general guidelines are given for using each of these products.

Note These charts do not consider your business restrictions (like special security requirements) or resource restrictions (like skills of your developers), which may play a part in your decision. Consider these charts as a guideline only.

Relational Data Transfer

The following decision chart should help you determine which tool to use when connecting to your relational data source.



Each decision "diamond" and "triangle" is discussed in a general logical order:

1. Do you need user-initiated access to individual records on the host — and can they be identified through a simple keyed lookup?

   DECS supports those requirements.

2. Do you need to get data from several records on the host to create keyword lists? Do you want to run stored procedures or do you want to look up a single field value?

   Any of these requirements can be solved using the @DBColumn, @DBCommand and @DBLookup formula language functions for retrieving data from and sending simple commands to ODBC compliant databases.

3. Is this a high volume data transfer? Do you require scheduled data transfer? Do you need to synchronize data between Domino and the relational database? Do you require use of MetaConnectors?

   For any of these requirements LEI is the best tool. DECS will also support use of MetaConnectors for live data access in the future. If you want to use MetaConnectors in connection with scheduled transfer or synchronization of data, LEI will still be the tool of choice.

4. Do you require more flexibility than is offered by the above mentioned tools?

   Then you have to program your solution using either Java or LotusScript.

5. If your language preference is Java, do you require the performance offered by native connectivity?

   If the answer to this questions is yes, you should use the Lotus Connector Java classes.

   If native connectivity is not required, you can connect to the relational data using standard JDBC.

6. If your language preference is LotusScript, do you require the performance offered by native connectivity?

   If the answer is yes, you have one more decision point.

   Does your data reside in DB2 and do you access it from Domino running on Windows NT?

   If the answer is yes, and you require access to Character Large Objects (CLOBs) or Binary Large Objects (BLOBs), you should use the DB2LSX.

   If you already have experience in programming the LotusScript Data Object (LSDO), you should also consider the DB2 LSX, as it is very similar to LSDO.

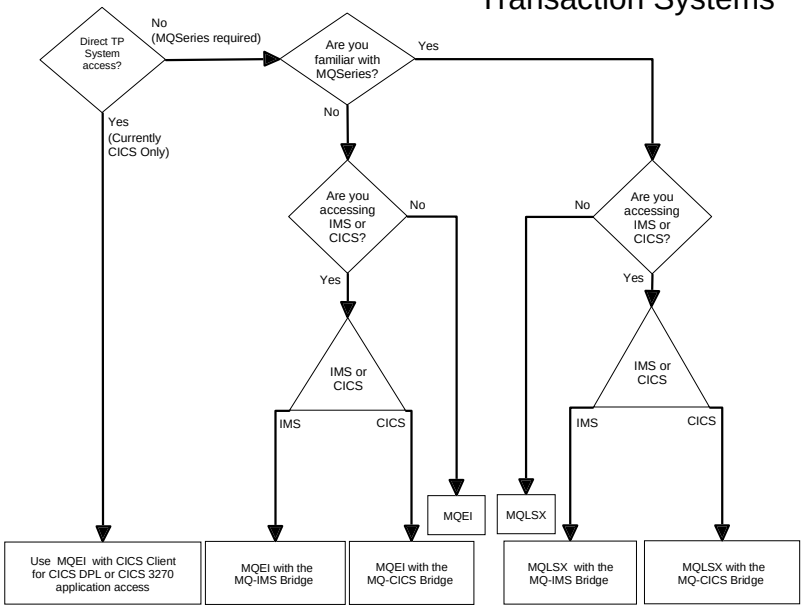
   For all other types of native LotusScript access to relational data you should use the Lotus Connector LSX.

7. If native access is not required or perhaps is not possible (for example if you want to access MS SQL Server), the question is are you familiar with ADO?

   If you are familiar with ADO/COM and the database you want to access has an OLEDB driver, you should use ADO.

Transaction Services

The following decision chart should help you determine which tool to use when connecting to your enterprise transaction system.



Each decision "diamond" is discussed in a general logical order:

1. Do you want to go directly to CICS or do you want to use MQSeries to access your TP data?

   If you want to connect directly to your CICS system, you should use MQEI with CICS client for CICS DPL or CICS 3270 application access.

2. If you want to connect to your transaction data via MQSeries, the question is whether you have experience with programming for MQSeries?

   If you want to connect to your TP system without programming for MQSeries, you should use MQEI.

   - If your TP system is IMS, you should use MQEI with the MQ-IMS bridge.
   - If your TP system is CICS, you should use MQEI with the MQ-CICS bridge.
   - For all other MQSeries-enabled TP systems use MQEI

3. Do you require more flexibility than MQEI supplies?

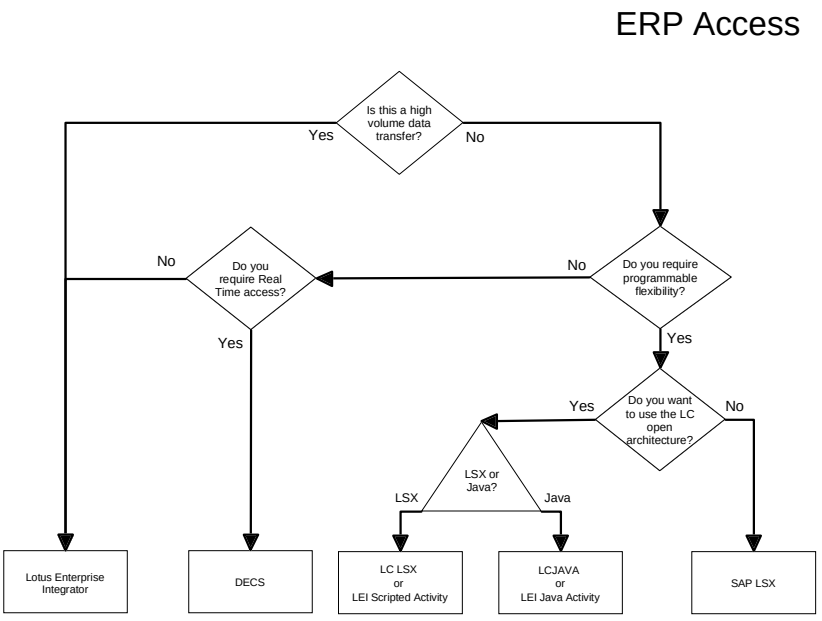
   If so, you should use MQ LSX.

   - If your TP system is IMS, you should use MQ LSX with the MQ-IMS bridge.

- If your TP system is CICS, you should use MQ LSX with the MQ-CICS bridge.
- For all other MQSeries-enabled TP systems use MQ LSX.

Enterprise Relationship Planning (ERP) Systems

The following decision chart should help you determine which tool to use when connecting to your ERP system.



This figure does not indicate all of the ERP systems that you can access. Only the previously released SAP LSX is unique to the decision chart. All other ERP Systems' connectors can be accessed by the resultant tool in this chart. The supported ERP systems are:

- J.D. Edwards
- Oracle Financials
- PeopleSoft
- SAP R/3
- Lawson

Each decision "diamond" and "triangle" is discussed in a general logical order:

1. Is this a high volume data transfer?

If your application requires you to move high amounts of data, LEI is the tool best suited for this.

2. Do you require programmable flexibility?

   If you require programmable flexibility, you have the choice of using the LC open architecture LC Java and LC LSX classes or the existing SAP LSX.

3. Do you want to use the LC open architecture?

   If you are working with an existing Notes/SAP application that was written with the older database specific drivers, you can continue using SAP LSX. However, the open architecture LC LSX and LC Java classes should be used if you're developing a new application. An additional option is the LEI Scripted Activity or Java Activity.

4. Do you prefer LotusScript or Java?

   Before making this choice, consider how you want your application to trigger its data transfer. Only LotusScript can be triggered by a form event. If your data transfer is not form-event based, you can still use LotusScript (as an agent) or LC Java, LEI Scripted Activities or LEI Java Activities (as indicated above).

5. Do you require real-time access?

   Continuing from item three above, if your application requires real-time access but you do not want to get involved with programming, you should use the LEI Real Time Activity or DECS, when all the ERP connectors become available for this product. If you require real-time access but still need the flexibility provided by scripting, you again have various options. You could use the LEI scripted activity or Java activity if you need to access high amounts of data in real-time. Another option is the LC LSX or LC Java classes. Since LotusScript can be event driven, you can program buttons on a form to do a lookup or refresh a form. LEI activities can only be triggered on a scheduled basis or using the RunASAP agent in the LEI Administrator Database.

Other Services

The following four connectors exist for unique and implied connections from Domino:

- File System

  This connector allows you to connect to a networked file system from Domino, or to your Domino server's computer file system. The file system can be accessed from LEI, DECS, LCLSX and LC Java. Follow the decision chart for relational data transfer to determine the best tool.

- Text

  This connector allows you to connect to a text file on your Domino Server's computer file system. File layouts for the source and destination data are defined in Zmerge Information Description (ZID) files. Follow the decision chart for relational data transfer to determine the best tool.

- Notes

  From the Enterprise Integration perspective as well as LEI's perspective, Notes is a connector. This connection is implied when using DECS, LCLSX and LC Java by using standard LotusScript and Java classes for Notes. The decision to connect to Notes is assumed and is not mentioned in the charts.

  Note LEI can transfer data between non-Notes systems. This explains the need for the Notes connector in LEI.

- EDA/SQL

  This connector transfers data to and from sources supported by Information Builder's EDA Server. Follow the decision chart for relational data transfer to determine the best tool.

Scenarios

The following scenarios use the detailed decision charts to select a tool suitable to perform the requested task.

Example 1: Access to SAP R/3

Customer requirement: Make SAP R/3 Server product price data available to Notes and Web clients in real-time.

Select the ERP Access decision chart and ask yourself the following questions:

1. Is this a high volume data transfer?

   The answer to this question is no because the data is accessed in real-time and not moved from its original source.

2. Do you require programming flexibility?

   The requirement is to have data from SAP R/3 displayed by Notes and Web clients. No modifications are required to the data, so the answer is no.

3. Do you require real-time access?

   Yes. That leaves us with the choices of DECS or the LEI Real Time Activity. Since LEI would be an additional expense, DECS should be used.

Use Domino Enterprise Connections Services (DECS) to create a connection from a Domino application to the SAP R/3 Server product price table. Map fields in the Domino form to SAP table fields (that populates the Domino application) within a DECS Real Time Activity. Save and enable the Activity on the Domino Server, providing Domino end users with SAP R/3 data when Domino forms are accessed.

Example 2: Access to PeopleSoft

Customer requirement: Transfer Domino employee change data to PeopleSoft twice per day.

Select the ERP Access decision chart and ask yourself the following question:

1. Is this a high volume data transfer?

   With data for all employees being transferred twice a day, let's assume this will be at least a moderate amount of data. Answer yes to this question. LEI should be used since its strength lies in scheduled and high volume data transfer.

Create connections to the PeopleSoft Panel definition application and the Domino application using LEI Connectors for PeopleSoft and Notes. Create a Direct Transfer Activity and map fields between the Domino form and PeopleSoft Panel data field definitions. Schedule the Direct Transfer Activity to execute twice per day (select the specific times or interval periods in the Schedule section of the Activity document) and enable the Activity on the LEI Server.

Example 3: Access to J.D. Edwards

Customer requirement: Create purchase orders in a Domino application, manage approvals through Domino workflow processes, and send the approved purchase order data to J.D. Edwards. Return the J.D. Edwards generated purchase order number, obtained through the update, to the Domino application.

Select the ERP Access decision chart and ask yourself the following questions:

1. Is this a high volume data transfer?

   The process is handling individual purchase orders, so the answer is no.

2. Do you require programming flexibility?

   Since the requirement is for individual purchase order numbers to be returned to the Domino application once approval is given for an order, you could enable this through an approval button that runs LotusScript to transfer the data to J.D. Edwards. Choose Yes for this question.

3. Do you want to use the LC open architecture?

Yes. In this case we have to since the J.D. Edwards connection is only available with LC LSX or LC Java. However, we will use LotusScript to program the Domino Workflow and will thus use LC LSX.

Create the purchase order application in a Domino form. Optionally, copy inventory data from J.D. Edwards to the Domino application using an LEI Direct Transfer Activity (if the inventory data is fairly static; otherwise, a real-time lookup approach is recommended). Create Domino processes to appropriately route the purchase order for approvals. Use the LC LSX to transfer the approved form data from Domino to the J.D. Edwards purchase order MBF table. This automatically causes an update and generates a purchase order number from the J.D. Edwards purchase order MBF processing. Return the purchase order number to the Domino application.

Example 4: Access to Oracle Financial Applications

Customer requirement: Publish Oracle Applications — Financials cost center report data to a Java-based Domino intranet Web site on an hourly basis.

Select the ERP Access decision chart and ask yourself the following questions:

1. Is this a high volume data transfer?

   This depends on how many updates are made on an hourly basis. Let's make the assumption that frequent updates mean small amounts of data are transferred at a time.

2. Do you require programming flexibility?

   The cost center reports are created in Oracle before being transferred. We do not need programming flexibility.

3. Do you require real-time access?

   No. Data is updated hourly, not in real-time. This leaves LEI as the tool to use to satisfy this customer requirement.

Load the Lotus Connector Java Class Library and use it to create a Java Server program that connects to a specific Oracle Applications Financials cost center consolidations table via the Domino Connector for Oracle Applications. Use Lotus LEI to schedule a Java Activity, identify the Java Server program and hourly time interval to execute the query, and transfer to the Java Server program.

Example 5: Access CICS/ESA Transactions and DB2 on MVS

Customer Requirement: Capture Contact Management changes in Domino and transfer the data to a CICS transaction on MVS. Additional header information for each transaction must be included to handle the business process this transaction starts. Immediately return a receipt message to

Domino that must be processed as soon as possible. The contact data changes are stored on DB2. Contact data changes from other customer systems (Call Center and intranet) are also stored in DB2 (their transfer to DB2 is outside of this requirement) and their changes must set a trigger column in DB2, indicating a change has occurred that Domino does not know about. These changes must be sent back to Domino as they occur, and the trigger column must be reset once Domino has the data. This customer has MQSeries installed. The Call Center and intranet solutions are not Domino-based.

This requirement requires us to use the Transaction Systems decision chart and the Relational Data Transfer decision chart.

Select the Transaction Systems decision chart and ask yourself these questions:

1. Do you want to access CICS directly or via MQSeries?

   We want to take advantage of the existing MQSeries infrastructure at the customer.

2. Do you require programmable flexibility?

   The requirement for additional header data and receipt message processing can be handled either way. The answer to this question lies with your business restrictions, infrastructure restrictions and resource skills. To allow us to continue we will assume the customer wishes to limit the hardware and software budget and attempt to use whatever is available to them. The customer is running S/390 on MVS, MQSeries (not part of their current Contact Management System), CICS TS, DB2 with UNIX Services and TCP/IP for MVS. They are also running their current Domino base on Windows NT Servers. It is obvious here that a complete investigation is required before you make a decision.

   Selecting the non-programmatic approach leaves you with LEI or MQEI. LEI is part of the solution and is a viable approach, however the CICS connector is currently unavailable. MQEI must be purchased, but the customer wishes to limit the budget.

   This leaves us with the programming solution. We are connecting to CICS so we must use the MQLSX with the MQ-CICS/ESA DPL Bridge. Make sure you investigate the need for DPL or 3270 Bridge. The new MQ-CICS bridge includes both. The DPL bridge can be used if your CICS transactions are separated ECI and EPI programs. The DPL bridge interacts with the ECI program (CICS Communication Interface — better known as the COMMAREA). The 3270 Bridge interacts with CICS programs that combine ECI and EPI, forcing you to write BMS Mapping exits for the 3270 Bridge.

We must create new queues and channels to connect MQSeries on the Domino machine to MQSeries on MVS. Since MVS has TCP/IP, the channel

configuration will be simple. We are also using the DPL Bridge from MQSeries to CICS; therefore we only need to know the transaction name and communication area layout to create our message. To simplify the programming of this solution, a submit button is created on the Contact Management form that creates a transaction message document on the Domino server in an Agent database. Similar components of the sample code provided by MQLSX are used. This includes the Agent database where transaction documents are stored and the MQSeries Application Transaction Mapping (MATM) database where the transaction formats are stored. This limits the solution to pseudo real-tim but it does improve the performance of the server-based contact management application. The MQLSX Agent database contains your Domino agent that runs against "new and modified" documents. Server NOTES.INI settings can reduce the time that the agent manager waits before it checks for the agent trigger condition (refer to the "Lotus EI Product Updates" chapter for these settings). The agent creates the MQSeries message and sends it to the remote queue definition on the Domino Server. MQSeries transmits that message to MVS, at which point the Bridge's trigger monitor retrieves the message, reads the CICS transaction name stored on the message, puts the message into that transaction's comm area and starts the CICS transaction.

When that transaction completes, it can start another CICS program or linked program to send an MQSeries reply message (with relevant status data) back to Domino. When that message is received on the Domino machine, the MQSeries Trigger Monitor for Domino starts an agent on the Agent database designed to accept reply messages. This agent updates the source document with the appropriate information to indicate that the original submission was received.

Now for the Relational Data Transfer part of the requirements. Select that chart and ask yourself these questions:

3. Do you need user-initiated access to individual records on the host — and can they be identified through a single key?

   No, the change of data should be triggered by data from other systems arriving in DB2, not through some action by a Domino user.

4. Do you need to get data from several records on the host to create keyword lists? Do you want to run stored procedures or do you want to lookup a single field value?

   No, we want to transfer full records to Domino unattended.

5. Is this a high volume data transfer?

   Since the customer wishes to receive DB2 as the changes occur, it is likely not high volume. One possible issue with this is that we are dealing with Call Center data. Depending on the size of the contact

(client) base of the customer, this may very well be high volume. The answer is no for now; however, you must keep a yes answer as a viable solution.

6. Do you require scheduled data transfer?

   The data must be transferred unattended. No Domino user should be involved in transferring the data, so we will answer Yes. The actual schedule is determined by when updates from the Call center and intranet arrive in DB2. This leads us to choose LEI, where we can use the polling activity to detect updates in DB2 and initiate a scheduled transfer.

The second part of the solution requires us to keep the Domino Contact Management application up to date with contact information not created or modified in Domino. Based on the triggering requirements, you use an LEI Polling activity that watches the trigger column on the contact management DB2 table on MVS. The polling activity remains active in LEI and you set the polling frequency to 60 seconds (you can play with this setting once you become familiar with the frequency of change in the MVS contact management table). The longer this frequency setting, the more potential for higher volume. LEI's Direct Transfer Activity is an excellent performer for potential high volumes. The Polling activity runs the Direct Transfer activity once a successful poll occurs. The direct transfer activity only selects DB2 rows where the trigger column is set and it transfers that DB2 data over to Notes. The Direct Transfer activity must set the "Try Update before Insert" option so we do not get duplicate documents in Notes. Once the Direct Transfer activity is complete, it runs a Command Activity that sets all the trigger columns off.

Note If the frequency of change in the DB2 table is high, you may find the Command Activity updating rows that have changed since the direct transfer finished (a matter of seconds). This gap can only be closed if the Direct Transfer Activity could execute post activity statements. If you require a short polling frequency, you may want to consider a scripted direct transfer that uses the DB2 Lotus Connector to move the data and reset the trigger at the same time. This action keeps the DB2 table (page or row) locked until the transfer is done. If your polling frequency is very long, consider a scheduled Direct transfer activity and dependent Command Activity to run during non business hours or times of minimal activity on DB2.

Example 6: Web User Access to DB2

Customer Requirement: Provide real-time Web access from a Domino database to DB2.

Select the Relational Data Transfer decision chart and ask yourself the following question:

1. Do you need user-initiated access to individual records on the host — and can they be identified through a single key?

   The transfer is user-initiated and a record can be identified through a single key so the answer is yes, which leads us to select DECS as the tool to use for this scenario.

General Guidelines

DECS, LEI, and the LC programming tools are all powerful products that can be applied in a number of different solutions. In fact, some solutions could incorporate all technologies, while others could be solved by any one. Following are some high-level guidelines for determining when one is better suited than the other:

- DECS: This technology is the most productive way to incorporate enterprise data into your Domino Applications. Without programming, it allows you to map fields of data from a Domino form to a field in an enterprise system table or view. Among other things, this can be used to build Domino applications that front end database or ERP systems, accept query parameters, display results, and so forth. With DECS, data can be left in its original data store and still be part of the Domino application. If a customer's only requirement is to access data real-time, DECS is their best choice.
- LEI: This technology is a data distribution server. It is the best way to transfer and synchronize high volumes of data between enterprise systems and Domino. Actually, it can be used to move data between supported Connector sources, rather than strictly between Domino and another source. Also, since its activities can be run on a schedule, it may be useful for moving large amounts of data during non-peak usage times.
- LCLSX: The Lotus Connector classes for LotusScript can be used for virtually every Domino-controlled integration solution. It strengths lie in it programmable flexibility and forms-based LotusScript event triggering in Notes.
- LC Java classes: The Lotus Connectors for Java can be used for most Domino integration solutions. Their appeal is preferential, but they also have advantages for integration control outside of Domino. This allows you to create Web servlets that can run and access enterprise data and store it in Domino. LC Java can run as an external servlet, a Domino HTTP servlet, or a Domino Agent.

Appendix A
Lotus Connectors

This appendix describes the supported connectors.

DB2

This connector transfers data to and from DB2 Universal Database sources using DB2 Connect Personal Edition Version 5, DB2 Connect Enterprise Edition Version 5, or CAE (Client Application Enabler) version 2.1.2 or later. You can also connect to DB2 on an AS/400 or mainframe using a DDCS gateway.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- OS/390
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types. See "Domino Connections Setup" (lccon.nsf) for requirement details.

Terminology

The following table lists corresponding Enterprise Integration and DB2 terms.

Enterprise Integration	*DB2*
Server	NA
Database	Database
Metadata	Table
Index	Index
Field	Column
Record	Row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Database, UserID, Password
Standard Functionality	Full
Writeback Support	Full except when the result set is ordered. In this case, writeback functionality is not available and is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
Writeback Index	Index strongly advised, but not required. Use the command CREATE INDEX <indexname> ON <tablename> (<column1>, <column2>, ..., <columnN>). (Simulated by keyed operations when cursors are not supported.)
Statement Syntax	Full DB2 SQL Syntax, including UDFs (User Defined Functions)
Condition Syntax	Valid when placed in the following DB2 statement: SELECT ... WHERE <condition> ORDER BY ...
Connection Properties	CommitFrequency, RollbackOnError, CreateMaxLogged, NoJournal
Array Transfer	Insert
Actions Supported	Clear, Reset, Truncate, Commit, Rollback
Catalog Types	Database (no connection required), Metadata, Field
Create Types	Metadata, Index
Drop Types	Metadata, Index

ODBC

This connector transfers data to and from ODBC-compliant sources, including MS-SQLServer. ODBC drivers must be appropriate to the operating system and 32-bit for Windows NT and OS/2. They must be thread safe. The ODBC Administrator must be present and must define the data sources under "System DSN."

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- OS/390
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types. See "Domino Connections Setup" (lccon.nsf) for requirement details.

Terminology

The following table lists corresponding Enterprise Integration and ODBC terms.

Enterprise Integration	*ODBC*
Server	NA
Database	Database
Metadata	Table
Index	Index
Field	Column
Record	Row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Database, UserID, Password
Standard Functionality	Full
Writeback Support	Full except when the result set is ordered. In this case, writeback functionality is not available and is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
Writeback Index	Index strongly advised, but not required. Use the command CREATE INDEX <indexname> ON <tablename> (<column1>, <column2>, ..., <columnN>). (Simulated by keyed operations when cursors are not supported.)
Statement Syntax	Full SQL Syntax, including UDFs (User Defined Functions)
Condition Syntax	Valid when placed in the following ODBC statement: SELECT ... WHERE <condition> ORDER BY ...
Connection Properties	CommitFrequency, RollbackOnError, CreateMaxLogged, NoJournal
Array Transfer	Insert
Actions Supported	Clear, Reset, Truncate, Commit, Rollback
Catalog Types	Database (no connection required), Metadata, Field
Create Types	Metadata, Index
Drop Types	Metadata, Index

Oracle

This connector transfers data to and from Oracle sources. SQL*Net must be on the Domino or LEI machine and must be the same version as on the Oracle data server: version 1 or 2 for NT; version 2 for OS/2. Native Oracle connectivity support requires Oracle version 7.2 or later. OS/2 works only with Oracle 7.3.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- OS/390
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types. See "Domino Connections Setup" (lccon.nsf) for requirement details.

Terminology

The following table lists corresponding Enterprise Integration and Oracle terms.

Enterprise Integration	*Oracle*
Server	Host String
Database	NA
Metadata	Table
Index	Index
Field	Column
Record	Row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, UserID, Password
Standard Functionality	Full
Writeback Support	Full
Writeback Index	Index strongly advised, but not required. Use the command CREATE INDEX <indexname> ON <tablename> (<column1>, <column2>, ..., <columnN>).
Statement Syntax	Full PL/SQL syntax. To execute stored procedures, use the following syntax: BEGIN <stored procedure> (<parm1>, <parm2>, ..., <parmN>) ; END ; ;. Note the extra semicolon required at the end of the statement.
Condition Syntax	Valid when placed in the following Oracle statement: SELECT ... WHERE <condition> ORDER BY ...
Connection Properties	CommitFrequency, RollbackOnError, CreateLongColumn, CreateLongByUser
Array Transfer	Fetch, Insert
Actions Supported	Clear, Reset, Truncate, Commit, Rollback
Catalog Types	Server (no connection required), Metadata, Index, Field
Create Types	Metadata, Index
Drop Types	Metadata, Index

Sybase

This connector transfers data to and from Sybase sources. A network connection must exist between Domino and the Sybase SQL Server. NT and OS/2 require System 10 Netlib.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later

- OS/390
- AS/400

Description

See “LEI Documentation” (leidoc.nsf) for detailed descriptions of the connection properties and data types. See “Domino Connections Setup” (lccon.nsf) for requirement details.

Terminology

The following table lists corresponding Enterprise Integration and Sybase terms.

Enterprise Integration	*Sybase*
Server	Server
Database	Database
Metadata	Table
Index	Index
Field	Column
Record	Row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, Database, UserID, Password
Standard Functionality	Full
Writeback Support	Full
Writeback Index	Index required. Use the command CREATE UNIQUE INDEX <indexname> ON <tablename> (<column1>, <column2>, ..., <columnN>)
Statement Syntax	Full Transact-SQL syntax, including stored procedures
Condition Syntax	Valid when placed in the following Sybase statement: SELECT ... WHERE <condition> ORDER BY ..
Connection Properties	Bulk, BulkEnable, CreateShortBound, CreateShortUnbound, DisableCursor
Array Transfer	None (Bulk Insert)
Actions Supported	Clear, Reset, Truncate
Catalog Types	Database (no connection required), Metadata, Index, Field
Create Types	Metadata, Index
Drop Types	Metadata, Index

J.D. Edwards OneWorld

This connector integrates Domino with the J.D. Edwards OneWorld ERP system. The connector performs data transfers through the application layer to preserve all business logic validations. The Domino or LEI machine must contain J.D. Edwards OneWorld Client version B732 or later.

Product Status

This connector is sold as a separate product by Lotus.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and J.D. Edwards terms.

Enterprise Integration	*J.D. Edwards*
Server	Environment
Database	NA
Metadata	Module, Table, or View
Index	NA
Field	Field
Record	Record

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, UserID, Password
Standard Functionality	Archive (to JD Edwards), Direct Transfer (to JD Edwards), Replication, RealTime
Writeback Support	Supported on LCXSelect, LCXUpdate, LCXRemove
Writeback Index	Not supported; writeback is always on last fetched record
Statement Syntax	NA
Condition Syntax	NA
Connection Properties	ClientName, Autocommit (default is TRUE), ContinueOnError (Default is FALSE), CommitOnMetadataChange (default is FALSE), OrderNumberme, ApplicationPassword, UseTransactions, TranTimeout, CommitFrequency
Array Transfer	None
Actions Supported	Clear, Reset, Commit, Rollback
Catalog Types	Metadata, Fields, Procedure
Create Types	NA
Drop Types	NA

Oracle Financial Applications

This connector integrates Oracle Applications Business Modules data seamlessly into Domino environments.

Product Status

This connector is sold as a separate product by Lotus.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later

- OS/390
- AS/400

PeopleSoft

This connector enables Domino applications to exchange data with PeopleSoft Application data. You need the PeopleSoft Message Agent and ODBC drivers appropriate for the Domino or LEI machine. The drivers must be 32-bit for Windows NT and OS/2. The ODBC Administrator must be present and data sources for PeopleSoft must be defined.

Product Status

This connector is sold as a separate product by Lotus.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later

Description

See "LEI D documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and PeopleSoft terms.

Enterprise Integration	*PeopleSoft*
Server	Activity
Database	NA
Metadata	Message Definition
Index	NA
Field	Field
Record	Row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, UserID, Password , ODBCDSN, ODBCUser, ODBCPassword (ODBCDSN, ODBCUser, and ODBCPassword are specific to the Connector used to connect to the PeopleSoft database using ODBC)
Standard Functionality	Select, Insert, Update, Message Definition (see below)
Writeback Support	NA
Writeback Index	NA
Statement Syntax	NA
Condition Syntax	NA
Connection Properties	TopicName, ODBCUser, ODBCPassword, ODBCDSN, SupportMultiValue
Array Transfer	None
Actions Supported	Clear, Reset
Catalog Types	Server, Metadata, Field
Create Types	NA
Drop Types	NA

The PeopleSoft connector interacts with the PeopleSoft Application Interface through the Message Agent. The Message Agent allows users to set and get the fields on PeopleSoft Panels and processes the panel action. This functionality is provided by the connector through Select and Insert functions.

Select sets the fields on the panel and processes the panel action. A subsequent fetch returns the fields that are retrieved on the PeopleSoft Panels. Insert sets the fields on the panel and processes the panel action, and does not care about the data that will be retrieved after action is processed.

Select, Insert, and Update cover all the functionality that can be provided by the Message Agent. All other connector functions, such as Remove and Execute, are not provided by the PeopleSoft connector.

A message definition must be specified before performing any operation on the PeopleSoft panel. The Message Agent uses the message definition as a dictionary for the information on the panel and field accessibility.

SAP R/3

This connector integrates Domino and SAP applications, providing a framework for optimal use of Domino and SAP R/3 data. Reading and writing data is always through the application layer, maintaining all RFC and R/3 Transaction business rules.

Product Status

This connector is sold as a separate product by Lotus.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and SAP R3 terms.

Enterprise Integration	*SAP R3*
Server	Host
Database	RFC/Transaction
Metadata	Table
Index	Index
Field	Field
Record	Row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Client, System Number, Language
Standard Functionality	Replication not supported, Archive only to R/3, not from
Writeback Support	Full
Writeback Index	NA
Statement Syntax	RFC simple input parameters, separated by commas
Condition Syntax	Valid when placed as follows: param1="*",param2="what ever",...
Connection Properties	Needed only when calling R/3 transactions
Array Transfer	Full
Actions Supported	Reset
Catalog Types	Server, Database, Metadata, Field
Create Types	NA
Drop Types	NA

Lawson Enterprise/400

This connector integrates Domino with the Lawson Enterprise/400 system.

Product Status

This connector is sold as a separate product by Lawson.

Platforms

This connector is supported on the following platforms:

- AS/400

CICS

This connector provides access to CICS DPL programs and CICS 3270 transactions running on any CICS server that can communicate with one of the supported CICS clients.

Product Status

This connector is sold as a separate product by Lotus. It requires Domino R5.x and LEI 3.1.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- AIX 4.14 or later
- OS/2 Warp
- Sun Solaris 2.51 or later

Description

Terminology

The following table lists corresponding Enterprise Integration and CICS terms.

Enterprise Integration	*CICS*
Server	CICS system name
Database	NA
Metadata	Metadata
Index	NA
Field	Field
Record	Commarea or 3270 data stream
Procedure	DPL program name or 3270 transaction transid

IMS

This connector connects Domino applications with IMS applications.

Product Status

This connector is sold as a separate product by Lotus. It requires Domino R5.x and LEI 3.1.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

MQSeries

This connector gives access to most MQSeries-enabled applications, and also provides explicit support for the MQSeries MVS/ESA bridges MQSeries-IMS and MQSeries-CICS. The MQSeries connector uses the MQI to invoke MQSeries services, and can be used with either the MQSeries queue manager or client.

Product Status

This connector is sold as a separate product by Lotus. It requires Domino R5.x and LEI 3.1.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

Description

Terminology

The following table lists corresponding Enterprise Integration and MQSeries terms.

Enterprise Integration	*MQSeries*
Server	Queue Manager
Database	NA
Metadata	Message Format
Index	NA
Field	Field
Record	NA
NA	Queue

Transarc Encina TXSeries

This connector connects Domino applications with Transarc Encina applications.

Product Status

This connector is sold as a separate product by Lotus. It requires Domino R5.x and LEI 3.1.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- AIX 4.14 or later
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later

BEA Tuxedo

This connector connects Domino applications with Tuxedo-managed domains, and responds to remote events and broadcasts generated by Tuxedo and its clients. Tuxedo applications can gain access to Domino data. You need Tuxedo version 6.x or later for the operating system containing Domino or LEI. Both native (local to the Tuxedo domain) and workstation client (remote connections) versions of Tuxedo are supported; choose one or the other.

Product Status

This connector is sold as a separate product by Lotus.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- OS/390
- AS/400

Description

See “LEI Documentation” (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and Tuxedo terms.

Enterprise Integration	*Tuxedo*
Server	WSNADDR/TUXCONFIG
Database	NA
Metadata	Service
Index	Index
Field	Field
Record	An indexed set of fields that emulate a record/row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server (WSNADDR/TUXCONFIG), UserID, Password, Metadata (Service)
Standard Functionality	Direct Transfer and RealTime Activities and any activities that use the same NPX APIs
Writeback Support	Writeback is not available and is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
Writeback Index	Only supported by commands passed as is to TUXEDO service
Statement Syntax	Passed as is to TUXEDO service
Condition Syntax	NA
Connection Properties	ClientName, ApplicationPassword, UseTransactions, TranTimeout, CommitFrequency
Array Transfer	None
Actions Supported	Clear, Reset, Commit, Rollback
Catalog Types	Server, Metadata (Tuxedo WSNADDR’s available, Tuxedo service names)
Create Types	NA
Drop Types	NA

Domino Directory (PNAB)

This connector transfers data to and from Domino Directories (formerly called Public Name & Address Books).

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- AIX 4.14 or later
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400 (requires Domino R5.x and LEI 3.1)

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and Domino Directory terms.

Enterprise Integration	*Domino Directory*
Server	Server
Database	Address Book
Metadata	Form
Index	NA
Field	Field
Record	Document

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, Suffix
Standard Functionality	Full
Writeback Support	Writeback is not available and is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
Writeback Index	None
Statement Syntax	Simple SQL select syntax. For example: select attributename [as renamedattribute] [, attributename] from objectclass [where attributename (= | >= | <= | !=) value]
Condition Syntax	Valid when placed in the following SQL statement: SELECT ... WHERE <condition>
Connection Properties	CreateUserID, OrganizationalUnits, CertifierIDFilesPath, CertifierIDPassword, UserProfileName, MinimumPasswordLength, CreateInternationalID, CreateMailFileNow, OverwriteExistingEntries
Array Transfer	None
Actions Supported	Reset
Catalog Types	Server (no connection required), Metadata, Field
Create Types	NA
Drop Types	NA

Lightweight Directory Access Protocol (LDAP)

This connector transfers data to and from directories using LDAP.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- AIX 4.14 or later
- HP-UX 11.0 or later

- Sun Solaris 2.51 or later
- AS/400 (requires Domino R5.x and LEI 3.1)

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and LDAP terms.

Enterprise Integration	*EDA/SQL*
Server	Server
Database	Directory
Metadata	Object Class
Index	NA
Field	Attribute
Record	Entry or Directory Entry

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, Suffix, UserID, Password
Standard Functionality	Full
Writeback Support	Writeback functionality is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
Writeback Index	NA
Statement Syntax	A simplified SQL select is the only command supported. The where clause follows the format laid out by RFC 1960 as used in LDAP filters. The syntax is shown below.
Condition Syntax	Valid when placed in the following SQL statement: SELECT ... WHERE <condition>
Connection Properties	NA
Array Transfer	NA
Actions Supported	Reset
Catalog Types	Metadata, Field
Create Types	NA
Drop Types	NA

The statement syntax is as follows:

```
select attributename [as renamedattribute] [, attributename]
from objectclass [where (attributename (= | >= | <= |
!(attributename=value) value)]
```

Examples:

Return the distinguished name, given name, surname (sn), and common name (cn) for all entries in the inetorgperson objectclass where the surname has the value "carter":

```
select distinguishedname, givenname, sn, cn from inetorgperson
where (sn=carter)
```

Return the distinguished name, given name, surname (sn), and common name (cn) for all entries in the inetorgperson objectclass where the surname has the value "carter" and the given name starts with "s":

```
select distinguishedname, givenname, sn, cn from inetorgperson
where (&(sn=carter) (givenname=s*))
```

Return the distinguished name, given name, surname (sn), and common name (cn) for all entries in the inetorgperson objectclass where the surname has the value "carter" and the given name starts with "s" or "k" while renaming several of the columns:

```
select distinguishedname as dn, givenname as firstname, sn as
lastname, cn as fullname from inetorgperson where (&(sn=carter)
(| (givenname=s*) (givenname=k*)))
```

Return the distinguished name for all entries in the inetorgperson where the last name is "carter" and the first name does not start with an "s," or all entries with the givenname starting with "ba":

```
select distinguishedname from inetorgperson where
(|(&(sn=carter) (!(givenname=s*))) (givenname=ba*))
```

Novell Directory Services (NDS)

This connector transfers data to and from Novell directories.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0 or later
- OS/2 Warp

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and NDS terms.

Enterprise Integration	*NDS*
Server	NDS Tree
Database	Directory
Metadata	Object Class
Index	NA
Field	Attribute
Record	Entry or Directory Entry

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, Suffix, UserID, Password
Standard Functionality	Full
Writeback Support	Writeback functionality is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
Writeback Index	NA
Statement Syntax	A simplified SQL select is the only command supported. The where clause follows the format laid out by RFC 1960 as used in LDAP filters. The syntax is shown below.
Condition Syntax	Valid when placed in the following SQL statement: SELECT ... WHERE <condition>
Connection Properties	CreateHomeDirectory

continued

Category	*Supported Characteristics*
Array Transfer	None
Actions Supported	Reset
Catalog Types	Server (no connection required), Metadata, Field
Create Types	NA
Drop Types	NA

The statement syntax is as follows:

```
select attributename [as renamedattribute] [, attributename]
from objectclass [where (attributename (= | >= | <= |
!(attributename=value) value)]
```

Examples:

Return the distinguished name, given name, surname (sn), and common name (cn) for all entries in the inetorgperson objectclass where the surname has the value "carter":

```
select distinguishedname, givenname, sn, cn from inetorgperson
where (sn=carter)
```

Return the distinguished name, given name, surname (sn), and common name (cn) for all entries in the inetorgperson objectclass where the surname has the value "carter" and the given name starts with "s":

```
select distinguishedname, givenname, sn, cn from inetorgperson
where (&(sn=carter) (givenname=s*))
```

Return the distinguished name, given name, surname (sn), and common name (cn) for all entries in the inetorgperson objectclass where the surname has the value "carter" and the given name starts with "s" or "k" while renaming several of the columns:

```
select distinguishedname as dn, givenname as firstname, sn as
lastname, cn as fullname from inetorgperson where (&(sn=carter)
(| (givenname=s*) (givenname=k*)))
```

Return the distinguished name for all entries in the inetorgperson where the last name is "carter" and the first name does not start with an "s," or all entries with the givenname starting with "ba":

```
select distinguishedname from inetorgperson where
(|(&(sn=carter) (!(givenname=s*))) (givenname=ba*))
```

File System

This connector allows you to connect to the file system on the Domino or LEI computer.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types. See "Domino Connections Setup" (lccon.nsf) for requirements details.

Terminology

The following table lists corresponding Enterprise Integration and File System terms.

Enterprise Integration	*File System*
Server	NA
Database	Directory
Metadata	File Specification
Index	NA
Field	NA
Record	File

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Database
Standard Functionality	Full. Keyed operations are restricted to the Filename field.
Writeback Support	Full
Writeback Index	NA
Statement Syntax	File specification with optional wildcards
Condition Syntax	File specification with optional wildcards
Connection Properties	Binary
Array Transfer	None
Actions Supported	Clear, Reset, Truncate
Catalog Types	Server, Metadata, Field
Create Types	Database, Metadata
Drop Types	Database, Metadata. Since the only key supported by the File Connection is the Filename field, Replication Activities must use this as the key field.

EDA/SQL connector

This connector transfers data to and from sources supported by EDA Servers from Information Builders (*http://www.ibi.com*). You must install the EDA/Client on the same computer as DECS or LEI. You must also have an EDA Server on the platform where the EDA supported database resides.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

Description

See “LEI Documentation” (leidoc.nsf) for detailed descriptions of the connection properties and data types. See “Domino Connections Setup” (lccon.nsf) for requirements details.

Terminology

The following table lists corresponding Enterprise Integration and EDA/SQL terms.

Enterprise Integration	*EDA/SQL*
Server	EDA Server
Database	Engine
Metadata	Table
Index	Index
Field	Column
Record	Row

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, Database (Engine), UserID, Password
Standard Functionality	Selection, Retrieval, and Insert. Insert requires an EDA Synonym (Metadata) and a relational target.
Writeback Support	NA
Writeback Index	NA
Statement Syntax	ANSI SQL syntax. To execute stored procedures, use the following syntax: EXEC <stored procedure> [parm1] [,parm2] [,parm3] ... [parmn].
Condition Syntax	Valid when placed in the following EDA/SQL statement: SELECT ... WHERE <condition> ORDER BY ...
Connection Properties	CommitFrequency
Array Transfer	Insert
Actions Supported	Clear, Reset, Truncate, Commit, Rollback
Catalog Types	Server, Metadata, Field

Notes

This connector transfers data to and from Domino databases.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows 95
- Windows NT 4.0
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and Domino terms.

Enterprise Integration	*Domino*
Server	Server
Database	Database
Metadata	Form
Index	View
Field	Field or Item
Record	Document

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Server, Database
Standard Functionality	Full
Writeback Support	Full
Writeback Index	When an OrderList is provided, creates a temporary view LEIIndexViewXXX, where XXX is the first available number.
Statement Syntax	Notes formula language. Excludes data modification statements but includes "execute <agent name>" where <agent name> is a Domino agent that is not run on a view.
Condition Syntax	Valid when placed in the following Notes formula: SELECT <condition> ...
Connection Properties	Port, EnforceForm, MultiValueAsText, BulkStore, FilePath, UpdateViews, CreateDatabase, TemplateServer, TemplateDatabase, View, Agent, FullTextQuery, TextMaxLength, FetchViewData, AllForms, ViewResponses, LoadUnid, LoadNoteid, LoadRef, LoadFile, CopyHierarchy, CopyFile, ExecDelete, InsertMail, MailEmbedForm
Array Transfer	None (Bulk insert)
Actions Supported	Clear, Reset, Truncate
Catalog Types	Server, Database (no connection required), Metadata, Index, Field
Create Types	Database (cannot be connected), Metadata, Index
Drop Types	Database (cannot be connected), Metadata, Index

Text

This connector transfers data to and from text files. You define formats for the source and destination data in a ZMerge Information Description (ZID) script.

Product Status

This connector is bundled with DECS and LEI.

Platforms

This connector is supported on the following platforms:

- Windows NT 4.0 or later
- Digital NT Alpha
- AIX 4.14 or later
- OS/2 Warp
- HP-UX 11.0 or later
- Sun Solaris 2.51 or later
- AS/400

Description

See "LEI Documentation" (leidoc.nsf) for detailed descriptions of the connection properties and data types.

Terminology

The following table lists corresponding Enterprise Integration and Text terms.

Enterprise Integration	*Text*
Server	NA
Database	File Name
Metadata	Replacement Substrings
Index	NA
Field	Field, as defined in ZMerge information description (ZID) area
Record	Record, as defined by record delimiter or record length

Supported Characteristics

The following table lists the supported characteristics.

Category	*Supported Characteristics*
Connection Properties	Database
Standard Functionality	Execute, Fetch, Insert
Writeback Support	NA
Writeback Index	NA
Statement Syntax	Use ZID file instead
Condition Syntax	NA

continued

Category	*Supported Characteristics*
Connection Properties	Log, MaxRecSize, MaxRecCount, Trace, FixedRecSize, CheckForQuotes, RecordDelim, Filetype, MinRecSize, SkipRecords, SkipBytes, AppendOutput, FillCharacter, ZID, Connectortype, EchoZid, DayNames, MonthNames, RepStrings, AMString, PMString, MaxText, HeaderRec, TrailerRec, CharacterSet
Array Transfer	None
Actions Supported	Clear, Reset, Truncate
Catalog Types	NA
Create Types	NA
Drop Types	NA

Specify long file names as is (with any embedded spaces). Do not put quotation marks around the file name since the quotation marks will be considered part of the file name.

Appendix B
Character Sets

This appendix provides information about character sets, including character set translation, character sort order, and a list of the character sets supported by the LC LSX.

The list is given as text stream format constants. Any of these values may be used as a stream format for a text stream when creating scripts using the Lotus Connectors LotusScript Extensions. To indicate the character set on the local machine, use the constant LCSTREAMFMT_NATIVE.

Stream Format Constant	*Description*
LCSTREAMFMT_LICS	Lotus International Character Set
LCSTREAMFMT_IBMCP851	MS-DOS PC Greek (CP 851)
LCSTREAMFMT_IBMCP852	MS-DOS PC Eastern European (CP 852)
LCSTREAMFMT_IBMCP853	MS-DOS PC Turkish (CP 853)
LCSTREAMFMT_IBMCP857	MS-DOS PC Turkish (CP 857)
LCSTREAMFMT_IBMCP862	MS-DOS PC Hebrew (CP 862)
LCSTREAMFMT_IBMCP864	MS-DOS PC Arabic (CP 864)
LCSTREAMFMT_IBMCP866	MS-DOS PC Cyrillic Unicode (CP 866)
LCSTREAMFMT_IBMCP437	MS-DOS PC US (CP 437)
LCSTREAMFMT_IBMCP850	MS-DOS PC Western European (CP 850)
LCSTREAMFMT_IBMCP855	MS-DOS PC Cyrillic (CP 855)
LCSTREAMFMT_IBMCP860	MS-DOS PC Portuguese (CP 860)
LCSTREAMFMT_IBMCP861	MS-DOS PC Icelandic (CP 861)
LCSTREAMFMT_IBMCP863	MS-DOS PC Canadian French (CP 863)
LCSTREAMFMT_IBMCP865	MS-DOS PC Norwegian (CP 865)
LCSTREAMFMT_IBMCP869	MS-DOS PC Greek (CP 869)
LCSTREAMFMT_IBMCP899	IBM Code Page 899 (CP 899)
LCSTREAMFMT_IBMCP932	MS-DOS PC Japanese Microsoft Shift-JIS (CP 932)
LCSTREAMFMT_IBMCP942	MS-DOS PC Japanese Microsoft Shift-JIS (CP 942)

continued

Stream Format Constant	*Description*
LCSTREAMFMT_IBMCP891	MS-DOS PC Korean (CP 891)
LCSTREAMFMT_DECMCS	DEC Multinational Character Set
LCSTREAMFMT_EUC	Extended Unix Code
LCSTREAMFMT_KS	MS-DOS Korean - KSC 5601
LCSTREAMFMT_IBMCP949	MS-DOS Korean (CP 949)
LCSTREAMFMT_TCA	TCA
LCSTREAMFMT_BIG5	MS-DOS Taiwan (traditional) Chinese (BIG-5)
LCSTREAMFMT_IBMCP950	MS-DOS Taiwan (traditional) Chinese (CP 950)
LCSTREAMFMT_GB	MS-DOS PRC (simplified) Chinese (GB 2312)
LCSTREAMFMT_IBMCP936	MS-DOS PRC (simplified) Chinese (CP 936)
LCSTREAMFMT_NECESJIS	MS-DOS PC Japanese NEC Shift-JIS (CP 932)
LCSTREAMFMT_ISO646	ASCII
LCSTREAMFMT_ASCII	ASCII
LCSTREAMFMT_ISO88591	ISO Latin-1 US, Western European (ISO-8859-1)
LCSTREAMFMT_IBMCP819	ISO Latin-1 US, Western European (CP 819)
LCSTREAMFMT_ISO88592	ISO Latin-2 Eastern European (ISO-8859-2)
LCSTREAMFMT_IBMCP912	ISO Latin-2 Eastern European (CP 912)
LCSTREAMFMT_ISO88593	ISO Latin-3 Southern European (ISO-8859-3)
LCSTREAMFMT_ISO88594	ISO Latin-4 Northern European (ISO-8859-4)
LCSTREAMFMT_ISO88595	ISO Cyrillic (ISO-8859-5)
LCSTREAMFMT_IBMCP915	ISO Cyrillic (CP 915)
LCSTREAMFMT_ISO88596	ISO Arabic (ISO-8859-6)
LCSTREAMFMT_IBMCP1008	ISO Arabic (CP 1008)
LCSTREAMFMT_ISO88597	ISO Greek (ISO-8859-7)
LCSTREAMFMT_IBMCP813	ISO Greek (CP 813)
LCSTREAMFMT_ISO88598	ISO Hebrew (ISO-8859-8)
LCSTREAMFMT_IBMCP916	ISO Hebrew (CP 916)
LCSTREAMFMT_ISO88599	ISO Latin-5 Southern European (ISO-8859-9)
LCSTREAMFMT_IBMCP920	ISO Latin-5 Southern European (CP 920)
LCSTREAMFMT_HPROMAN	HP Roman (LaserJet)
LCSTREAMFMT_HPGREEK	HP Greek (LaserJet)

continued

Stream Format Constant	*Description*
LCSTREAMFMT_HPTURKISH	HP Turkish (LaserJet)
LCSTREAMFMT_HPHEBREW	HP Hebrew (Laserjet)
LCSTREAMFMT_HPARABIC	HP Arabic (Laserjet)
LCSTREAMFMT_HPTHAI	HP Thai (Laserjet)
LCSTREAMFMT_HPJAPAN	HP Japanese (Laserjet)
LCSTREAMFMT_HPKANA	HP Kana (Laserjet)
LCSTREAMFMT_HPKOREA	HP Korean (Laserjet)
LCSTREAMFMT_HPPRC	HP Simplified Chinese (Laserjet)
LCSTREAMFMT_HPROC	HP Traditional Chinese (Laserjet)
LCSTREAMFMT_IBMCP37	IBM EBCDIC US/Canadian English (CP 37)
LCSTREAMFMT_IBMCP28709	IBM Code Page 28709 (CP28709)
LCSTREAMFMT_IBMCP273	IBM EBCDIC German - Austrian (CP 273)
LCSTREAMFMT_IBMCP278	IBM EBCDIC Finnish, Swedish (CP 278)
LCSTREAMFMT_IBMCP280	IBM EBCDIC Italian (CP 280)
LCSTREAMFMT_IBMCP284	IBM EBCDIC Spanish, Latin American (CP 284)
LCSTREAMFMT_IBMCP285	IBM EBCDIC UK (CP 285)
LCSTREAMFMT_IBMCP290	IBM EBCDIC Japanese (Katakana) (CP 290)
LCSTREAMFMT_IBMCP297	IBM EBCDIC French (CP 297)
LCSTREAMFMT_IBMCP500	IBM EBCDIC International (CP 500)
LCSTREAMFMT_IBMCP277	IBM EBCDIC Danish, Norwegian (CP 277)
LCSTREAMFMT_IBMCP1047	IBM EBCDIC Latin-1 Open Systems (CP 1047)
LCSTREAMFMT_IBMCP1250	Windows Eastern European (CP 1250)
LCSTREAMFMT_IBMCP1251	Windows Cyrillic (CP 1251)
LCSTREAMFMT_IBMCP1252	Windows ANSI (CP 1252)
LCSTREAMFMT_ANSI	ANSI
LCSTREAMFMT_IBMCP1253	Windows Greek (CP 1253)
LCSTREAMFMT_IBMCP1254	Windows Turkish (CP 1254)
LCSTREAMFMT_IBMCP1255	Windows Hebrew (CP 1255)
LCSTREAMFMT_IBMCP1256	Windows Arabic (CP 1256)
LCSTREAMFMT_IBMCP1257	Windows Baltic (CP 1257)
LCSTREAMFMT_IBMCP1363	Windows Korean (CP 1363)
LCSTREAMFMT_MACSCRIPT0	Macintosh Roman (Script 0)

continued

Stream Format Constant	*Description*
LCSTREAMFMT_MACSCRIPT1	Macintosh Japanese (Script 1)
LCSTREAMFMT_MACSCRIPT2	Macintosh Traditional Chinese (Script 2)
LCSTREAMFMT_MACSCRIPT3	Macintosh Korean (Script 3)
LCSTREAMFMT_MACSCRIPT4	Macintosh Arabic (Script 4)
LCSTREAMFMT_MACSCRIPT5	Macintosh Hebrew (Script 5)
LCSTREAMFMT_MACSCRIPT6	Macintosh Greek (Script 6)
LCSTREAMFMT_MACSCRIPT7	Macintosh Cyrillic (Script 7)
LCSTREAMFMT_MACSCRIPT8	Macintosh Right-left symbol (Script 8)
LCSTREAMFMT_MACSCRIPT9	Macintosh Devanagari (Script 9)
LCSTREAMFMT_MACSCRIPT10	Macintosh Gurmukhi (Script 10)
LCSTREAMFMT_MACSCRIPT11	Macintosh Gujarati (Script 11)
LCSTREAMFMT_MACSCRIPT12	Macintosh Oriya (Script 12)
LCSTREAMFMT_MACSCRIPT13	Macintosh Bengali (Script 13)
LCSTREAMFMT_MACSCRIPT14	Macintosh Tamil (Script 14)
LCSTREAMFMT_MACSCRIPT15	Macintosh Telugu (Script 15)
LCSTREAMFMT_MACSCRIPT16	Macintosh Kannada/Kanarese (Script 16)
LCSTREAMFMT_MACSCRIPT17	Macintosh Malayalam (Script 17)
LCSTREAMFMT_MACSCRIPT18	Macintosh Sinhalese (Script 18)
LCSTREAMFMT_MACSCRIPT19	Macintosh Burmese (Script 19)
LCSTREAMFMT_MACSCRIPT20	Macintosh Khmer/Cambodian (Script 20)
LCSTREAMFMT_MACSCRIPT21	Macintosh Thai (Script 21)
LCSTREAMFMT_MACSCRIPT22	Macintosh Laotian (Script 22)
LCSTREAMFMT_MACSCRIPT23	Macintosh Georgian (Script 23)
LCSTREAMFMT_MACSCRIPT24	Macintosh Armenian (Script 24)
LCSTREAMFMT_MACSCRIPT25	Macintosh Simplified Chinese (Script 25)
LCSTREAMFMT_MACSCRIPT26	Macintosh Tibetan (Script 26)
LCSTREAMFMT_MACSCRIPT27	Macintosh Mongolian (Script 27)
LCSTREAMFMT_MACSCRIPT28	Macintosh Geez/Ethiopic (Script 28)
LCSTREAMFMT_MACSCRIPT29	Macintosh EastEurRoman/Slavic (Script 29)
LCSTREAMFMT_MACSCRIPT30	Macintosh Vietnamese (Script 30)
LCSTREAMFMT_MACSCRIPT31	Macintosh extended Arabic/Sindhi (Script 31)
LCSTREAMFMT_MACSCRIPT32	Macintosh uninterpreted symbols (Script 32)

continued

Stream Format Constant	*Description*
LCSTREAMFMT_MACSCRIPT0 CROATIAN	Macintosh Roman variant - Croatian
LCSTREAMFMT_MACSCRIPT0 GREEK	Macintosh Roman variant - Greek
LCSTREAMFMT_MACSCRIPT0 ICELANDIC	Macintosh Roman variant - Icelandic
LCSTREAMFMT_MACSCRIPT0 ROMANIAN	Macintosh Roman variant - Romanian
LCSTREAMFMT_MACSCRIPT0 TURKISH	Macintosh Roman variant - Turkish
LCSTREAMFMT_THAI	MS Thai Windows
LCSTREAMFMT_IBMCP874	MS-DOS PC Thai (CP 874)
LCSTREAMFMT_ISO885911	ISO Thai (ISO-8859-11)
LCSTREAMFMT_TIS620	Thai Industrial Standard (TIS620-2529)
LCSTREAMFMT_UNICODE	Unicode (ISO 10646)
LCSTREAMFMT_IBMCP1200	Unicode (IBM CP 1200)
LCSTREAMFMT_ISO10646	Unicode (ISO 10646)
LCSTREAMFMT_UTF7	Unicode Transformation Formats 7
LCSTREAMFMT_UTF8	Unicode Transformation Formats 8
LCSTREAMFMT_LMBCS	Lotus MultiByte Character Set (LMBCS)
LCSTREAMFMT_DECNRCUK	DEC National Replacement Char - UK
LCSTREAMFMT_DECNRCDUTCH	DEC Nat'l Replacement Char - Dutch
LCSTREAMFMT_DECNRCFINNISH	DEC Nat'l Replacement Char - Finnish
LCSTREAMFMT_DECNRCFRENCH	DEC Nat'l Replacement Char - French
LCSTREAMFMT_DECNRCFRENCH CANADIAN	DEC Nat'l Replacement Char - French Canadian
LCSTREAMFMT_DECNRCGERMAN	DEC Nat'l Replacement Char - German
LCSTREAMFMT_DECNRCITALIAN	DEC Nat'l Replacement Char - Italian
LCSTREAMFMT_DECNRCNORWEG IANDANISH	DEC Nat'l Replacement Char - Norwegian Danish
LCSTREAMFMT_DECNRC PORTUGUESE	DEC Nat'l Replacement Char - Portuguese
LCSTREAMFMT_DECNRCSPANISH	DEC Nat'l Replacement Char - Spanish
LCSTREAMFMT_DECNRCSWEDISH	DEC Nat'l Replacement Char - Swedish
LCSTREAMFMT_DECNRCSWISS	DEC Nat'l Replacement Char - Swiss
LCSTREAMFMT_T61	Teletex T.61

continued

Stream Format Constant	*Description*
LCSTREAMFMT_T50	Teletex T.50
LCSTREAMFMT_ASN1	ANSI Standard Notation (ASN.1)
LCSTREAMFMT_IBMCP856	MS-DOS PC Hebrew (CP 85)
LCSTREAMFMT_IBMCP1004	MS-DOS PC Desktop Publishing (CP 1004)
LCSTREAMFMT_IBMCP1002	IBM EBCDIC DCF (CP 1002)
LCSTREAMFMT_IBMCP1003	IBM EBCDIC US Text Subset (CP 1003)
LCSTREAMFMT_IBMCP1025	IBM EBCDIC Cyrillic (CP 1025)
LCSTREAMFMT_IBMCP1026	IBM EBCDIC Turkish (CP 1026)
LCSTREAMFMT_IBMCP1028	IBM EBCDIC Hebrew Publishing (CP 1028)
LCSTREAMFMT_IBMCP256	IBM EBCDIC International #1 (CP 256)
LCSTREAMFMT_IBMCP259	IBM EBCDIC Symbols Set 7 (CP 259)
LCSTREAMFMT_IBMCP274	IBM EBCDIC Belgian (CP 274)
LCSTREAMFMT_IBMCP275	IBM EBCDIC Brazilian (CP 275)
LCSTREAMFMT_IBMCP281	IBM EBCDIC Japanese (Latin) (CP 281)
LCSTREAMFMT_IBMCP282	IBM EBCDIC Portugese (CP 282)
LCSTREAMFMT_IBMCP361	IBM EBCDIC International #5 (CP 361)
LCSTREAMFMT_IBMCP382	IBM EBCDIC Austrian, German, Switzerland (CP 382)
LCSTREAMFMT_IBMCP383	IBM EBCDIC Belgian (CP 383)
LCSTREAMFMT_IBMCP384	IBM EBCDIC Brazilian (CP 384)
LCSTREAMFMT_IBMCP385	IBM EBCDIC Canadian (French) (CP 385)
LCSTREAMFMT_IBMCP386	IBM EBCDIC Danish, Norwegian (CP 386)
LCSTREAMFMT_IBMCP387	IBM EBCDIC Finnish, Swedish (CP 387)
LCSTREAMFMT_IBMCP388	IBM EBCDIC French, Swiss (CP 388)
LCSTREAMFMT_IBMCP389	IBM EBCDIC Italian, Swiss (CP 389)
LCSTREAMFMT_IBMCP390	IBM EBCDIC Japanese (Latin) (CP 390)
LCSTREAMFMT_IBMCP391	IBM EBCDIC Portugese (CP 391)
LCSTREAMFMT_IBMCP392	IBM EBCDIC Spanish, Philippines (CP 392)
LCSTREAMFMT_IBMCP393	IBM EBCDIC Latin American (Spanish Speaking) (CP 393)
LCSTREAMFMT_IBMCP394	IBM EBCDIC UK, Australian, Hong Kong, Ireland, New Zealand (CP 394)
LCSTREAMFMT_IBMCP395	IBM EBCDIC US, Canadian (English) (CP 395)
LCSTREAMFMT_IBMCP423	IBM EBCDIC Greek 183 (CP 423)

continued

Stream Format Constant	*Description*
LCSTREAMFMT_IBMCP424	IBM EBCDIC Hebrew (CP 424)
LCSTREAMFMT_IBMCP803	IBM EBCDIC Hebrew Character Set A (CP 803)
LCSTREAMFMT_IBMCP870	IBM EBCDIC Eastern Europe (CP 870)
LCSTREAMFMT_IBMCP871	IBM EBCDIC Icelandic (CP 871)
LCSTREAMFMT_IBMCP875	IBM EBCDIC Greek (CP 875)
LCSTREAMFMT_IBMCP880	IBM EBCDIC Cyrillic (CP 880)
LCSTREAMFMT_IBMCP905	IBM EBCDIC Turkish (CP 905)
LCSTREAMFMT_IBMCP948	IBM Extended Taiwanese (CP 948)
LCSTREAMFMT_IBMCP938	IBM Taiwanese (CP 938)
LCSTREAMFMT_IBMCP1381	IBM GBK = GB + Hanzi (CP 1381)
LCSTREAMFMT_IBMCP1386	IBM Traditional Chinese (CP 1386)
LCSTREAMFMT_EACC	East Asian Character Code Set (ANSI Z39.64-1989)
LCSTREAMFMT_JIS	Japanese Information Standard 0201 (JIS 201)
LCSTREAMFMT_CCCII	Chinese Character Code for Information Interchange (Taiwan)
LCSTREAMFMT_XEROXCJK	Xerox CJK
LCSTREAMFMT_IBMCP944	IBM Extended Korean (CP 944)
LCSTREAMFMT_IBMCP934	IBM Korean (CP 934)
LCSTREAMFMT_IBMCP737	MS-DOS PC Greek (CP 737)
LCSTREAMFMT_IBMCP775	MS-DOS PC Baltic (CP 775)
LCSTREAMFMT_ISO6937	Latin chars (non-spacing accents) similar to T.61
LCSTREAMFMT_BASE64	Content-Transfer-Encoding
LCSTREAMFMT_JIS2	Japanese Information Standard 0208 (JIS 208)
LCSTREAMFMT_EUCJ	Extended Unix Code - Japanese
LCSTREAMFMT_EUCT	Extended Unix Code - Taiwanese
LCSTREAMFMT_EUCK	Extended Unix Code - Korean
LCSTREAMFMT_ISOKR	ISO-2022-KR switching: treated as EUCK
LCSTREAMFMT_EUCC	Extended Unix Code - Chinese
LCSTREAMFMT_IBMCP921	Replacement for Lithuanian (CP 921)
LCSTREAMFMT_IBMCP922	Russian (CP 922)
LCSTREAMFMT_KOI8	Cyrillic Internet Support
LCSTREAMFMT_IBMCP720	IBM Code Page 720 (CP 720)

continued

Stream Format Constant	*Description*
LCSTREAMFMT_IBMCP1258	Windows Vietnamese (CP 1258)
LCSTREAMFMT_ISO885910	ISO Latin-6 (ISO-8859-10)
LCSTREAMFMT_JP1TEXT	OSI/JIS X 5003-1987 X.400 Japanese ISP
LCSTREAMFMT_VIQRI	Vietnamese Quoted Readable
LCSTREAMFMT_VISCII	Vietnamese VISCII 1.1 (VICSII)
LCSTREAMFMT_VISCII1	TCVN Vietnamese Orthographic (VCSII-1)
LCSTREAMFMT_VISCII2	TCVN Vietnamese Graphic (VCSII-2)*/
LCSTREAMFMT_IBMCP838	IBM EBCDIC SBCS Thai (CP 838)
LCSTREAMFMT_IBMCP9030	IBM EBCDIC SBCS Thai (CP 9030)
LCSTREAMFMT_IBMCP833	IBM EBCDIC SBCS Korean - extended (CP 833)
LCSTREAMFMT_IBMCP836	IBM EBCDIC SBCS PRC (simplified) Chinese (CP 836)
LCSTREAMFMT_IBMCP1027	IBM EBCDIC SBCS Japanese Latin - extended (CP 1027)
LCSTREAMFMT_IBMCP420	IBM EBCDIC Arabic (CP 420)
LCSTREAMFMT_IBMCP918	IBM EBCDIC Code Page 918 (CP 918)
LCSTREAMFMT_IBMCP1097	IBM EBCDIC Code Page 1097 (CP 1097)
LCSTREAMFMT_IBMCP1112	IBM EBCDIC Code Page 1112 (CP 1112)
LCSTREAMFMT_IBMCP1122	IBM EBCDIC Code Page 1122 (CP 1122)
LCSTREAMFMT_IBMCP1123	IBM EBCDIC Code Page 1123 (CP 1123)
LCSTREAMFMT_IBMCP1129	IBM EBCDIC Code Page 1129 (CP 1129)
LCSTREAMFMT_IBMCP1130	IBM EBCDIC Code Page 1130 (CP 1130)
LCSTREAMFMT_IBMCP1132	IBM EBCDIC Code Page 1132 (CP 1132)
LCSTREAMFMT_IBMCP1133	IBM EBCDIC Code Page 1133 (CP 1133)
LCSTREAMFMT_IBMCP930	IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 930)
LCSTREAMFMT_IBMCP933	IBM EBCDIC EUC Korean Mixed (CP 933)
LCSTREAMFMT_IBMCP935	IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 935)
LCSTREAMFMT_IBMCP937	IBM EBCDIC EUC Taiwan (traditional) Chinese Mixed (CP 937)
LCSTREAMFMT_IBMCP939	IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 939)
LCSTREAMFMT_IBMCP931	IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 931)

continued

Stream Format Constant	*Description*
LCSTREAMFMT_IBMCP1388	IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 1388)
LCSTREAMFMT_IBMCP5026	IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 5026)
LCSTREAMFMT_IBMCP5035	IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 5035)
LCSTREAMFMT_IBMCP300	IBM EBCDIC DBCS Japanese (CP 300)
LCSTREAMFMT_IBMCP834	IBM EBCDIC DBCS Korean (CP 834)
LCSTREAMFMT_IBMCP835	IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 835)
LCSTREAMFMT_IBMCP837	IBM EBCDIC DBCS PRC (simplified) Chinese (CP 837)
LCSTREAMFMT_IBMCP930X	IBM EBCDIC DBCS Japanese (CP 930X)
LCSTREAMFMT_IBMCP933X	IBM EBCDIC DBCS Korean (CP 933X)
LCSTREAMFMT_IBMCP935X	IBM EBCDIC DBCS PRC (simplified) Chinese (CP 935X)
LCSTREAMFMT_IBMCP937X	IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 937X)
LCSTREAMFMT_IBMCP939X	IBM EBCDIC DBCS Japanese (CP 939X)
LCSTREAMFMT_IBMCP931X	IBM EBCDIC DBCS PRC (simplified) Chinese (CP 931X)
LCSTREAMFMT_IBMCP1388X	IBM EBCDIC DBCS PRC (CP 1388X)
LCSTREAMFMT_IBMCP1383	IBM Traditional Chinese (CP 1383)
LCSTREAMFMT_IBMCP806	ISO Devnagiri (CP 806)
LCSTREAMFMT_IBMCP1137	IBM EBCDIC Devnagiri (CP 1137
LCSTREAMFMT_VISCII3	TCVN3 Vietnamese (VCSII-3)
LCSTREAMFMT_TCVN3	TCVN3 Vietnamese (VCSII-3)

Appendix C
DB2 Employee Table Definitions

The following table shows the layout of the DB2 EMPLOYEE table that has been used throughout the examples of this redbook:

DB2 Column	*Type*	*Length*	*Precision*	*Nullable*
EMPNO	CHAR	6	0	No
FIRSTNME	VARCHAR	12	0	No
MIDINIT	CHAR	1	0	No
LASTNAME	VARCHAR	15	0	No
WORKDEPT	CHAR	3	0	Yes
PHONENO	CHAR	4	0	Yes
HIREDATE	DATE	4	0	Yes
JOB	CHAR	8	0	Yes
EDLEVEL	SMALLINT	2	0	No
SEX	CHAR	1	0	Yes
BIRTHDATE	DATE	4	0	Yes
SALARY	DECIMAL	9	2	Yes
BONUS	DECIMAL	9	2	Yes
COMM	DECIMAL	9	2	Yes

Appendix D
MQSC Sample File for the MQSeries Trigger Monitor for Lotus Notes Agents

MQSeries Script Command (MQSC) sample file from SupportPac MA7E details the parameter settings required to use the MQTM for Lotus Notes Agents. The settings will be shown in sections separated by MQSC definitions.

Application Queue

The Notes application queue receives the message from the remote MQ system. It is defined as follows:

```
 DEFINE QLOCAL('NOTE') REPLACE +
*   Queue description
 DESCR('Sample Notes Agent Application Program Queue') +
*   Initiation queue
 INITQ('SYSTEM.SAMPLE.NOTES.AGENT.INITQ') +
*   Process name
 PROCESS('SYSTEM.SAMPLE.NOTES.AGENT.PROCESS') +
*   Triggering is active
 TRIGGER +
*   Trigger on arrival of first message with suitable priority
 TRIGTYPE(FIRST) +
*   Trigger data inserted in the trigger message.
*   For the IBM MQSeries link LotusScript Extension, this can
*   be specified as blank or omitted if not required.
*   For the IBM MQSeries Enterprise Integrator this will
*   typically specify the name of the enterprise service (in
*   the MQEI Definition database) that may be used by the
*   agent to get messages from this queue.
 TRIGDATA('MQEIService1')
```

Initiation Queue

The initiation queue is monitored by the trigger monitor waiting for an initiation message to be placed on its queue. The MQSeries Queue Manager performs this when a message enters the application queue whose INITQ and PROCESS parameters are set. The initiation queue is defined as follows:

```
 DEFINE QLOCAL('SYSTEM.SAMPLE.NOTES.AGENT.INITQ') REPLACE +
*   Queue description
 DESCR('Sample Triggered Notes Agent Initiation Queue')
```

Process Definition

The process definition (sometimes referred to as a process queue), contains the information that is stored in the initiation message. This allows you to define one process for each application queue but only require one initiation queue and hence one trigger monitor. The process is defined as follows:

```
 DEFINE PROCESS('SYSTEM.SAMPLE.NOTES.AGENT.PROCESS') REPLACE +
* Process description
 DESCR('Sample Triggered Notes Agent Process') +
* Application type (22) is Notes agent
 APPLTYPE(22) +
* Application identifier
* This parameter specifies the name of the agent database
* (.nsf)
* file followed by the name of the agent. Note that name of the
* agent may contain spaces but the agent database file name may
* not
  APPLICID('gmqlxtra.nsf MQLSX Link Extra Agent') +
* User data used by the agent. This can be specified as blank
* or
* omitted if not required.
  USERDATA('User data used by MQLSX link extra agent')
```

Special Notices

This publication is intended to help you use Lotus Domino Release 5.0 and related enterprise integration products from Lotus.

The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Domino. See the publications section of the announcement for Lotus Domino and related products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM products, programs, or services may be used. Any functionally equivalent program that does not infringe on any IBM intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available subject to appropriate terms and conditions, including, in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendors, and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate

and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

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

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF, when it becomes available to each customer according to the normal IBM PTF distribution process.

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

AIX
AS/400
DB2
IBM®
MQSeries
OS/2
OS/Warp

The following are trademarks of Lotus Development Corporation in the United States and/or other countries:

LotusScript®
Lotus SmartSuite®
Notes Mail®
NotesPump
NotesSQL
Lotus®
Lotus Domino

Lotus Notes®
RealTime Notes
SmartIcons®

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

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

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

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries. (For a complete list of Intel trademarks see www.intel.com/tradmarx.htm)

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

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

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

International Technical Support Organization Publications

For information on ordering these ITSO publications see, "How To Get ITSO Redbooks."

- *Connecting Domino to the Enterprise Using Java*, IBM form number SG24-5425, Lotus part number CT6EMNA
- *Lotus Domino for AS/400: Integration with Enterprise Applications*, IBM form number SG24-5345, Lotus part number CT7BMNA
- *Lotus Notes 5.0: A Developers Handbook*, IBM form number SG24-5331, Lotus part number CT6HPIE
- *Lotus Solutions for the Enterprise*, *Volume 1. Lotus Notes: An Enterprise Application Platform*, IBM form number SG24-4837, Lotus part number 12968
- *Lotus Solutions for the Enterprise*, *Volume 2. Using DB2 in a Domino Environment*, IBM form number SG24-4918, Lotus part number CT69BNA
- *Lotus Solutions for the Enterprise*, *Volume 3. Using the IBM CICS Gateway for Lotus Notes*, IBM form number SG24-4512
- *Lotus Solutions for the Enterprise*, *Volume 4. Lotus Notes and the MQSeries Enterprise Integrator*, IBM form number SG24-2217, Lotus part number 12992
- *Lotus Solutions for the Enterprise*, *Volume 5. NotesPump*, *the Enterprise Data Mover*, IBM form number SG24-5255, Lotus part number CT69DNA
- *Enterprise Integration with Domino for S/390*, IBM form number SG24-5150

Other Lotus-Related ITSO Publications

The publications listed in this section may also be of interest:

- *The Three Steps to Super.Human.Software: Compare*, *Coexist*, *Migrate; From Microsoft Exchange to Lotus Domino*, *Part One: Comparison*, IBM form number SG24-5614, Lotus part number CT7QTNA

- *The Three Steps to Super.Human.Software: Compare, Coexist, Migrate; From Microsoft Exchange to Lotus Domino, Part Two: Coexistence and Migration*, IBM form number SG24-5615, Lotus part number CT7QWNA
- *Lotus Notes and Domino R5.0 Security Infrastructure Revealed*, IBM form number SG24-5341, Lotus part number CT6TPNA
- *Lotus Notes and Domino: The Next Generation in Messaging. Moving from Microsoft Mail to Lotus Notes and Domino*, IBM form number SG24-5152, Lotus part number CT7SBNA
- *Eight Steps to a Successful Messaging Migration:A Planning Guide for Migrating to Lotus Notes and Domino*, IBM form number SG24-5335, Lotus part number CT6HINA
- *Lotus Notes 4.5: A Developers Handbook*, IBM form number SG24-4876, Lotus part number AA0425
- *LotusScript for Visual Basic Programmers*, IBM form number SG24-4856, Lotus part number 12498
- *Secrets to Running Lotus Notes: The Decisions No One Tells You How to Make*, IBM form number SG24-4875, Lotus part number AA0424
- *Deploying Domino in an S/390 Environment*, IBM form number SG24-2182, Lotus part number 12957
- *Developing Web Applications Using Lotus Notes Designer for Domino 4.6*, IBM form number SG24-2183, Lotus part number 12974
- *The Next Step in Messaging: Case Studies on Lotus cc:Mail to Lotus Domino and Lotus Notes*, IBM form number SG24-5100, Lotus part number 12992
- *Lotus Notes and Domino: The Next Generation in Messaging. Moving from Novell GroupWise to Lotus Notes and Domino*, IBM form number SG24-5321, Lotus part number CT7NNNA
- *High Availability and Scalability with Domino Clustering and Partitioning on Windows NT*, IBM form number SG24-5141, Lotus part number CT6XMIE
- *From Client/Server to Network Computing, A Migration to Domino*, IBM form number SG24-5087, Lotus part number CT699NA
- *Lotus Domino Integration Guide for IBM Netfinity and IBM PC Servers*, IBM form number SG24-2102
- *Lotus Domino Release 4.6 on IBM RS/6000: Installation, Customization and Administration*, IBM form number SG24-4694, Lotus part number 12969
- *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, IBM form number SG24-5163, Lotus part number CT7J0NA
- *AS/400 Electronic-Mail Capabilities*, IBM form number SG24-4703
- *Mail Integration for Lotus Notes 4.5 on the IBM Integrated PC Server for AS/400*, IBM form number SG24-4977
- *Using Lotus Notes on the IBM Integrated PC Server for AS/400*, IBM form number SG24-4779

- *Lotus Domino for AS/400: Installation, Customization and Administration*, IBM form number SG24-5181, Lotus part number AA0964
- *Lotus Domino for S/390 Release 4.5: Installation, Customization & Administration*, IBM form number SG24-2083, Lotus part number AA0963
- *Lotus Domino for S/390 Performance Tuning and Capacity Planning*, IBM form number SG24-5149, Lotus part number CT6XNIE
- *Porting C Applications to Lotus Domino on S/390*, IBM form number SG24-2092, Lotus part number AB1720
- *Managing Domino/Notes with Tivoli Manager for Domino, Enterprise Edition, Version 1.5*, IBM form number SG24-2104
- *Measuring Lotus Notes Response Times with Tivoli's ARM Agents*, IBM form number SG24-4787, Lotus part number CT6UKIE
- *Using ADSM to Back Up Lotus Notes*, IBM form number SG24-4534

Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	*Collection Kit Number*
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
Application Development Redbooks Collection	SK2T-8037
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
AS/400 Redbooks Collection	SK2T-2849
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Networking and Systems Management Redbooks Collection	SK2T-6022
System/390 Redbooks Collection	SK2T-2177

Other Publications

These publications and Web sites are also relevant as further information sources:

- Lotus Enterprise Integration Web site
  `http://www.lotus.com/enterpriseintegration`
- Lotus Developer Web site
  `http://www.lotus-developer.com`
- IBM MQSeries Family Web site
  `http://www.software.ibm.com/ts/mqseries`
- Information Builders Web site
  `http://www.ibi.com`
- SAP AG Web site
  `http://www.sap-ag.de`
- Sun Java Web site
  `http://java.sun.com`
- Microsoft Universal Data Access Web site
  `http://www.microsoft.com/data`
- Microsoft Component Object Model (COM) technologies Web site
  `http://www.microsoft.com/com`

The documentation installed with DECS and LEI include:

- "Domino Connections Setup" (lccon.nsf)
- "DECS Documentation" (decsdoc.nsf)
- "DECS Administrator" (decsadm.nsf)
- "LEI Documentation" (leidoc.nsf)
- "LEI Administrator" (leiadm.nsf)
- "LSX for Lotus Connectors" (lsxlc.nsf)
- "Lotus Connector Java Documentation" (lcjdoc.zip)
- "Lotus Connector 3.0 API Manual" (lc30api.nsf)

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com`
  Search for, view, download or order hardcopy/CD-ROM redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.
- **E-mail Orders**
  Send orders by e-mail including information from the redbooks fax order form to:

	e-mail address
In United States:	usib6fpl@ibmmail.com
Outside North America:	Contact information is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada (toll free)	1-800-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information for customers may be found at http://www.redbooks.ibm.com/ and for IBM employees at http://w3.itso.ibm.com/.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access `MyNews` at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbook Fax Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

❑ Invoice to customer number

❑ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, MasterCard, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

Symbol

A

B

C

D

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Y

Z

ITSO Redbook Evaluation

Lotus Domino R5.0 Enterprise Integration: Architecture and Products SG24-5593-00

Your feedback is very important to help us maintain the quality of ITSO redbooks.

Please complete this questionnaire and return it using one of the following methods:

- Use the online evaluation form at `http://www.redbooks.ibm.com/`
- Fax it to: USA International Access Code +1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?

___Customer ___Business Partner ___Solution Developer ___IBM employee

___None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes _____ No _____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

Printed in the U.S.A.

SG24-5593-00

Part No. CT6QUNA