IBM Lotus Notes and Domino integration technologies for SAP.
Introduction: advantages of integrating SAP and IBM Lotus Domino applications

The IBM® Lotus® software team has offered tools for integration with SAP R/3 beginning with IBM Lotus Notes® and Domino® release 4. This continues to be true with Lotus Notes and Domino 7 software. A wide variety of integration technologies can be used to integrate Lotus Notes and Domino software with SAP applications, addressing customer requirements to extend SAP R/3 managed data to internal staff, customers and business partners. The resulting integration provides a seamless integration framework for optimal utilization of Lotus Domino and SAP R/3 data and services.

Accessing SAP data from Lotus Notes and Domino applications, including mail and workflow-enabled collaborative applications, provides an opportunity for cost savings due to reduced training, support and maintenance of multiple client packages, and improved efficiency. These benefits are extended to many of the client software options supported by the Lotus Domino server. Client options supported by Lotus Domino software that can take advantage of SAP integration include Lotus Notes, IBM Lotus Domino Web Access and select mobile devices. For simplicity, all subsequent references to applications will use the term Lotus Notes application. But keep in mind that these integration approaches can be used from the range of client options mentioned above, except where noted.

In a typical scenario, businesses deploy an SAP server and business modules to manage the back office with enterprise resource planning (ERP) or customer relationship management (CRM) applications. These SAP applications provide high-volume business service processing capabilities, such as employee leave or vacations, work in process schedules, credit and collections, accounting, inventory and sales order management. Lotus Notes and
Domino collaborative applications manage front-end business processes, such as electronic mail, product order placement and associated management approval—using Lotus Domino workflow processes—and self-service applications enabling internal employees and customers to easily access key business data, such as order status, product catalog information, human resources information and problem reporting status.

Case study: Lotus Notes and Domino software plus SAP—the two systems taken independently

How can the Lotus Notes and Domino environment extend the functionality of SAP, or make its services available to more users? It is a common situation that ERP systems like SAP reside in one central location. The end users, on the other hand, are spread all over a country or even all over the world. The costs of maintaining high bandwidth connectivity prevent many companies from considering a distributed environment.

Lotus Domino software works well in this topology. Lotus Domino software can use low bandwidth connectivity options, such as telephone lines, to periodically send e-mails and replicate data from a central site to remote sites. The end users work in their local offices without needing a dedicated network connection to another site.

However, access to SAP services requires user log-in. By deploying Lotus Domino software, together with the tools that connect Lotus Domino server with SAP, we can enable a front end to the services of SAP at the user locations.

For example, one use of SAP is to support a purchase requisition system. How can our remote users see what items are available to be ordered and then order them, when they can't directly access SAP? We can build a Lotus Notes application that does the following:

1. Periodically download the material inventory from SAP to Lotus Domino server, so our users know what they may order, and replicate this database to remote sites

2. Allow the use of a Lotus Notes form to place an order while disconnected

3. Either generate an e-mail from the completed form or save it in a database that replicates, using phone lines, to the central location where the SAP application resides
4. Send the purchase requisition to SAP on a scheduled or an event-driven basis

5. If the transaction is successful, send an order confirmation to the end user via e-mail

6. If the transaction is unsuccessful, send error information to the application administrator

Using this Lotus Notes application, all of our users may take advantage of the services of the SAP system without requiring dedicated online access.

What are the benefits of developing applications to integrate Lotus Domino software and SAP?

The flexible development environment and variety of tools for integration of Lotus Domino server with SAP applications extends traditionally hard-to-reach server data to a wider class of users. In doing so, data access from Lotus Notes applications can reduce the application complexity and end-user training requirements frequently encountered when Lotus Notes users must switch to SAP client applications to access server-managed data.

Using Lotus Domino integration tools for SAP, data may be accessed and updated in SAP in a variety of ways. This technical document is designed to provide an overview of integration options that support real-time access or update, high-volume, scheduled exchange and synchronization, and messaging-based data transfers. The resulting integration can help extend your investments in each application environment with improved information exchange between applications for corporate office employees as well as remote and extended enterprise users.

The following scenarios present a sampling of business requirements that have prompted existing customer organizations to leverage Lotus Domino and SAP integration technologies to help improve everyday business processing.

**Improved analysis of SAP data**—End users of Lotus Notes applications enjoy a familiar user interface to collect and present key business information and transmit updates to SAP.

**Improved distribution of specific SAP data**—Using Lotus Domino replication services eliminates geographic boundaries related to enterprise-wide collaborative efforts, and can help improve data availability across the enterprise.
**Improved reporting of SAP data**—SAP output from multiple application interfaces can be created, maintained and distributed through a single, customized Lotus Notes application.

**Powerful end-user interface to SAP data**—Users of Lotus Notes and Lotus Domino Web Access software can utilize Lotus Notes application interfaces to trigger SAP transactions or perform queries to SAP data, helping to reduce training and deployment costs. In addition, workflow processes based on Lotus Domino server can control end-user interaction with SAP, managing client initiation of updates and inquiries from Lotus Domino to SAP applications.

**Support for remote and mobile users**—SAP reports and extracted data can be stored in Lotus Notes applications, enabling remote and mobile users access to SAP information locally while not connected to the network. Use of efficient Lotus Domino client/server replication processing can help reduce online connection costs.

**Provide SAP data for collaboration in context**—SAP data can be made available to end users in context, where they can then collaborate to achieve their objectives using Lotus Domino extended products, such as IBM Lotus Sametime® software.

What is the typical course of action when integrating Lotus Notes and Domino software with SAP?

This is a wide open question that can have a variety of answers. The big issue is where to start and how complex the first project should be. Lotus Notes and Domino software has provided integration with SAP for more than eight years. Over this period, organizations have found that successful integration between the two systems happens through a staged approach. This means starting with an integrated solution that leverages out-of-the-box capabilities before considering more complex solutions that involve custom applications.

When a business installs Lotus Notes and Domino software, the company often uses the software to provide its messaging infrastructure. People receive their e-mail in their Lotus Notes in-box. SAP can generate e-mail as well. By default, this means that the end users have two mail in-boxes, which is not a desirable situation. Hence, the first step that many customers take is to send mail between the two systems. This can be accomplished by utilizing SMTP or by using the IBM Lotus Domino Mail Transfer Agent (MTA) for R/3. More
details about the Lotus Domino Mail Transfer Agent for R/3 will be provided later in this paper. SAP can be configured to either send mail directly to Lotus Notes in-boxes, or to maintain in-boxes in SAP but automatically forward all e-mail to Lotus Notes in-boxes. This way, users have only one place to look for their e-mail, and that is the Lotus Notes in-box.

The next common integration approach applies to companies that have decided to use SAP Business Workflow. The primary issue is similar to the situation with e-mail – having more than one in-box. Action items generated from SAP Business Workflow, called workitems, are deposited into the user’s SAP in-box. If the user is required to manually log into SAP to check the workitem in-box, especially when this is not the user’s primary in-box, it is likely that workitems will be pending longer than need be. SAP workitems can expire if not processed in a timely fashion.

To help address this issue, IBM offers the Lotus Domino Access to SAP Business Workflow tool. This tool causes the workitem notifications in SAP to be available from the Lotus Notes client. From the notification in the Lotus Notes in-box, the user can invoke a dialog instance of the SAP graphical user interface (SAPGUI) in order to process the workitem. All it takes is a single click of the mouse. The result is that users have only one place to look for all their Lotus Notes messages and SAP workitems, and that is the Lotus Notes in-box.

These first two steps together enable a universal in-box that supports both Lotus Notes and SAP systems.

The next steps include more variation. They generally involve the creation of front-end applications that allow users to work in a Lotus Notes environment but perhaps have SAP data or services available to them, with or without knowing it. Or they might be unique solutions only available by linking different components of Lotus Notes and Domino software to those of SAP. The remainder of this paper provides examples of data and services integration methods.

What integration tools are available from IBM Lotus software?
IBM offers a software product called IBM Lotus Connector for SAP R/3 with MTA and Workflow. This single product is a set of three tools that provide integration between Lotus Domino and SAP R/3 systems.
• Lotus Connector for SAP R/3 software facilitates data transfer and access to SAP R/3 from Lotus Domino server.
• Lotus Domino Mail Transfer Agent for SAP R/3 software allows SAP Office users, SAP Business Workflow processes and R/3 custom applications to exchange mail messages with Lotus Domino mail users or applications.
• Lotus Domino Access to SAP Business Workflow enables Lotus Domino mail users to receive summaries of SAP workflow items in their Lotus Domino in-boxes, from which the open work item can be launched with the click of a button.

<table>
<thead>
<tr>
<th>Comparison—Lotus Domino integration tools for SAP R/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidirectional?</td>
</tr>
<tr>
<td>Real time?</td>
</tr>
<tr>
<td>Transactional?</td>
</tr>
<tr>
<td>Requires coding?</td>
</tr>
<tr>
<td>SAP Application Layer integration?</td>
</tr>
<tr>
<td>Data transfer limits?</td>
</tr>
<tr>
<td>Supports connection pooling?</td>
</tr>
<tr>
<td>Supports LotusScript?</td>
</tr>
</tbody>
</table>

Details about each of these tools are included within this technical paper.

IBM Lotus Connector for SAP R/3—application-to-application data transfer

IBM Lotus software provides several enterprise integration technologies, called Connectors, that are designed to enable seamless integration of enterprise data to and from Lotus Notes applications.

Lotus Connectors are system files, developed using a common object model C interface, called the Lotus Connector API, to provide native, optimized access to enterprise data from Lotus Domino software.
Several Connectors, called Base Connectors, are supplied with Lotus Domino server software:

- **IBM Lotus Connector for DB2®**
- **IBM Lotus Connector for File System**
- **IBM Lotus Connector for Lotus Notes**
- **IBM Lotus Connector for Oracle**
- **IBM Lotus Connector for Sybase**
- **IBM Lotus Connector for OLE DB with Microsoft® SQL Server and Microsoft Access**
- **IBM Lotus Connector for ODBC with Lotus branded DataDirect 5.0 driver (supplied) and with Wire Protocol (available on the IBM Passport Advantage® Web site) for Oracle, Sybase, DB2, SQL Server and IBM Informix® software**

Additional Connectors are available with separately acquired licenses. For example, IBM Lotus Connector for Text is provided with the IBM Lotus Enterprise Integrator® software product. IBM Lotus Connector for SAP R/3 is a Premium Connector packaged with Lotus Connector for SAP R/3 with MTA and Workflow.

The Lotus Connector for SAP can be installed on the Lotus Domino server or on the Lotus Notes client. For example, you can use the Lotus Connector for SAP on a Lotus Domino server running on Microsoft Windows® 2003 to access data from an SAP server operating on an IBM @server® zSeries® platform.
When used in conjunction with Lotus enterprise integration tools, the Lotus Connector for SAP integrates SAP application logic and data with Lotus Notes applications. The Lotus Connector for SAP is supported on Windows, IBM AIX®, Sun Solaris, Linux® and on IBM i5/OS® operating systems. It is supported for use with Lotus Notes, Lotus Domino, IBM Lotus Domino Designer and IBM Lotus Enterprise Integrator, versions 6 and higher software.

The Lotus Connector for SAP R/3 may be used with several IBM Lotus enterprise integration software options: Lotus Enterprise Integrator, Lotus Domino Enterprise Connection Services and Lotus Connector LotusScript eXtensions (LSX).

**Figure 3. How Lotus Enterprise Integrator software works**

**Lotus Enterprise Integrator**—A server-based data transfer product that facilitates scheduled, high-volume transfer and synchronization of data across Lotus Connector sources. The software also offers real-time data access to data sources supported by Lotus Connectors, including SAP. Lotus Enterprise Integrator software, available as a separate product from IBM, includes data transfer template forms for sophisticated scheduled data transfer or real-time data access without programming, and provides support for LotusScript and Java™ programmatic transfers.

**Lotus Domino Enterprise Connection Services**—A technology supplied with Lotus Domino server software that enhances Lotus Notes applications with real-time data access or update capabilities to external source systems supported by Lotus Connectors, including SAP R/3, without programming.
Lotus Connector LotusScript eXtensions—The Lotus Connector LSX enables programmatic access and manipulation of Lotus Connector source data, allowing full programmatic control over data transfer. The Lotus Connector LSX is available with all supported versions of Lotus Domino server and Lotus Domino Designer software. The Lotus Connector LSX can also be used in Lotus Enterprise Integrator scripted activities. All supported Lotus Connectors may utilize the same Lotus Connector API object model, exposed in LotusScript classes, to syntactically access a wide variety of enterprise data sources.

The Lotus Connector for SAP R/3 controls authentication and data transfer from Lotus Notes and Domino software to and from SAP R/3 Server application data. The Connector was developed using SAP’s Remote Function Call Software Development Kit (RFCSDK). The Connector enables execution of any SAP Remote Function Call (RFC) that is remotely callable, as well as all business application programming interfaces (BAPIs) and Transactions using Batch Data Input. Use of the Lotus Connector for SAP technology ensures that data transfers and queries are processed via the SAP application layer, preserving the business logic and data validations contained in SAP Remote Function Call and transaction interfaces that comprise SAP processes. Therefore, reads and writes of SAP data are always performed through the application layer and not by directly accessing back-end database tables. Hence, business rules provided by RFCs and SAP Transactions are maintained.

SAP supplies many Remote Function Calls that comprise the SAP application logic and server processes. It is also possible to create your own RFC functions on a given SAP system to be called from Lotus Notes applications. Creating your own RFC function and executing it through the Lotus Connector SAP R/3 provides a very powerful option for your application. The Lotus Connector for SAP R/3 supports access to all RFCs and BAPIs, and works with SAP R/3 application server releases 3.1i and higher, including releases 3.x, 4.x and 6.x.

IBM Lotus Enterprise Integrator software and the IBM Lotus Connector for SAP R/3
As mentioned previously, Lotus Enterprise Integrator software allows you to transfer data, synchronize data or access data in real time. The ability to transfer and synchronize data between systems accessed through any of the Lotus Connectors means you can integrate more than just Lotus Notes databases with SAP. You can also integrate SAP data with DB2 databases, Oracle, Sybase, Microsoft SQL Server or other systems accessed through ODBC. Real-time data access is strictly between Lotus Notes applications and SAP.
The Lotus Enterprise Integrator server provides the following activities, which are either fully or partially available to the Lotus Connector for SAP R/3.

**Direct Transfer**—Read data from a source and write it to a target. When SAP is the source, then RFCs or BAPIs may be called. When SAP is a target, then RFCs, BAPIs and Transactions may be called. You may choose to read from SAP and write to Lotus Domino, IBM DB2 Universal Database™ or Microsoft SQL Server systems. Again, any connector may read and write with any other connector.

**Replication**—Synchronize two data sets. This feature is supported in a limited fashion by the Lotus Connector for SAP R/3. With SAP, you have a one-way replication out of SAP. You may not replicate to SAP. The reason for this is that a full replication requires four distinct events—create, read, update and delete. It is never the case that one RFC or BAPI will provide all four of these services. For example, the RFC called RFC_CUSTOMER_GET will return customer address data. It will not create new customer records. It will not update customer data. It will not delete customer information. We need other RFCs and Transactions to perform these functions. The Replication activity in Lotus Enterprise Integrator software, however, only lets us define one RFC, BAPI or Transaction to be associated with a connection to SAP—not the four that would be needed in order to fully replicate. Therefore, Lotus Notes applications cannot fully replicate with SAP. But the one-way replication out of SAP is an extremely useful ability that many companies use. We will show an example of this later in this paper.

**Archive**—Read data from a source, write it to a target and delete it from the source. The SAP Connector supports this activity only when SAP is a target. SAP is not supported as a source for the same reasons that replication is not fully supported. A source must support both reading and deleting in this activity, and RFCs, BAPIs and Transactions typically do not support both of these operations.

**Scripted**—Schedule the execution of your LotusScript agents. This feature allows Lotus Enterprise Integrator software to manage running your agents for you, instead of letting the Lotus Domino Agent Manager handle the task. This gives you three benefits. First, the Agent Manager works with a first-in-first-out (FIFO) queue. If you have many agents that need to run at 10:00,
then the first agent runs at 10:00, and the second when the first is finished, and the third when the second is finished, and so on. It may be 11:30 before your critical agent runs, even when you needed it to run at 10:00.

When Lotus Enterprise Integrator software manages the agents, there is no FIFO queue. All agents are started on their respective Lotus Domino servers in new processes independent of the Lotus Domino Agent Manager. If you need them to start at 10:00, then they would all start at 10:00. Of course you need to make sure that the Lotus Domino server can handle the workload. The second benefit is that when these processes complete, they are gone. There is no long-running parent process (as with the Lotus Domino Agent Manager) spawning threads to handle the work under it. If the Agent Manager gets into trouble or starts accumulating unallocated memory, for example, your agents may not run as quickly or at all. But Lotus Enterprise Integrator software starts with fresh processes each time, eliminating these issues. A third benefit is performance. You may control the operating system priority for a particular agent. This allows you to increase or decrease the amount of CPU time that your agent receives. If you need an agent to run at 10:00 and as fast as possible, Lotus Enterprise Integrator software can deliver on that need.

Virtual Fields—Provide a live front end to your enterprise services. The Virtual Fields activity watches a Lotus Notes application for four events—Document Create, Open, Update and Delete. When one of these events happens, Lotus Domino server passes control to the Virtual Fields activity. For example, on Document Create, the data is not saved in Lotus Domino server, but instead a call to BAPI_REQUISITION_CREATE is made and the data is passed to SAP. A Document Open event could perhaps read from SAP table EBAN with RFC_READ_TABLE and show the information about a given requisition. Only the keys to the data in SAP would be saved in the Lotus Notes application. These keys—Requisition Number, for example—are then used as needed to read or write to the back-end system, in this case SAP. The Lotus Connector for SAP R/3 only supports watching one event per Virtual Field activity (for the same reason that replication is not fully supported). But you may have multiple activities, each with a different event, watching the same Lotus Notes application.
There are other Lotus Enterprise Integrator activities, but they are not supported by the Lotus Connector for SAP R/3. We strongly recommend that you take the time to read the white papers and IBM Redbooks™ publications available from ibm.com/lotus/lei that cover Lotus Enterprise Integrator software in great detail.

Lotus Enterprise Integrator example 1: direct transfer from SAP to Lotus Domino server
Using the Lotus Enterprise Integrator administrator database, we have to create two connection documents and an activity document. This Lotus Enterprise Integrator Direct Transfer activity will read from a table under SAP using RFC_READ_TABLE and write that data to a Lotus Notes application.

First, we need to define our connection to our SAP application server and specify which RFC, BAPI or Transaction we want to access. In this case the RFC is RFC_READ_TABLE. The SAP Connector has a special feature to make working with this RFC easier. When you concatenate the source table name to the end of the string RFC_READ_TABLE, the SAP Connector will build in memory structures to let you work with this table as if RFC_READ_TABLE had actually returned the table. In reality, RFC_READ_TABLE returns a data table in which each column of the table is converted to one big text string. Without the SAP connector, you would need to then parse up this 512 byte string and convert the data types back to what they should be. Figure 4 is a sample SAP Connection document for RFC_READ_TABLE where we want to read table KNA1, which contains customer information.
The Hostname is either your SAP application server or the SAP messaging server, if you use load balancing. The Destination is used to identify SAP instances and Groups. System Number tells which TCP port to use for your connection to SAP. The port values are found in your TCP/IP services file. By default, they range from 3300 to 3399 for RFC connections to SAP. System number 0 is port 3300 and system number 99 is 3399. Client is your standard SAP client identifier. Username and Password are the SAP account information. Language is the language you want to use for login. These are one- or two- letter codes, like E or EN for English, D or DE for Deutsch (German) and so on.

Next we define whether this connection is for an RFC, BAPI or Transaction. Then we name the RFC in the RFC or BAPI field. Debug Level allows us to create SAP RFC Trace files that contain useful information. This parameter also allows us to invoke the ABAP debugger, if the SAPGUI is available, to debug what is happening on the SAP side. RFC User tells SAP if you are an RFC only user or not.
Next we need a connection to the target environment where we will write the data, in this case the Lotus Notes application.

![Figure 5. Lotus Notes connection document for Read Table KNA1 Direct Transfer activity](image)

There are many more Lotus Notes Connection form options that are not covered here. In this case, we simply specify which Lotus Domino server to use and the name of the database on the server to use.

Now we are ready to create our Direct Transfer activity. We want to read from table KNA1 and write the results to our SAP Customers database.
We see that the source is our SAP Connection document for RFC_READ_TABLEKNA1. We will read the data from table KNA1. We will have a data target of SAP Customers, which is our Lotus Notes database, and we want to write to the Lotus Notes form called Customer.

The select statement is a bit complex when SAP is the data source. The select statement refers to the Imports elements or Import Tables of a given RFC function. The SAP Connector tries to map the names of the Imports and Tables to the names given in the select statement within a Direct Transfer activity document (with SAP as the source). A comma separates individual elements. Some RFCs and BAPIs require no inputs—in this case, simply leave this field blank. The following is the syntax to use in the select statement.
Structures are defined in this select statement by placing a period “.” at the end of the structure name and then adding the field in the structure after the period. For example, to access field NAME in structure SUPPLIER, you would write the following in your select statement.

SUPPLIER.NAME="String"

Tables are defined in the select statement by placing a period “.” at the end of the table name, then adding a row number followed by another period “.” and finally adding the field name of the table column that you want. For example, your RFC takes as input the table FIELDS. You need to list various data in this table. In order to add the first row to your input table, you would write FIELDS.1. Now you need to specify the column in the table for which you want to set a value. If you wanted to use column NAME in table FIELDS, you would write FIELDS.1.NAME="Mech Ltd". To set another field in the first row of your table, you would write FIELDS.1.STREET="Main Street". Row two is defined as FIELDS.2.NAME="Glog Parts Co". You may not enter values for row two before you have written row one.

Our RFC_READ_TABLE call requires us to define which fields from table KNA1 that we want to have delivered to us. This is done by filling out the desired field names in the input table FIELDS, which is part of RFC_READ_TABLE. Here, that means that to get nine fields from the table, we must have a select statement as follows.


This will fetch fields KUNNR, NAME1, ORT01, STRAS, PSTLZ, PFACH, ANRED, TELF1 and TELFX from table KNA1 for us.

The next section is Field Mapping, where we specify which field in our source should be written to which field in our target.
There are additional performance and scheduling options available in the Direct Transfer activity form, but for this simple example, we are finished. Now all that we need to do is run the activity to have our data read from SAP and written to Lotus Domino server.

Lotus Enterprise Integrator example 2: direct transfer from Lotus Domino server to SAP using BAPI

In our case study earlier in this paper, we discussed creating purchase requisitions in Lotus Domino server at a remote location, sending them to SAP and finally sending a status e-mail to our end user who created the request. The needed pieces are a connection to our source, in this case Lotus Domino server, a connection to our target, in this case SAP and BAPI_REQUISITION_CREATE, and finally a Direct Transfer activity. In order to send the final status mail, we will have our Direct Transfer activity start another activity when it is finished that sends the mail. We will assume that the material inventory from SAP has already been downloaded into our application.

First, we define our source Lotus Notes connection.
Next, we set up our data target SAP connection.

![Connection to SAP](image)

*Figure 8. Lotus Notes connection for BAPI_REQUISITION_CREATE direct transfer*

Next we set up our Direct Transfer activity. Our SAP administrators must tell which fields we need to fill in and tell us the proper range of values for each of those fields.
The select statement is now a Lotus Notes select statement. The statement above is designed to send documents to SAP that were not already sent in a previous run of this scheduled activity. The two fields EXPORTSNUMBER and RETURNMESSAGE are empty unless filled in by this activity. A very important piece of information from BAPI_REQUISITION_CREATE is the requisition number. We also want to see any errors. Beginning in Lotus Enterprise Integrator 3.2 software, this information is available in the Direct Transfer activity. Under Direct Transfer Options, we find the following.
We must first click the option **Target Metadata Is Stored Procedure**. This causes **Accept Output Parameters into Source** to appear in the form. We must also select that option to be able to do the additional field mapping required.

EXPORTNUMBER contains our new requisition number if we were successful, and RETURNMESSAGE contains any error information.

Under General Options we can specify that we want a dependent activity to run after this one. In our case, we want to send our user a status e-mail about what happened.
One more thing to note (in Figure 9) is the metadata selection on the SAP side. You see the “*” symbol displayed there. That means that we want to have access not just to one target table for writing data, but to multiple tables. The SAP connector allows access to either one or all tables (but not just a few of them).

We won’t go into the details of this other activity, except to say that it is a Scripted activity that reads fields from the Lotus Notes documents and then sends an e-mail with the EXPORTSNUMBER to Lotus Notes users or the error to a Lotus Notes administrator.

Lotus Enterprise Integrator example 3: direct transfer from Lotus Domino server to SAP using Transaction

Working with SAP Transactions is a more involved process than working with RFCs or BAPIs. You need to provide to SAP all of the information needed to run your transaction from start to finish. You must define the SAP programs to use, the screens of the transactions and the fields of the screens.

To start, you need to know how to run the transaction from start to finish, and you need to know the appropriate values for the transaction. The best way to do this is to have your SAP administrators record the transaction in the SAPGUI using transaction SM35. This will record everything that happens in a transaction, and it will show you at the end how to call your transaction through batch input mode.

In this example, we want to update SAP Customer address data. This is done in the SAPGUI with transaction XD02. Using transaction SM35, we record our execution of transaction XD02, to see the batch input values. Here is a sample recording.

```
0000  T  XD02
SAPMF02D  0101  X
0000  BDC_CURSOR  RFO2D-D0110
0000  BDC_OKCODE  /00
0000  RFO2D-KUNNR 1001
0000  RFO2D-D0110  X
SAPMF02D  0110  X
0000  BDC_CURSOR  KNA1-NAME1
0000  BDC_OKCODE  =UPDA
0000  KNA1-ANRED  Firma
0000  KNA1-NAME1  Lampen-Markt AG
0000  KNA1-SORTL  LAMPEN
```
This tells us that we called XD02. Program SAPMF02D was used to display screen 0101. We then see four fields under the 0101 section.

BDC_CURSOR was our last mouse click in this screen and may be ignored. It is not required in batch input mode. BDC_OKCODE is the method by which we exited the screen. Every screen must contain this field and its corresponding value. In this case, value /00 means we clicked on the green check symbol to leave the screen. RF02D-KUNNR was the name of the field where we entered the identifier of the customer record for which we wanted to change the address data. RF02D-D0110 was a check box that said we want to modify address data.

Next we see the next screen used to update address data—screen 0110, and then the fields that were used. If there are fields here that we do not wish to update—or allow our users to update—we may remove them. For example, KNA1-LAND1 is country and KNA1-SPRAS is language. Most likely these will never change, so why make them available to our users? Doing so invites the possibility of errors.

The next step is to convert this recording into a format that the SAP Connector expects, as shown in Figure 12.

You can see that many of the fields have been removed from screen 0110. This is to prevent users from being able to update them. Fields that have hardcoded values in SAP need not be mapped in the Direct Transfer activity. This Direct Transfer activity will use the SAP Customer Lotus Notes connection document used previously.
Figure 12. SAP Connection for XDO2 direct transfer

Now we move to our Direct Transfer activity.
We need to build a little logic into our Lotus Notes form, such that when a document is edited, a field called IsMod is set to “1”. This allows us to easily find the modified documents that need to be sent to SAP. It also requires a dependent activity to reset these values after a successful completion, so that the changes are not sent again the next time that this activity runs. You can see the field mapping, which fills in the values that were not hardcoded in the SAP Connection document.

Note: Typically we select the metadata—Form Name for Lotus Notes and Table Name for SAP—using the Action Button “Select Metadata” in Lotus Enterprise Integrator software. When we work with SAP Transactions, this does not work for us, and we must manually enter the transaction code in the Table Name field.
Now, every time that we write to an SAP Transaction, we MUST set the Stored Procedure field.

The *Accept Output Parameters into Source* option was selected to get the results of the transaction. That is what it takes to write to an SAP Transaction. The most important thing to remember is that if you upgrade SAP, or even install patches to it, your transactions may change. If that happens, this activity may need to be updated as well.

Lotus Enterprise Integrator example 4: replication — one way from SAP to Lotus Domino server

Let us change the SAP to Lotus Domino direct transfer example to replicate the customer address data from table KNA1 out of SAP into the Lotus Notes application, instead of just transferring it. This is the best way to regularly update your data target. Replication will only make updates to the data target if something has changed. A Direct Transfer activity, on the other hand, updates all target documents, even when nothing has changed. If your Lotus Domino database replicates with other Lotus Domino servers, this results in the unnecessary replication of documents.

The source and target connection documents are unchanged from the direct transfer example. We need to create a Replication activity document.
In this scenario, we set up SAP as the source and the Lotus Notes application as the target. We need to set a key field. This is what Lotus Enterprise Integrator software uses to match a record in table KNA1 against a document in our Lotus Notes database. Finally we map the fields. But we are not yet finished. We still need to define for RFC_READ_TABLE which fields we want from our table KNA1. We do this by using Replication Options, as shown in Figure 14.

Here it is not called a select statement; it is called a conditional clause. But the concept is the same. We must define which fields will be read, using our select statement syntax for SAP. Again, there are more options available in this...
activity, such as scheduling, but for this example we are finished. We can now replicate SAP customer address data out of SAP and keep our Lotus Notes application current.

**Virtual Fields activities and the Lotus Connector for SAP R/3**

Virtual Fields activities in Lotus Enterprise Integrator software and Lotus Domino Enterprise Connection Services enable real-time access to your back-end services and data. The goal is a live connection to the other system (in this case, SAP), not duplication of the data. We don’t want two copies—one in SAP and one in the Lotus Notes database.

The Virtual Fields feature of Lotus Enterprise Integrator software was so popular and successful that this capability has been made available as a standard Lotus Domino feature called Lotus Domino Enterprise Connection Services (DECS). Virtual Fields activities in Lotus Enterprise Integrator software have some additional capabilities compared to Virtual Fields activities in Lotus Domino Enterprise Connection Services. For the purpose of this paper, our Virtual Fields examples will reflect Lotus Enterprise Integrator capabilities and screen shots.

![Figure 15. How Virtual Fields activities work](image-url)
A Virtual Fields activity watches a given Lotus Notes application for up to four different events that might occur. They are Document Create, Document Open, Document Update and Document Delete. When one of these events occurs, Lotus Enterprise Integrator software is given control from Lotus Domino server on how to respond or what action to take. One difference between a Virtual Fields activity and Lotus Enterprise Integrator batch transfer activities is that one of the two connectors in Virtual Fields always connects to a Lotus Notes application. A Virtual Fields activity can only watch a Lotus Notes database for the four events. It cannot watch another connector for them.

What happens for each of the four events?

**Document Create**—A user creates a new document in the Lotus Notes application. The first time the user saves the document, fields from the document are passed to Lotus Enterprise Integrator software, which attempts to insert them into the back-end system. With a relational database management system (RDBMS), that means inserting a new record into a table. With SAP, it means calling an RFC, BAPI or Transaction. After the data is successfully sent to the back-end system, all of the Virtual Fields are removed from the Lotus Notes document, leaving behind only the key or keys to the data in the back-end system. For example, if we want to create a new customer record in SAP, the customer number could be the key. The customer name and address information would be the Virtual Fields fields, and they would not be stored in the Lotus Notes document.

**Document Open**—A user opens an existing Lotus Notes document. This document contains keys to data in the back-end system. These keys are passed to Lotus Enterprise Integrator software, which, in the case of SAP, calls an RFC, BAPI or Transaction. If in Document Create, we created a new customer record and only stored the customer number in the Lotus Notes application, then the customer number would be sent to RFC_CUSTOMER_GET in order to obtain the customer address data. By the time that the Lotus Notes document is fully opened using the appropriate form, all of the fields are available for display.

The Lotus Notes user has no idea that the address data is not stored in the Lotus Notes database; it is all transparent to the end user. An additional benefit is that the end user does not need to be a human being. It can be a Lotus
Domino Full Text Indexer, for example, or an agent. Regardless of whether the end user is a program or a human being, when Document Open occurs, the Virtual Fields activity processes first. When the document is closed, only the key fields remain (and, of course, any fields unrelated to the Virtual Fields activity document).

**Document Update**—A user edits an existing Lotus Notes document and saves it. All of the data associated with fields in the Virtual Fields activity is passed to Lotus Enterprise Integrator software, which then calls an RFC, BAPI or Transaction in SAP. Finally, the fields associated with the Virtual Fields activity are removed from the document before it is stored in the Lotus Notes database.

**Document Delete**—A user deletes a Lotus Notes document. All of the data associated with fields in the Virtual Fields activity is passed to Lotus Enterprise Integrator software, which then calls an RFC, BAPI or Transaction in SAP. Finally, the document is removed from the Lotus Notes database.

As mentioned earlier, with SAP, only one event may be defined in a given Virtual Fields activity, but multiple activities may watch the same Lotus Notes application. We recommend that you take the time to read the online documentation on Virtual Fields that is provided with Lotus Domino and with Lotus Enterprise Integrator software. Another good source of information is the IBM Redbooks publication, *Implementing IBM Lotus Enterprise Integrator*, available at ibm.com/redbooks.

**Virtual Fields example: document open event**

In this example, a key stored in our Lotus Notes document will be an SAP customer number. When our user opens the document, Lotus Enterprise Integrator software will pass the key to RFC_CUSTOMER_GET, which will return the address data for this customer.

Step one is to create a connection to SAP and RFC_CUSTOMER_GET. You will see that much of the information is available in the connection form for SAP.
There are a few important differences to note. First, two additional fields in this form, called **Table(s)** and **Columns**, are provided to aid the Lotus Enterprise Integrator application developer. The form gives the developer a list of available data fields and their data types. In this example, the value of Table(s) is “*”. When working with Virtual Fields and SAP, the value of Table(s) must always be this symbol. With Virtual Fields, we always want to work with the complete interface of the RFC or BAPI. This is because we have no select statement here and need to be able to map fields into the import parameters and import tables. This means we almost always will work with more than one table when we implement Virtual Fields – the import table (which is a dummy table used to provide access to the import parameters of your RFC or BAPI) and your data table(s).
Another field, called *Debug Level*, is designed to help generate appropriate error information to help diagnose a problem. When this parameter is set to 1, an RFC trace file is created. This file is found in your Lotus Domino program directory and named rfcxx_xxx.trc, where “x” is a number. Go to the end of the file and scroll up until you see RfcReceive and the data that it returns. Here is an example.

>>> RfcReceive...
>>> £101 <unknown> : EXT iwdf9015 >>> FLUSH(READ).
>>> £101 <unknown> : EXT iwdf9015 >>> READ.
000000 | 05000000 05000331 00040000 00040331 ˓→l→l| 000010 | 03360000 00000000 03360336 00040000 ⥺b→b→b→b→b→| 000020 | 00000000 05030512 00000512 ˓→b→b→b→b→| 000030 | 02010000 40455353 47020102 03011445 ˓→MESSG→E| 000040 | 46352020 20202020 20202020 20202020 |F5 | 000050 | 20202020 34313520 436f666b 616e7920 | Company | 000060 | 636f6465 20697320 6e6f7420 64656669 ˓→code is not defi| 000070 | 6c656465 20202020 20202020 20202020 |ned | This format is admittedly not easy to read, but the text on the right reveals the problem. In this case, a Company code is not defined.

Now we are ready to create our Virtual Fields activity.
IBM Lotus Notes and Domino integration technologies for SAP.

The left side of the form specifies the Lotus Notes application that we want to watch. We also define that we want to watch the Customer form for our event in the application. The right side is our SAP Connection to RFC_CUSTOMER_GET.

The field mapping has two sections—the Keys and the Fields. The Fields are known as the Virtual Fields activity fields. Our key must already exist in the Lotus Notes document for this Document Open activity to run successfully.

The Virtual Fields activity offers a service known as Key Initialization. When working with an activity that watches for the Document Open event, you have the option to run a manual process called Initialize Keys from the Actions menu. Using this feature with SAP is most often NOT possible. Let us consider an example to understand why.

Let us say that we want to call BAPI_REQUISITION_GETDETAIL when a user opens a given Lotus Notes document. The key field for this call would be a requisition number. Now, if we select our BAPI_REQUISITION_GETDETAIL activity
document and then run the Initialize Keys agent, it will attempt to get the key values from the only defined source, which in this example is BAPI_REQUISITION_GETDETAIL. The problem is that this BAPI takes the requisition number as input, but is not designed to provide requisition numbers as output. We would need a different RFC or BAPI that can in fact provide the list of requisition numbers. But our one Virtual Fields activity does not let us define two different SAP calls. Thus, use of the Initialize Keys agent will not work for us.

Key Initialization is an important task that needs to be run regularly. This is because new records are quite often created in the back-end system. Without new documents in Lotus Notes software that contain keys for these new records in the back end, the related data will not be available to the Lotus Notes users. If you are using Lotus Enterprise Integrator software, you could easily set up a one-way Replication activity that could call the RFC or BAPI that would give you the needed keys for your Virtual Fields Document Open activity. If you are using Lotus Domino Enterprise Connection Services for Virtual Fields, write a simple script based on the Lotus Connector LSX to refresh your keys on a regular basis.

Let us return to our Document Open example. The data flow is as follows. The contents of the CUSTOMER_TKUNNR field will be passed from the Lotus Notes document to the import parameter IMPORTSKUNNR. Then the call to RFC_CUSTOMER_GET will be made. RFC_CUSTOMER_GET will return the RealTime fields, which will then be displayed in the corresponding fields in the Lotus Notes document as defined by the mapping.

The last item we need to define is which event to watch. In Figure 17 you can see that Open is selected. Provided that we have key documents in our Lotus Notes database, we are ready to start the activity. Typically, companies set these RealTime activities to start up automatically when the Lotus Domino server starts. If the dependent RealTime activity is not running, your Lotus Notes application will not work.
What if you want to call BAPI1, then, depending on the results of this call, call either BAPI2 or BAPI3, and display some data from each of these calls in a single Lotus Notes document? This is application logic that cannot be defined by using forms supplied with Lotus Enterprise Integrator software or Lotus Domino Enterprise Connection Services. Fortunately, a scripted interface to Lotus Connectors allows us to build the logic needed to meet our goals.

In this section we will go through a number of script examples and discuss how they work. The Lotus Connector for SAP R/3 comes with an LCLSX sample database filled with many examples for you to examine and try out.

The Lotus Connector LSX offers four methods: Insert, Remove, Execute and Call. The Insert and Remove methods are not recommended for use with SAP, because these calls do not return a result set. We would not be able, for example, to work with our BAPI export parameters to see what really happened. Due to our increased control in a scripted environment, we can accomplish what we need by using either the Execute or Call method. The Execute method involves coding select statements that can become quite complex. The advantage is that we may define input tables for our calls to SAP with multiple rows of data. The Call method has no select statement, but it only allows a single row of input data at a time. Let us look at examples of each.

Lotus Connector LSX example 1: direct transfer from SAP to Lotus Domino server

Our first example will duplicate the functionality of the Lotus Enterprise Integrator direct transfer scenario which read from table KNA1.

```vbscript
On Error Goto errorhandler
Dim session As New Lcsession

Lcsession is used here so that we may have access to error information.

Dim source As New Lcconnection("sap")
Dim target As New Lcconnection("notes")
```

Here we define our connections, one to SAP and one to the Lotus Notes application, just as we did with Lotus Enterprise Integrator software. There is, however, no requirement that we use the Lotus Notes connector to write to the Lotus Notes application. We may use the native Lotus Domino classes if we choose. In this example, we will use the Lotus Notes connector to try out one of the power features of the Lotus Connector LSX.
Here we define our field lists. These are the source and target objects. They are each defined with a “(10)” at the end. This means that we will be able to work with ten records at a time.

Dim sourcefieldlist As New Lcfieldlist(10)
Dim targetfieldlist As New Lcfieldlist(10)

Here we define which RFC or BAPI we want to call. Notice that the property is called Database. Although the term Database is not related to SAP, this term is used because the Lotus Connector LSX was first developed with RDBMS designers in mind. In this example, we will call RFC_READ_TABLE and read table KNA1.

source.Database = "RFC_READ_TABLEKNA1"

Here we see all of our SAP connection information, just like we did with Lotus Enterprise Integrator software or Lotus Domino Enterprise Connection Services.

source.Connect

If this line executes without resulting in an error, then we are successfully logged in to SAP.

target.Server = "eitech"

target.Database = "sapstdb.nsf"

Now we set up our Lotus Notes connection.

target.Connect

If this line executes without resulting in an error, then we have successfully opened our Lotus Notes database.
source.Metadata = "KNA1"

Here we specify which object will hold the data we want to read.

target.Metadata = "Customer"

Here we specify the Lotus Notes form to use to write the data.

selectstatement = |FIELDS.1.FIELDNAME="KUNNR";| + _
|FIELDS.2.FIELDNAME="NAME1";| + _
|FIELDS.3.FIELDNAME="STRAS";| + _
|FIELDS.4.FIELDNAME="ORT01";| + _
|FIELDS.5.FIELDNAME="ANRED";| + _
|FIELDS.6.FIELDNAME="PSTLZ";| + _
|FIELDS.7.FIELDNAME="PFACH";| + _
|FIELDS.8.FIELDNAME="TELF1";| + _
|FIELDS.9.FIELDNAME="TELFX"|

Now we set up our select statement. For RFC_READ_TABLE, the select statement defines which fields from our table we would like to read.

counter = source.Execute(selectstatement, sourcefieldlist)

Now we execute our select statement to get the data from SAP.

If (counter = 0) Then
MsgBox "No records returned by SAP"
Exit Sub
End If

Call targetfieldlist.MapName(sourcefieldlist,"KNA1ANRED," + _
"KNA1KUNNR,KNA1NAME1,KNA1ORT01," + _
"KNA1PFACH,KNA1PSTLZ,KNA1STRAS," + _
"KNA1TELF1,KNA1TELFX","CUSTOMER_TKUNNR," + _
"CUSTOMER_TNAME1,CUSTOMER_TORT01," + _
"CUSTOMER_TPFACH,CUSTOMER_TPSTLZ,CUSTOMER_TSTRAS," + _
"CUSTOMER_TTELF1,CUSTOMER_TTELFX")

target.MapByName = True
Field mapping is important here, since related fields in our table KNA1 and in the Lotus Notes form may have different names. If the field names are the same, then the MapName call is not needed, but the statement MapByName = True must be included in every script that you write with the SAP Connector.

```vba
Dim counter2 As Integer
    counter2 = source.Fetch(sourcefieldlist, 1, 10)
    While (counter2 > 0)
        Call target.Insert(targetfieldlist, 1, counter2)
        counter2 = source.Fetch(sourcefieldlist, 1, 10)
    Wend
```

Now we will see the power feature mentioned earlier. The Fetch call reads data from our result set. By default, it reads one row of data at a time. Here, we tell it to read ten rows of data in a single step. Then we insert these ten rows of data into the Lotus Notes application in a single step as well. This simple action provides a huge performance boost, compared to methods employed before the Lotus Connector LSX became available. Why? LotusScript is an interpreted language; it is not compiled. By nature, interpreted code runs more slowly than a compiled executable. For a table with 100 rows, reducing the number of read and write steps from 200 to 20 can have a significant impact. The compiled data link libraries (DLL) do all of the work in this case—not the LotusScript interpreter.

Another performance benefit of this approach is the way we write to Lotus Notes software. With traditional LotusScript, we would need to process one line of script for each field in our new Lotus Notes document, plus one to initialize the document and one to save the populated document. In our example, there are nine fields of data in each record. The traditional model would require eleven lines of script for each record found, versus one with the Lotus Connector LSX. Thus, use of Lotus Connector LSX can dramatically reduce the number of lines of script we need to write, and it can increase the performance of our LotusScript code.

```vba
Exit Sub
```

```vba
errorhandler:
Dim Msg As String
Dim Msgcode As Long
Dim status As Integer
Dim result As String
```
If session.status <> LCSUCCESS Then
    status = session.GetStatus(result, Msgcode, Msg)
End If
Msgbox result

This last section is a generic error handler that prints any errors that were encountered along the way.

Lotus Connector LSX example 2: direct transfer to SAP using Call
In the following example, the input values are hardcoded, not read from the Lotus Notes application. This example illustrates how to set up an input field list for our call, which in this case is BAPI_REQUISITION_CREATE.

On Error Goto errorhandler
Dim session As New lcsession

Lesession is used here to have access to error information.

Dim target As New LCConnection("sap")

Here we define our object that will create a connection to SAP.

    target.Database = "BAPI_REQUISITION_CREATE"

Here is the BAPI that we want to access.

    target.Userid = "muster"
    target.Password = "ides"
    target.Client="800"
    target.Destination="EI1"
    target.SystemNo=0
    target.Language="EN"
    target.Server="eitech"

This is standard SAP login information.

    target.Connect

If this line executes without error, then we are successfully logged in to SAP.

Dim fldLst As New LCFieldList
Dim fldresult As New LCFieldList
Dim field As LCField
Dim number As New LCNumeric

target.Metadata = "*"
We want to work with more than one table, so we use the "*" symbol as metadata.

```lisp
Set field = fldLst.Append
("REQUISITION_ACCOUNT_ASSIGNMENTCOST_CTR", LCTYPE_TEXT)
field.text = "0000001000"
```

Now we need to set up our input field list. This is a list of fields and their values, which will be passed to BAPI_REQUISITION_CREATE.

```lisp
Set field = fldLst.Append
("REQUISITION_ACCOUNT_ASSIGNMENTG_L_ACCT", LCTYPE_TEXT)
field.text = "0000400000"
```

```lisp
number.text = "1"
```

```lisp
Set field = fldLst.Append
("REQUISITION_ACCOUNT_ASSIGNMENTPREQ_ITEM", LCTYPE_NUMERIC)
Call field.setnumeric (1, number)
```

```lisp
number.text = "7"
```

```lisp
Set field = fldLst.Append
("REQUISITION_ACCOUNT_ASSIGNMENTPREQ_QTY", LCTYPE_NUMERIC)
Call field.setnumeric (1, number)
```

```lisp
Set field = fldLst.Append
("REQUISITION_ITEMSACCTASSCAT", LCTYPE_TEXT)
field.value = "K"
```

```lisp
Dim money As New LCCurrency
money.value = 75.00
Set field = fldLst.Append
("REQUISITION_ITEMSAMT_BAPI", LCTYPE_CURRENCY)
Call field.setcurrency (1, money)
```

```lisp
Dim flddate As New LCDatetime (2002,05,04)
Set field = fldLst.Append
("REQUISITION_ITEMSDELIV_DATE", LCTYPE_DATETIME)
Call field.setdatetime(1, flddate)
```

```lisp
Set field = fldLst.Append
("REQUISITION_ITEMSDOC_TYPE", LCTYPE_TEXT)
field.text = "NB"
```

```lisp
Set field = fldLst.Append
("REQUISITION_ITEMSITEM_CAT", LCTYPE_TEXT)
field.text = "0"
```
Set field = fldLst.Append
("REQUISITION_ITEMSPLANT", LCTYPE_TEXT)
   field.text = "1000"

number.text = "00001"
Set field = fldLst.Append
("REQUISITION_ITEMSORDERED_ITEM", LCTYPE_NUMERIC)
   Call field.setnumeric (1, number)

number.text = "7"
Set field = fldLst.Append
("REQUISITION_ITEMSQUANTITY", LCTYPE_NUMERIC)
   Call field.setnumeric (1, number)

Set field = fldLst.Append
("REQUISITION_ITEMSDESCRIPTION", LCTYPE_TEXT)
   field.text = "MY REQ"
Set field = fldLst.Append
("REQUISITION_ITEMSUNIT", LCTYPE_TEXT)
   field.text = "EA"
Set field = fldLst.Append
("REQUISITION_ITEMSMAT_GRP", LCTYPE_TEXT)
   field.text = "00107"
Set field = fldLst.Append
("REQUISITION_ITEMSPUR_GROUP", LCTYPE_TEXT)
   field.text = "000"
Set field = fldLst.Append
("REQUISITION_ITEMSCREATED_BY", LCTYPE_TEXT)
   field.text = "SMORRIS"
Set field = fldLst.Append
("REQUISITION_ITEMSNAME", LCTYPE_TEXT)
   field.text = "SMORRIS"
Set field = fldLst.Append
("REQUISITION_ITEMSDISTRIBUTION", LCTYPE_TEXT)
   field.text = ""
Set field = fldLst.Append
("REQUISITION_ITEMSIR_IND", LCTYPE_TEXT)
   field.text = "X"
Set field = fldLst.Append
("REQUISITION_ITEMSGR_IND", LCTYPE_TEXT)
   field.text = "X"

Dim count, counter2 As Integer
target.MapByName = True
We must always set this when using the SAP connector.

    count = target.call(fldLst, 1, fldresult)

This Call method has no select statement. It passes an input field list and returns a result set.

    Dim index As Long
    Call target.Fetch (fldresult, 1, 1)

This code does not use a loop to fetch all rows of data, because this particular BAPI only returns one row of data.

    Set field = fldresult.lookup("EXPORTSNUMBER", index)

This syntax illustrates how to find a given field in our field list.

    If field.text(0) <> "" Then
        Print "EXPORTSNUMBER = " & field.text(0)
    End If

If the content of EXPORTSNUMBER is not blank, our BAPI_REQUISITION_CREATE was successful, and the content of the field is our requisition number.

    Set field = fldresult.lookup("RETURNMESSAGE", index)
    If field.text(0) <> "" Then
        Print "RETURNMESSAGE = " & field.text(0)
    End If

If the content of RETURNMESSAGE is not blank, our BAPI_REQUISITION_CREATE was not successful, and the content of the field is the error message.

    Exit Sub

errorhandler:

    Dim Msg As String
    Dim Msgcode As Long
    Dim status As Integer
    Dim result As String

    If session.status <> LCSUCCESS Then
        status = session.GetStatus(result, Msgcode, Msg)
    End If
    MsgBox result
This last section is a generic error handler that prints any errors encountered along the way. Please note that if the BAPI returns a message in the RETURNMESSAGE field, this error handler code will not be run. The reason is that, although the result was not what we had hoped for, technically the BAPI did successfully complete. It is up to us to interpret the information in RETURNMESSAGE in our LotusScript to provide the real error handling for this BAPI.

Lotus Connector LSX example 3: direct transfer to SAP using Execute

We will now repeat the previous example using the Execute method instead of Call. You will see that in our requisition request, we may order more than one line of items. That can be accomplished only with the Execute method.

```languagelotusscript
On Error Goto errorhandler
Dim session As New lcsession
Dim target As New LCConnection("sap")

target.Database = "BAPI_REQUISITION_CREATE"
target.UserID = "muster"
target.Password = "ides"
target.Client = "&00"
target.Destination = "EI1"
target.SystemNo = 0
target.Language = "EN"
target.Server = "eitech"

target.Connect

Dim fldList As New LCFieldList
Dim fldResult As New LCFieldList
Dim field As LCField
Dim number As New LCNumeric

target.Metadata = "*"

Dim selectstatement As String

Now we must set up our select statement. In this example, the information is hardcoded.

```
At this point, you can see that our select statement syntax writes a second row to the input table called REQUISITION_ITEMS. This can only be done with a select statement. We cannot do this with the input field list approach used in the Call method.
IBM Lotus Notes and Domino 
integration technologies for SAP.

Page 45

Dim count, counter2 As Integer
target.MapByName = True
count = target.Execute(selectstatement, fldresult)

Dim index As Long
Call target.Fetch (fldresult, 1, 1)
Set field = fldresult.lookup("EXPORTSNUMBER", index)
If field.text(0) <> "" Then
    Print "EXPORTSNUMBER = " & field.text (0)
End If
Set field = fldresult.lookup("RETURNMESSAGE", index)
If field.text(0) <> "" Then
    Print "RETURNMESSAGE = " & field.text (0)
End If

After the call to Execute, everything else is the same as in the previous example.

Exit Sub

errorhandler:

Dim Msg As String
Dim Msgcode As Long
Dim status As Integer
Dim result As String
If session.status <> LCSUCCESS Then
    status = session.GetStatus(result, Msgcode, Msg)
End If
Msgbox result

So far in this paper, we have touched on the major features of the Lotus Connector for SAP R/3. When used together with Lotus Enterprise Integrator software and the Lotus Connector LSX, Lotus Connector for SAP R/3 is a powerful tool for connecting Lotus Notes applications to SAP.

Let us now explore Lotus Domino Mail Transfer Agent for R/3 and Lotus Domino Access to SAP Business Workflow.

Lotus Domino Mail Transfer Agent for R/3

When a business process is based on a transaction-oriented system, it is important to confirm that individual steps complete. SAP addressed this need by creating the SAP Connect interface, which supports a number of external
interfaces, including e-mail. This interface provides for a robust level of logging, delivery confirmations, return receipts and rich text support.

IBM Lotus software developed the Lotus Domino Mail Transfer Agent for R/3 to connect to this interface to provide a bidirectional mail interface with Lotus Domino server.

Why use the Mail Transfer Agent? Having more than one mail in-box is not desirable. One in-box will be primary, and the others will be slowly ignored. With the Mail Transfer Agent, it is possible to get all Lotus Domino and SAP e-mail in the Lotus in-box, making this problem disappear.

Design
The Lotus Domino Mail Transfer Agent for R/3 is made up of two programs: the Object Server and the Object Client. Since both Lotus Domino administration and SAP administration skills are needed to implement this tool, typically two people are involved. The first part of the configuration applies to the Lotus Domino side.

Object Server
The Object Server is both an RFC Server application and a Lotus Domino server add-in task. We may run multiple instances of the Object Server on a single Lotus Domino server, thus allowing one Lotus Domino server to collect mail from many SAP systems.
On startup, the Object Server refers to two files: saprfc.ini and notes.ini.

The saprfc.ini file contains connection information that will allow the Object Server to locate its corresponding SAP system and register itself to that system. Here is an example of a typical configuration in this file.

```
DEST=eitech_objserv
TYPE=R
GWHOST=eitech.ibm.com
GWSERV=sapgw00
PROGID=eitech/EITECH
SYSNO=00
```

DEST is a case sensitive string that will be used on the command line to load the ObjServ task on the Lotus Domino server. It specifies the section in saprfc.ini that ObjServ should use. TYPE=R indicates that this is an RFC Server application. Unlike an RFC Client application, an RFC Server application registers itself with SAP; it does not log in. It notifies SAP that it is available to provide a service and specifies how to call it. In our case, the service made available is a mail acceptance service.

GWHOST is our SAP system. GWSERV is the gateway used to connect to SAP. PROGID is the name under which this Object Server registers itself with SAP. It is case sensitive. SYSNO is the TCP/IP port used to connect to SAP. In this case, 00 means use port 3300 to connect.

The next file needed by the Object Server is notes.ini. It contains configuration parameters to determine how the Mail Transfer Agent runs. Here are the most commonly used parameters.

```
#ObjServerLogLevel=1
```

This variable lets us control the degree of detail for logging. 1 is selected most often. 2 and 3 provide further detail for debugging purposes.

```
#ObjServerDontAttachBody=1
```

By default, the text found in the body of your e-mail message body will also be attached to the mail as a file. Setting this parameter to 1 prevents the creation of the attachment.
This is an example of the syntax to configure a specific instance of the Object Server, should we need to run more than one. An instance is a command line option that lets us define which notes.ini variables we want to use. If we load the ObjServ task with the instance identifier of 2, it will first search for all notes.ini variables with names that end with the number 2 (for example, $ObjServerDontAttachBody2). If a given variable is not found, it looks for the same variable without a number at the end ($ObjServerDontAttachBody). If that is not found, it will use the default value for the variable.

SAP used RAW format for text files. Set this to 1 to change RAW to TXT on file attachments.

By default, SAP uses Internet addressing for e-mail that it sends to Lotus Domino server. If you are not using Internet addresses, these parameters allow you to convert certain characters as desired and to allow spaces in e-mail addresses if necessary.

The Object Server will check every four seconds to see if it is still connected to SAP. If not, it will attempt to reconnect after a specified amount of time. The default is every five minutes. Here it has been changed to every one minute.

This variable allows you to turn off the reconnect feature.
If you are sending reports through the Mail Transfer Agent, you may want to use a monospace font to ensure that columns line up when the e-mail reaches its destination. Set this parameter to 1 to obtain this behavior.

$\texttt{ObjServerMinConfirmation}=6$

This variable lets you turn off delivery confirmations and return receipts. Since most companies that use the Mail Transfer Agent route mail in only one direction—from SAP to Lotus Domino server, this variable is widely used.

$\texttt{ObjServerAddressIsInternet}=1$

By default, additional information is stored with e-mail addresses for use with delivery confirmations and return receipts. If you have turned off delivery confirmations and return receipts, or are not routing mail back to SAP at all, set this parameter to 1 to disable the storage of information that is never used.

For a complete list and detailed descriptions of Object Server variables, please refer to the online documentation supplied with the Mail Transfer Agent software.

To load the Object Server on Lotus Domino server, the following syntax is used.

```
> load objserv -D<DEST string in your saprfc.ini, case sensitive> <Object Server instance number - not SAP instance number>
```

Which, for example, might look like the following.

```
> load objserv -Deitech_objserv 2
```

If you are running multiple instances of the Object Server, it is best to make copies of the executable and assign each copy a unique name. This allows you to start and stop each copy individually on the Lotus Domino server.

The following is an example.

```
> load objserv01 -Deitech_objserv 1
```

The instance identifier is an optional parameter. It is not needed if you are only running one instance.
The second part of the configuration requires SAP administrator involvement. It is made up of two parts:

1. Configuring an RFC Destination in transaction SM59. This defines what is registering itself with the SAP system and how SAP can contact it.

2. Configuring the SAP Connect interface with transaction SCOT. This defines the mail node that will allow SAP to route messages to your RFC Destination, which will point to the Object Server.

Once the configuration steps for both Lotus Domino software and SAP are complete, your environment is set up to route mail from SAP to Lotus Domino server.

Object Client

The second application incorporated into the Lotus Domino Mail Transfer Agent for R/3 is the Object Client. Like the Object Server, the Object Client is both an RFC application and a Lotus Domino server add-in task. The Object Client sends mail from Lotus Domino server to SAP. Mail includes standard e-mail messages as well as status notifications, such as delivery confirmations and return receipts. As mentioned earlier, many customers choose not to run this program, because they want no mail routed to SAP. They expect their users to get all of their mail through their Lotus Notes in-boxes.

On startup, the Object Client reads the same saprfc.ini and notes.ini files that the Object Server uses. Here is a sample saprfc.ini section for the Object Client.

```
DEST=lotcc09_objclnt
TYPE=A
ASHOST=lotcc09.ibm.com
SYSNR=00
```

DEST is a case-sensitive identifier that is referenced by the $SapDestination variable in notes.ini. TYPE=A indicates that this is an SAP client application. This means the application will not register itself with SAP. Instead, when there is mail to be sent, the Object Client will log in to SAP and send it. ASHOST is the SAP application server. SYSNR is the TCP/IP port used to connect to SAP—in this case, 00 refers to port 3300.

Variables in the notes.ini file are also involved in the Object Client configuration. As with the Object Server, a number at the end of the variable name means that the variable is specific to a given instance of the Object Client.
$SapDestination1=sapaix_objclnt

Indicates which sapfc.ini section to use. The 1 at the end means that this value applies only to instance 1 of the Object Client.

$SapClient1=800

Defines which SAP client to use.

$SapUsername1=muster

Specifies the account that our RFC Client application will use to log in to SAP.

$SapLanguage1=e

This is a language identifier. For example, English = E or EN, Deutsch (German) = D or DE, and so on.

$SapDestination2=lotcc09_objclnt
$SapClient2=800
$SapUsername2=muster
$SapLanguage2=en
$SapClient2=800
$ObjClientLogLevel=2

This variable is typically set to 1. A value of 2 is set for more robust logging, best used to debug problems.

$ObjClientSearchChars1="@ "
$ObjClientSearchChars2="@ "
$ObjClientReplaceChars="&_"

Lotus Notes mail addresses often include special characters, such as “@” and spaces. These variables convert these characters into an Internet address format that SAP will accept.

$ObjClientDontReconnect=1

Setting this variable to 1 disables the reconnect feature.

$ObjClientReconnectDelay1=1

If we have the reconnect feature enabled, this parameter is used to specify an interval different from the default of every five minutes for attempting to reconnect after a disconnect is detected.
For the Object Client, we need to create a foreign domain document in the Lotus Domino directory. The content of this document defines the mail database where Lotus Domino server should place mail for our SAP system. When the Object Client finds a message in this mail database, it will route the message to SAP.

If you run multiple instances of the Object Client, each instance will need its own mail database.

To load the Object Client, run the following command on the Lotus Domino console.

```
> load objclnt Password [ClntNo [SapMail [SapServer]]]
```

Password is the password of the user named in the $SapUsername variable of notes.ini. ClntNo is the instance number used to find specific notes.ini variables, if you are using more than one instance. SapMail is the path to the mail database used for the Object Client. The default path is mail\sapconn.nsf. If the mail database does not reside locally with the Object Client, the SapServer parameter, contrary to what the name would suggest, is the name of the Lotus Domino server where the mail database can be found.

For a complete list of variables and information on the Object Client, please see the Mail Transfer Agent documentation.

Lotus Domino Mail Transfer Agent for R/3 and SAP support for SMTP

IBM typically prefers a standards-based approach, if one is available. SAP R/3 Release 4.7 (basis version 6.x) has implemented SMTP natively in the SAP kernel. SAP has tested this approach with Lotus Domino server, and has determined that it offers a superior mail interface between Lotus Domino server and SAP. In fact, we currently recommend using SMTP with SAP R/3 Release 4.7 and later. When SAP stops supporting SAP R/3 releases 3.x and 4.x, we expect to discontinue the Lotus Domino Mail Transfer Agent for R/3 in favor of SMTP.

Lotus Domino Access to SAP Business Workflow

Even if you set up your e-mail to forward from SAP to Lotus Domino server, the workitem notifications found in the SAP Workitem in-box do not forward. This is because they are not actually e-mail messages. They are pointers to SAP Business Workflow applications. For quite some time, this meant that
Lotus Notes users who participated in SAP workflow had two in-boxes: a Lotus Notes in-box for e-mail and Lotus Domino workflow, and the SAP workitem in-box for SAP workflow.

This situation can be counterproductive for SAP workflow and, accordingly, can have a negative impact on your business. Workitems in SAP may have an expiration date. If the workflow step that expires is the approval on a customer order, the result may be an unhappy customer. Or, if a workitem is not processed in a timely fashion, you may not reorder inventory on time, and your customers may experience unreasonable delays.

Lotus Domino Access to SAP Business Workflow is designed to address these types of issues by making SAP workitem notifications visible in the Lotus Notes user's mail in-box.

Integration with SAP workflow items is done by adding design elements to your users' Lotus Notes mail databases. One of these is a periodic agent. This agent will access a local profile (or multiple profiles) that contains SAP connection and account information. A logon to SAP would be made, and the workitem notifications would be obtained. These workitem notifications are converted to documents that appear in the Lotus Notes in-box.

Your Lotus Notes user would then see a new e-mail message—not from a person, but instead from a given SAP system. It opens up with a special SAP workitem form. This form will display information about the workitem to be
processed. The form also contains action buttons, such as Display and Execute. Clicking on Execute will invoke a dialog instance of the SAPGUI, log the user in, take the user into the workitem in-box and finally execute the workitem. When finished, the user automatically returns to the Lotus Notes application.

The default is to run the periodic agent every two hours. It is possible for an SAP workitem to be sent to multiple users. The first user to process it takes control of the workitem, and it is no longer available to the other users. This could lead to a situation in which a workitem notification that is no longer valid is found in a Lotus Notes in-box. The next run of the periodic agent will clean this up.

Local profiles are created in a central administration database that contains information about the SAP systems and a mapping of Lotus Notes user names to SAP user names. These profiles are then mailed to the Lotus Notes user and, using LotusScript, are hidden in the Lotus Notes user's in-box for use by the periodic agent and the SAP workitem form.

If you are using SAP R/3 4.6C or higher, it is possible to update your SAP password from Lotus Domino Access to SAP Business Workflow, in the event your password has expired. Lotus Domino Access to SAP Business Workflow will catch the expired condition and prompt the user to enter a new password. If the new password meets the SAP security settings, it will be accepted by SAP, and then the local profile would be updated.

Performance considerations
Many people worry unnecessarily about the performance impact of the periodic agent on Lotus Domino server and on SAP. For example, if I have 1,000 people on a Lotus Domino server who use this product, might I sometimes have 1,000 connections from these agents to SAP? The answer is no. The maximum number of connections from any one Lotus Domino server will be the number of agent manager processes that you have running on it. If two processes are running, then two periodic agents may run at once—meaning two concurrent connections to SAP. Two agents running simultaneously has little effect on the Lotus Domino server. For 1,000 users, there would typically be a nearly constant but relatively small load on both Lotus Domino server and SAP.
Summary
IBM Lotus Notes and Domino software provides a proven platform for collaborative applications, while SAP is one of the established leaders in enterprise resource planning systems. Integrating your SAP enterprise data into Lotus Notes applications can help improve your return on investment for not only Lotus Notes and Domino software, but for SAP. Providing a front end for your SAP data and business processes with Lotus Notes software can help you reduce your administrative and end-user training costs. You may also be able to increase your utilization of SAP, because end users will access SAP using the same familiar client interface they use to access the rest of their Lotus Notes applications, and their SAP data can now be accessed offline.

IBM Lotus Connector for SAP R/3 with MTA and Workflow software delivers a robust solution for integrating custom applications, mail and workflow. Whether it be routing mail between the two systems, surfacing SAP workflow items in Lotus Notes mail, integration of SAP data in Lotus Notes applications or updating SAP with data from Lotus Notes databases, Lotus Connector for SAP R/3 with MTA and Workflow software gives you the flexibility to implement a solution that meets your business needs.

For more information
To learn more about IBM Lotus Connector for SAP R/3 with MTA and Workflow software, visit:

ibm.com/lotus/sapconnector