Lotus. software

IBM

Lotus Notes access for SAP solutions

Out-of-the-box SAP integration for your Lotus Notes users

Advanced customization techniques

Additional integration features

Philip Monson
Spencer Shropshire
Kathy Howard
Frank Droege

Redpaper

ibm.com/redbooks

International Technical Support Organization

Lotus Notes access for SAP solutions

February 2007

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

First Edition (February 2007)

This edition applies to IBM Lotus Notes Versions 7.0.1 and 7.0.2.

Contents

Notices

Preface

IBM® Lotus® Notes® access for SAP® solutions is a new feature of the Lotus Notes client that provides business value for companies using both Lotus Notes software and SAP enterprise systems. It lets you access SAP information and business processes in your familiar Lotus Notes messaging and collaborative environment. By making it easier for employees to access the information they need, from a single interface, use of this feature can result in higher user productivity and faster decision making with the opportunity to reduce training costs.

The Lotus Notes access for SAP solutions feature builds on proven IBM and SAP integration technology that has been on the market for eight years. The new capabilities extend this integration into calendars and scheduling, contact management, workflow processing, and other common business tasks. This feature leverages Lotus Notes strengths in personal information and workflow management to complement SAP application processes.

This IBM Redpaper starts with an introduction to the basic, out-of-the-box functionality of Lotus Notes access for SAP solutions. It then goes much deeper into customization techniques to provide the knowledge you need to meet your organization's integration needs.

The team that wrote this Redpaper

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

Philip Monson is a Project Leader at the ITSO Lotus Center in Cambridge, MA. Phil has been with Lotus/IBM for 16 years, joining the company when the early versions of Notes were rolled out for internal use. He has served in management, technical, and consulting roles in the IT, Sales, and Development organizations.

Spencer Shropshire is a Software Engineer with the level 2 Support organization in IBM Software Group, Lotus.

Kathy Howard is the UI Designer for the Lotus Notes access for SAP solutions development team.

Frank Droege is a Senior Consultant with BinaryEdge GmbH, Hamburg, Germany. He started out as a student with Lotus Notes 3.0 in 1993 and has since then achieved all important certifications in the Lotus field. Apart from the integration of SAP and other systems with Notes/Domino®, he has specialized in workflow and Web applications. When he is working offline, Frank does consulting in the business process management sector using tools such as UML.

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

Bob Balaban, Notes access for SAP solutions Architect, IBM Software Group, Lotus

Rocky Oliver, Senior Software Engineer, IBM Software Group, Lotus

Peter Janzen, Senior Product Manager, IBM Software Group, Lotus

Kendra Bowker, Technical Writer, IBM Software Group, Lotus

Scott Morris, Manager, LEI Development, LEI and Domino L3 Support, IBM Software Group, Lotus

Christian Holsing, IBM SAP Center of Competency, IBM Germany

Become a published author

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

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

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

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

1

Introduction

IBM Lotus Notes access for SAP solutions is a new feature of the Lotus Notes client that provides business value for companies using both Lotus Notes software and SAP enterprise systems. It lets you access SAP information and business processes in your familiar Lotus Notes messaging and collaborative environment. By making it easier for employees to access the information they need, from a single interface, use of this feature can result in higher user productivity and faster decision making with the opportunity to reduce training costs.

The Lotus Notes access for SAP solutions feature builds on proven IBM and SAP integration technology that has been on the market for eight years. The new capabilities extend this integration into calendars and scheduling, contact management, workflow processing, and other common business tasks. This feature leverages Lotus Notes strengths in personal information and workflow management to complement SAP application processes.

Lotus Notes access for SAP solutions is delivered with customization in mind. Using IBM Lotus Domino Designer®, developers can see all the code. And the Notes access for SAP solutions Install database lists all the new and modified design elements used by the feature, and provides a mechanism for building the production templates according to the capabilities you want to deploy to your users.

This chapter provides you with:

- Product overview
- Overview of SAP integration
- Audience and goals of this Redpaper

1.1 Product overview

The feature, included in the Lotus Notes 7.0.2 maintenance release, provides seven sets of integration capabilities as listed here. The new and modified Lotus Notes templates can be deployed to any set of users, or across your entire organization. Each user of these templates must be a licensed user of both Lotus Notes and the appropriate SAP applications.

- Time reporting: Employees can report billable (or other) time to SAP applications directly from the Lotus Notes calendar.
- Vacation/leave request: Employees can request vacation or leave time from their manager using the Lotus Notes calendar and Lotus Notes e-mail. Approved time is recorded in an SAP application.
- Report generation: Employees can schedule or run a report from SAP applications using the Lotus Notes client.
- Contact management: Employees can look up contact information from the mySAP™ HR and mySAP CRM systems and add it to their personal contacts in Lotus Notes.
- SAP workflow integration: Employees can review and process SAP workflow items using the Lotus Notes client.
- Employee self service/manager self service: Employees can update and managers can view HR information, such as name, address, and telephone, using the Lotus Notes client.
- Meeting scheduling: SAP users can track the meetings they schedule using SAP CRM on the Lotus Notes calendar.

For Notes application developers and administrators, the news is just as exciting.

Since 1999, IBM has provided developers the tools to build Notes applications that integrate with SAP. These same tools were used to build Lotus Notes access for SAP solutions—and you can use these same tools to modify it:

- Domino Designer
- LotusScript
- Lotus Connector for LotusScript Extensions (LC LSX)

The IBM Lotus Notes access for SAP solutions feature initially supports SAP R/3® 4.6C.

1.2 Overview of SAP integration

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 solution enterprise data into Lotus Notes applications can help improve your return on investment for not only Lotus Notes and Domino software, but also for SAP solutions. Providing a front end for your SAP application data and business processes with Lotus Notes software can help you reduce your administrative and user training costs.

You might also be able to increase your utilization of SAP solutions as well, because users will access SAP solutions using their tools at hand—the familiar client interface they use to access the rest of their Lotus Notes applications. Plus, their SAP solution data can now be accessed offline. IBM Lotus Connector for SAP Solutions technology delivers robust solutions for integrating custom applications and workflows. Whether it be surfacing SAP solution workflow items in Lotus Notes mail, integration of SAP solution data in Lotus Notes applications, or updating SAP solutions with data from Lotus Notes databases, IBM Lotus software gives you the flexibility to implement a solution that meets your business needs.

Figure 1-1 shows the high-level pieces of the Notes access for SAP solutions feature.

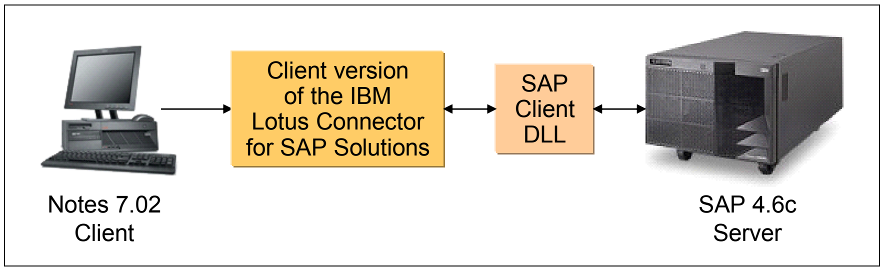


Figure 1-1 The major pieces of Lotus Notes access for SAP solutions

The Lotus Notes access for SAP solutions feature leverages a subset of the technology delivered with the Lotus Connector for SAP Solutions software product. The Lotus Notes access for SAP solutions feature is a Lotus Notes client-based solution. You do not need the Lotus Connector for SAP Solutions product in order to use the Lotus Notes access for SAP solutions feature. The Lotus Connector for SAP Solutions product is an extra-cost option that enables you to centralize connectivity to SAP applications on a Lotus Domino server.

The Lotus Notes access for SAP solutions feature comes with a special client-side version of the SAP Connector, which:

- Handles authentication with SAP.
- Provides access to Remote Function Calls (RFCs), bUsiness Application Programming Interfaces (BAPIs), and transactions. See the glossary in Appendix A, "Useful SAP information" on page 109 for a complete description of these terms.

Both the client-based feature and the server-based product use the Lotus Connector LSX to make remote function calls using a Lotus Connector designed for connectivity to SAP applications. The server-based product also includes additional features that allow this Lotus Connector to be used with IBM Lotus Enterprise Integrator® software or the Lotus Domino Enterprise Connection Services feature of the Lotus Domino server.

Both the client-based feature and the server-based product include features that allow SAP workflow items to be surfaced in their Lotus Notes Inbox. The SAP workflow item feature in the Lotus Notes Connector for SAP Solutions product uses a mix of Lotus Domino server-side integration and Lotus Notes client-side integration with SAP software. The workflow integration capability of the Lotus Notes access for SAP solutions feature is strictly Lotus Notes client-side integration. The Lotus Connector for SAP Solutions product is supported for use with Lotus Notes versions 6.0.x, 6.5.x, and 7.0, while the Lotus Notes access for SAP solutions feature is available for Lotus Notes 7.0.1 and later.

IBM Lotus Enterprise Integrator for Domino, also called Lotus Enterprise Integrator, enables you to develop enterprise integration solutions without needing to program in LotusScript. When used with Lotus Connector for SAP Solutions, Lotus Enterprise Integrator provides industrial strength, real-time data access to, as well as batch-mode data transfer and synchronization with, SAP applications. The business benefits of Lotus Enterprise Integrator include reducing application development time and improved administration of integrated solutions.

The Lotus Enterprise Integrator administrator database also provides scheduling, monitoring, logging, and reporting capabilities. Connectivity to SAP solutions is centralized on the Lotus Enterprise Integration server, providing a single point of access. Centralizing SAP solution connectivity on the server is required to integrate SAP information and business processes into Web-enabled Lotus Domino applications.

For detailed information about all Lotus Notes and Domino integration technologies from IBM, read the *IBM Lotus Notes and Lotus Domino integration technologies for SAP software* white paper at:

ftp://ftp.software.ibm.com/software/lotus/lotusweb/product/SAPconnector/SAP_integration_white_paper.pdf

To learn more about IBM Lotus Connector for SAP Solutions software, visit:

http://ibm.com/lotus/sapconnector

To learn more about the Lotus Enterprise Integrator, visit:

http://ibm.com/lotus/lei

To learn more about the Lotus Notes access for SAP solutions feature of Lotus Notes, version 7.0.1 software and later, visit:

http://ibm.com/lotus/notesforsap

1.3 Audience and goals of this paper

This IBM Redpaper provides information for various types of professionals. Executives, line of business managers, and users will benefit from the overview of capabilities described in Chapter 2, “Architecture and features” on page 5. Application developers can use the business scenario described in Chapter 3, “Customization” on page 35, Chapter 4, “Extending” on page 63 as a case study and guide for customizing Lotus Notes access for SAP solutions for their organizations, and Chapter 5, “Troubleshooting” on page 91 for troubleshooting problems that might come up.

After reading this paper, we hope you will:

- Understand the features and functionality of Lotus Notes access for SAP solutions.
- Be able to customize and extend the functionality that comes with the set of features.
- Be able to use your knowledge of LotusScript to access data stored in your SAP system.
- Have the tools to explore and troubleshoot SAP capabilities on your own, which in turn will enable you to communicate better with SAP experts in your company. You will know how to ask for what you need, which is the first step!

2

Architecture and features

In subsequent chapters, we provide technical information and show you how to not only customize the Lotus Notes access for SAP solutions templates, but actually create your own template. This chapter gets you started with Lotus Notes for SAP solutions, highlighting:

- Architecture
- Features
- Installation basics

2.1 Architecture

An important thing to keep in mind about the Notes access for SAP solutions feature is that it is runs entirely on the client; no server components are required.

Figure 2-1 shows the architecture of the Notes access for SAP solutions feature.

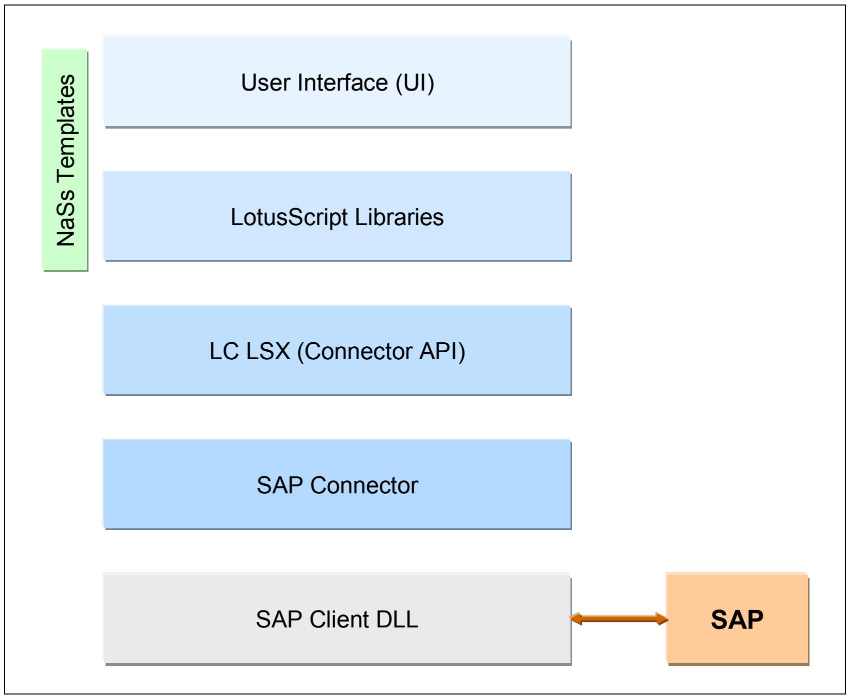


Figure 2-1 Notes access for SAP solutions building blocks

An SAP system contains thousands of functions referred to as BAPIs and RFCs. For typical Lotus Notes users, these BAPIs and RFCs can seem very complex. In this paper, we explore several BAPIs and RFCs in greater detail in later chapters. However, at their most basic level, these functions generate outputs as a result of submitting required preset inputs. The Lotus Notes access for SAP solutions script libraries (back end) and the Lotus Notes access for SAP solutions forms (front end) are specifically constructed to make passing these inputs and outputs between Notes and SAP a fairly rudimentary operation.

The Lotus Notes access for SAP solutions architecture can be divided in two major sections: the front end (Notes design elements) and the back end (Lotus Notes access for SAP solutions script libraries).

2.1.1 Lotus Notes access for SAP solutions front end

The Lotus Notes access for SAP solutions front end, as with all Notes applications, can be divided into the presentation layer, what we see in Notes, and the underlying design elements, forms, subforms, views, and so on. The Lotus Notes access for SAP solutions front end is nothing more than a Notes application. The Notes client UI presents intuitive options for users to request and view data. Beneath the UI, we find fields on the forms that format the data going to and from SAP in a usable manner. (See 4.5, “Step 4: Passing SAP data to the UI” on page 86 for more details.)

After the data is properly formatted, it is stored in a temporary Notes document. Using a Notes document to store all the data allows us to have a familiar container from which we can access all our data. We do not have to know the SAP details; they are hidden in the script libraries of the Lotus Notes access for SAP solutions back end. We only need to know how to access Notes fields on a form to retrieve SAP data for display in the Notes client UI.

2.1.2 Lotus Notes access for SAP solutions back end

The Lotus Notes access for SAP solutions back end consists of a group of script libraries that do the work needed to transform the Notes client UI inputs into commands that can run the SAP BAPIs and RFCs. The functions within these script libraries take care of the SAP details. These functions are then used in Notes agents to take the inputs from the Notes temporary storage document, use these inputs to execute SAP functions, and return output and table results sets from SAP back to the temporary Notes storage document. These values are stored as arrays in the Notes document and can easily be accessed by the Notes UI.

At the heart of the Lotus Notes access for SAP solutions feature are the LC LSX classes that have been available since Lotus Domino version 4.6. These classes allow Notes application developers to access back-end relational databases through LotusScript. At the center of the Lotus Notes access for SAP solutions script libraries are calls to the LC LSX classes. Because of the LC LSX classes, all that was needed to access SAP was to develop an SAP connection file that allows the LC LSX classes to talk to the SAP client software. Beginning with Lotus Notes 7.0.1, out-of-the-box installations include the SAP connector needed to allow the LC LSX classes to access an SAP system. Finally, the back-end (SAP) client software is required. The SAP client file (librfc32.dll) must be obtained directly from SAP.

To summarize, the Lotus Notes access for SAP solutions architects took the LC LSX classes that have been used for years to access back-end database systems and built a set of script libraries and Notes UI design elements to ease information swapping between Notes and SAP. They used the Notes document object as a conduit to pass information to and from the Notes UI to the SAP back end. Other than including the SAP connector in the Lotus Notes 7.0.1 client, no new client functionality was added. The Lotus Notes access for SAP solutions feature is based on existing Domino standards. The division of labor between the Notes UI and back-end LotusScript libraries coupled with a Notes document as a container for the inputs and outputs make the Lotus Notes access for SAP solutions architecture a relatively easy framework from which to integrate Notes and SAP data.

2.1.3 Templates

The Notes access for SAP solutions capabilities are delivered in three templates, as shown in Table 2-1 on page 8. The personal address book and mail templates are enhanced versions of the standard Notes 7.0.2 personal address book and mail templates. The Human Resources Self Service template is new.

In addition, Lotus Notes access for SAP solutions comes with a handy Install database that:

- Provides a list of all design elements (new and modified)
- Has a template builder that automates the process of creating the templates you will deploy

See 2.3, “Installing Notes access for SAP solutions” on page 30 for a description of the Install database.

Table 2-1 Templates and databases provided for Lotus Notes access for SAP solutions

Template or database	**Feature set**
Personal name and address book (required[a])	Connection and setup Contact management
Mail	Time reporting Vacation/leave request Workflow integration Meeting scheduling Report generation
Human Resources Self Service	Employee/manager self service
Install database	Template builder and list of design elements for administrators and application developers
Install Help database	Help file for Install database
End-user Help database	Help file for users

a. The personal name and address book is required for any deployment of Lotus Notes access for SAP solutions because it contains the information for connecting the user to SAP.

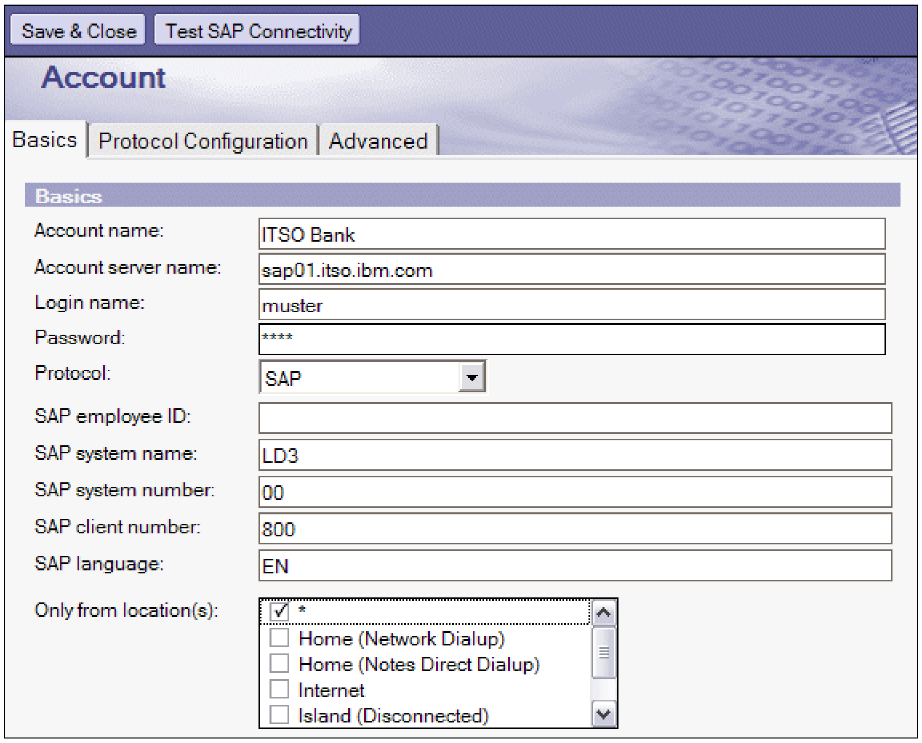
2.2 Features

This section provides details about the functionality of the seven Lotus Notes access for SAP solutions capabilities (let's call them features for simplicity) in Lotus Notes 7.0.2, plus brief information about connecting to SAP. For complete details, refer to the Help file as well.

2.2.1 Connecting to SAP

In the revised template for the local name and address book, the Account form now includes the ability to choose SAP as a protocol. This is what you use to configure your connection so that you never have to log in to SAP while using the Notes access for SAP solutions features.

Figure 2-2 shows the Account document. You enter your SAP login name and password and your SAP employee ID; your administrator will provide you with the rest of the information to enter.



Account name:	ITSO Bank
Account server name:	sap01.itso.ibm.com
Login name:	muster
Password:	****
Protocol:	SAP
SAP employee ID:	
SAP system name:	LD3
SAP system number:	00
SAP client number:	800
SAP language:	EN
Only from location(s):	✓ *
	Home (Network Dialup)
	Home (Notes Direct Dialup)
	Internet
	Island (Disconnected)

Figure 2-2 Account document in personal name and address book

Click the **Test SAP Connectivity** button to confirm that your connection works. If all is OK, you will see the Login successful window (Figure 2-3).

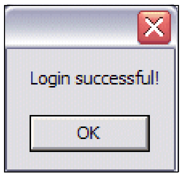

Login successful!

Figure 2-3 Successful connection to SAP

This is followed by an output of system administration information such as the SAP version shown in Figure 2-4.

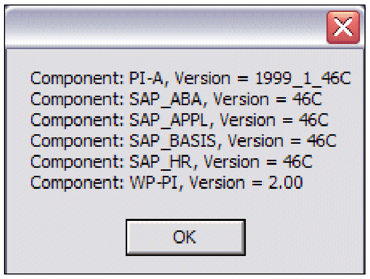

```
Component: PI-A, Version = 1999_1_46C
Component: SAP_ABA, Version = 46C
Component: SAP_APPL, Version = 46C
Component: SAP_BASIS, Version = 46C
Component: SAP_HR, Version = 46C
Component: WP-PI, Version = 2.00
```

Figure 2-4 SAP system data

If you are unable to connect, see Chapter 5, “Troubleshooting” on page 91 for details.

You can connect to only one SAP server at a time, so if you need to connect to multiple servers, create an Account document for each server, and then associate each Account document with a Notes location to allow easy switching between SAP servers.

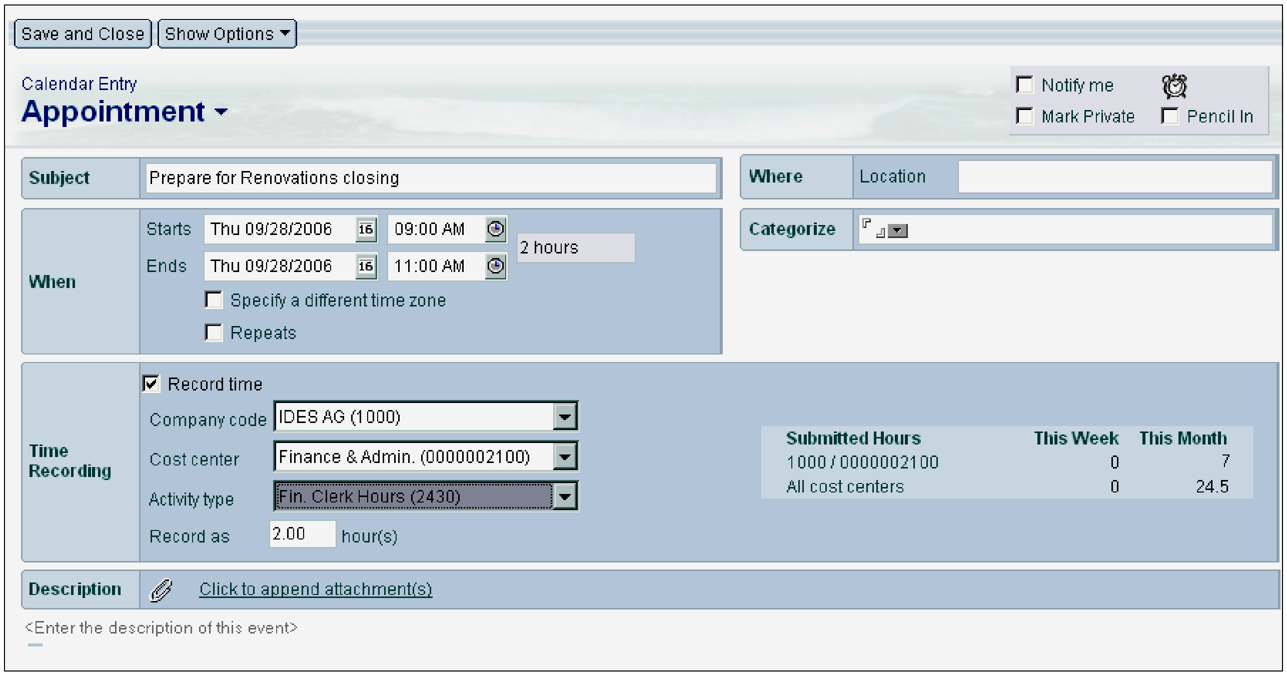
And here is a tip: You can easily get back to your SAP Account document by selecting **Tools** → **Edit SAP Account** in any of the databases enabled for use with SAP.[1]

2.2.2 Time reporting

Time reporting enables you to record, monitor, and report billable time to SAP, right from the familiar environment of your Notes calendar.

Recording time in meetings and appointments

In the Meeting and Appointment calendar forms, you will see a new section called “Time Recording,” as shown in Figure 2-5.



Subject: Prepare for Renovations closing

Where: Location

When: Starts Thu 09/28/2006 09:00 AM; Ends Thu 09/28/2006 11:00 AM; 2 hours

Specify a different time zone

Repeats

Categorize

Time Recording:

☑ Record time

Company code: IDES AG (1000)

Cost center: Finance & Admin. (0000002100)

Activity type: Fin. Clerk Hours (2430)

Record as: 2.00 hour(s)

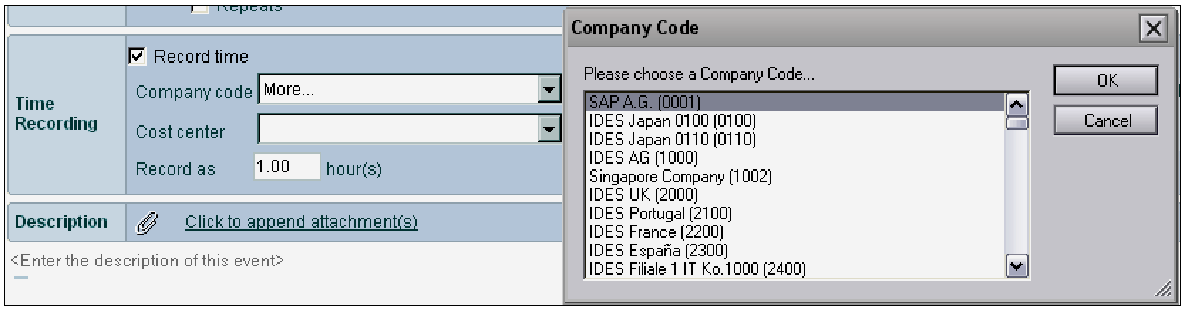
Submitted Hours	This Week	This Month
1000 / 0000002100	0	7
All cost centers	0	24.5

<Enter the description of this event>

Figure 2-5 Time Recording section in Appointment form

[1] © SAP AG 2006. All rights reserved.

You enter the company code, cost center, and activity type against which you want to report the time. The first time you use the feature, these drop-down lists will be blank; just click **More** to open the list of choices (Figure 2-6).



SAP A.G. (0001)
IDES Japan 0100 (0100)
IDES Japan 0110 (0110)
IDES AG (1000)
Singapore Company (1002)
IDES UK (2000)
IDES Portugal (2100)
IDES France (2200)
IDES España (2300)
IDES Filiale 1 IT Ko.1000 (2400)

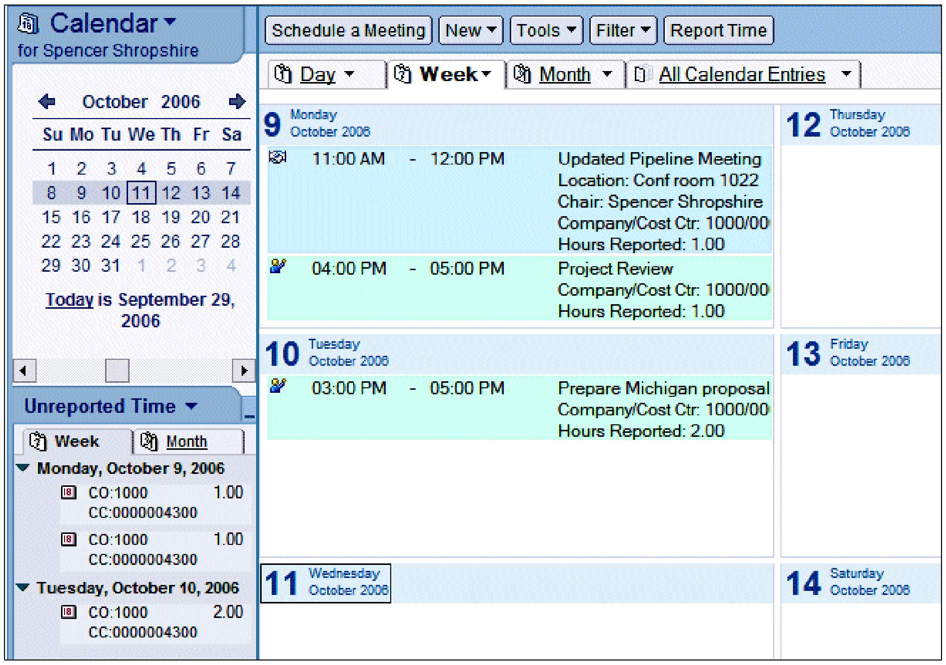
Figure 2-6 List of Company Codes, pulled live from the SAP server

Each time you choose a new code or type, it is added to the drop-down list, making it quick and easy to select the codes you use most often. These lists are stored locally, which means that you can record time when working offline.

If you want, you can adjust the amount of time to report by entering it in the “Record as” box. By default, it matches the meeting length.

Notice that after you enter the company code and cost center, the summary section on the right updates to show time already submitted in SAP for the company code and cost center you entered.

After you save the meeting or appointment, the entry in the Calendar view displays the company code and cost center and the number of hours recorded. And the Unreported Time mini-view shows a running tally of time you have recorded, but not reported, for the time period displayed in the view (Figure 2-7).



9 Monday October 2006
11:00 AM - 12:00 PM Updated Pipeline Meeting
Location: Conf room 1022
Chair: Spencer Shropshire
Company/Cost Ctr: 1000/00
Hours Reported: 1.00
04:00 PM - 05:00 PM Project Review
Company/Cost Ctr: 1000/00
Hours Reported: 1.00

10 Tuesday October 2006
03:00 PM - 05:00 PM Prepare Michigan proposal
Company/Cost Ctr: 1000/00
Hours Reported: 2.00

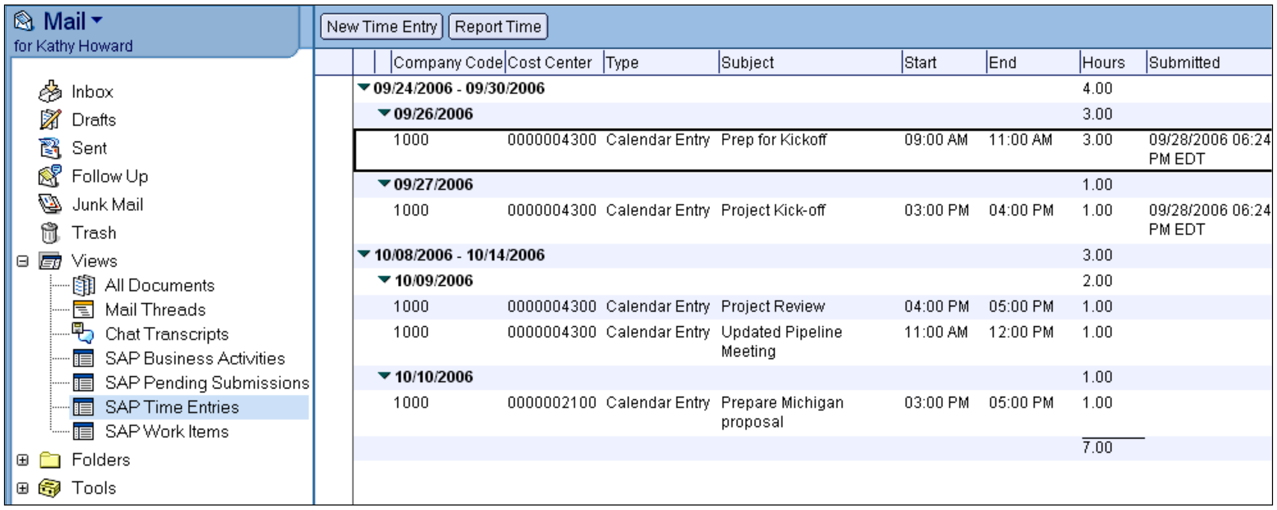
11 Wednesday October 2006
12 Thursday October 2006
13 Friday October 2006
14 Saturday October 2006

Monday, October 9, 2006
CO:1000 1.00
CC:0000004300
CO:1000 1.00
CC:0000004300
Tuesday, October 10, 2006
CO:1000 2.00
CC:0000004300

Figure 2-7 Entries in Calendar view and mini-view showing recorded time

Viewing unreported and reported time

The SAP Time Entries view shows all time you have recorded and reported. It is a great way to get the whole picture of time you have recorded, before you start your time report (Figure 2-8).



Company Code	Cost Center	Type	Subject	Start	End	Hours	Submitted
09/24/2006 - 09/30/2006						4.00	
09/26/2006						3.00	
1000	0000004300	Calendar Entry	Prep for Kickoff	09:00 AM	11:00 AM	3.00	09/28/2006 06:24 PM EDT
09/27/2006						1.00	
1000	0000004300	Calendar Entry	Project Kick-off	03:00 PM	04:00 PM	1.00	09/28/2006 06:24 PM EDT
10/08/2006 - 10/14/2006						3.00	
10/09/2006						2.00	
1000	0000004300	Calendar Entry	Project Review	04:00 PM	05:00 PM	1.00	
1000	0000004300	Calendar Entry	Updated Pipeline Meeting	11:00 AM	12:00 PM	1.00	
10/10/2006						1.00	
1000	0000002100	Calendar Entry	Prepare Michigan proposal	03:00 PM	05:00 PM	1.00	
						7.00	

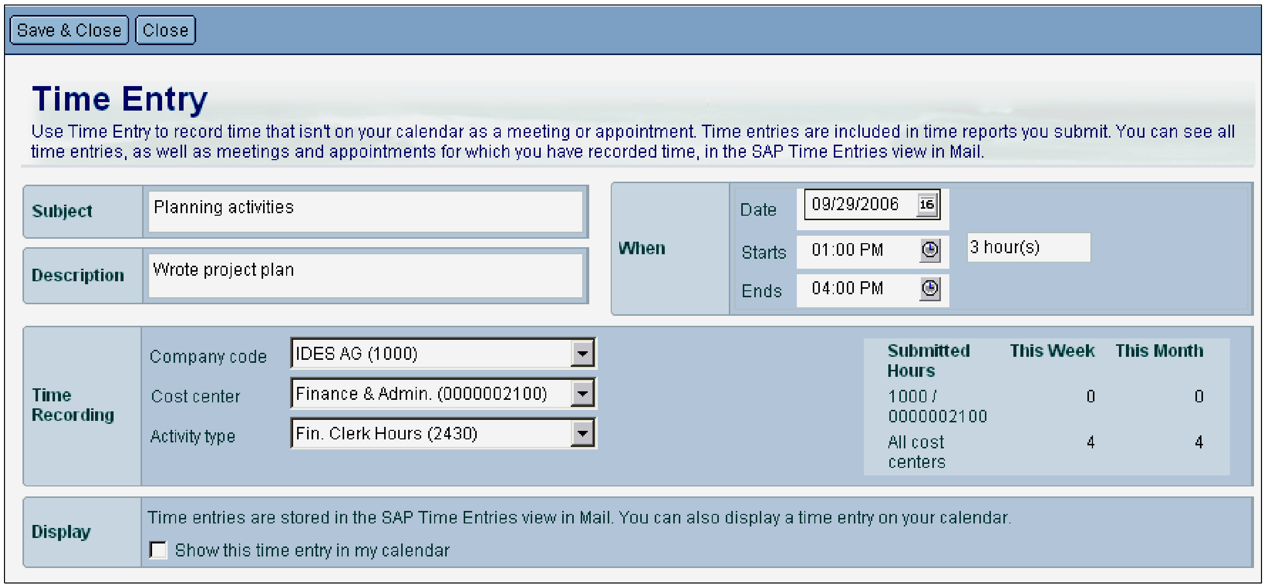
Figure 2-8 SAP Time Entries view showing both reported and unreported time

From the SAP Time Entries view, if you discover gaps in the time you want to report, you can add more time to by creating a time entry. And you can begin your time report right from this view as well.

Using the Time Entry form

The Time Entry form, available in both the Calendar view and the SAP Time Entries view, lets you record additional time that you might not want to have on your calendar, such as preparation time for a meeting.

This form contains just the fields for recording time and includes an option for showing the entry on the calendar (Figure 2-9).



Time Entry

Use Time Entry to record time that isn't on your calendar as a meeting or appointment. Time entries are included in time reports you submit. You can see all time entries, as well as meetings and appointments for which you have recorded time, in the SAP Time Entries view in Mail.

Subject: Planning activities

Description: Wrote project plan

When — Date: 09/29/2006; Starts: 01:00 PM; Ends: 04:00 PM; 3 hour(s)

Time Recording — Company code: IDES AG (1000); Cost center: Finance & Admin. (0000002100); Activity type: Fin. Clerk Hours (2430)

Submitted Hours	This Week	This Month
1000 / 0000002100	0	0
All cost centers	4	4

Display — Time entries are stored in the SAP Time Entries view in Mail. You can also display a time entry on your calendar.

☐ Show this time entry in my calendar

Figure 2-9 Time Entry form

Reporting time

When it is time to send in your time report, use the Report Time wizard to gather up all the hours you recorded.

Click the **Report Time** button on the action bar, as displayed in Figure 2-10, to start the 3-step process.

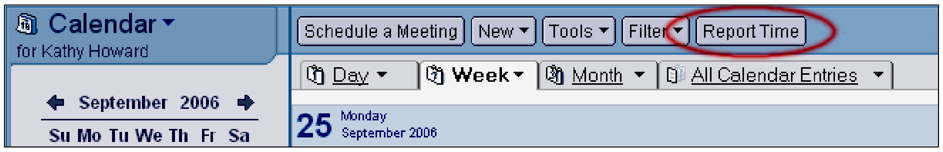
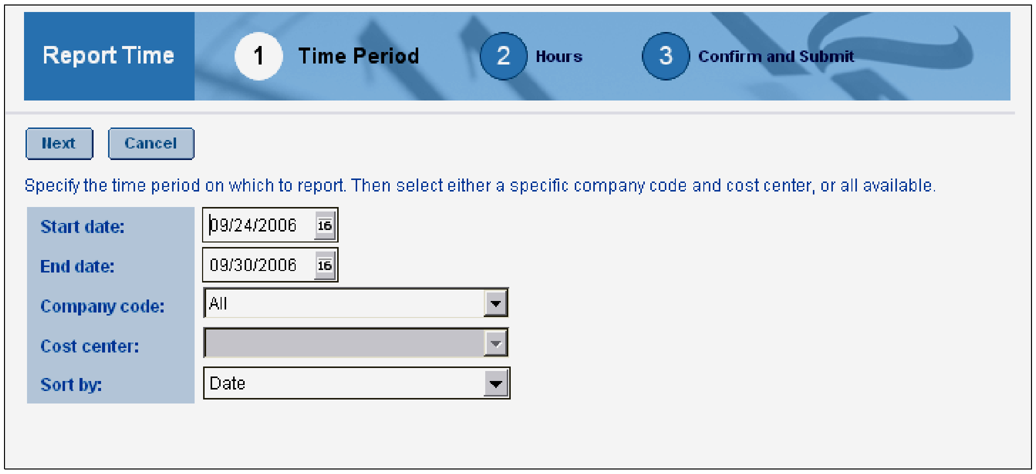


Figure 2-10 Report Time button

Step 1 of the wizard lets you choose the time period for the report and on which company codes and cost centers to report. "All" is the default (Figure 2-11).

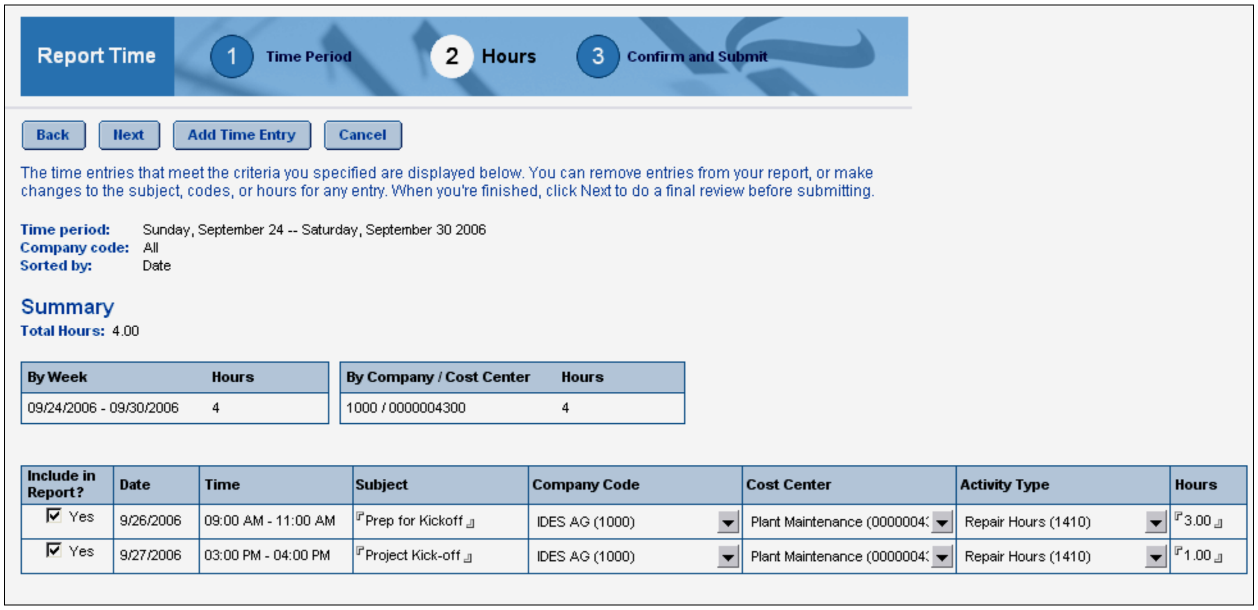


Specify the time period on which to report. Then select either a specific company code and cost center, or all available.

Start date:	09/24/2006
End date:	09/30/2006
Company code:	All
Cost center:	
Sort by:	Date

Figure 2-11 Step 1 of Report Time wizard

Step 2 of the wizard (Figure 2-12) lists the hours recorded for the time period chosen previously in step 1.



The time entries that meet the criteria you specified are displayed below. You can remove entries from your report, or make changes to the subject, codes, or hours for any entry. When you're finished, click Next to do a final review before submitting.

Time period: Sunday, September 24 -- Saturday, September 30 2006
Company code: All
Sorted by: Date

Summary
Total Hours: 4.00

By Week	Hours
09/24/2006 - 09/30/2006	4

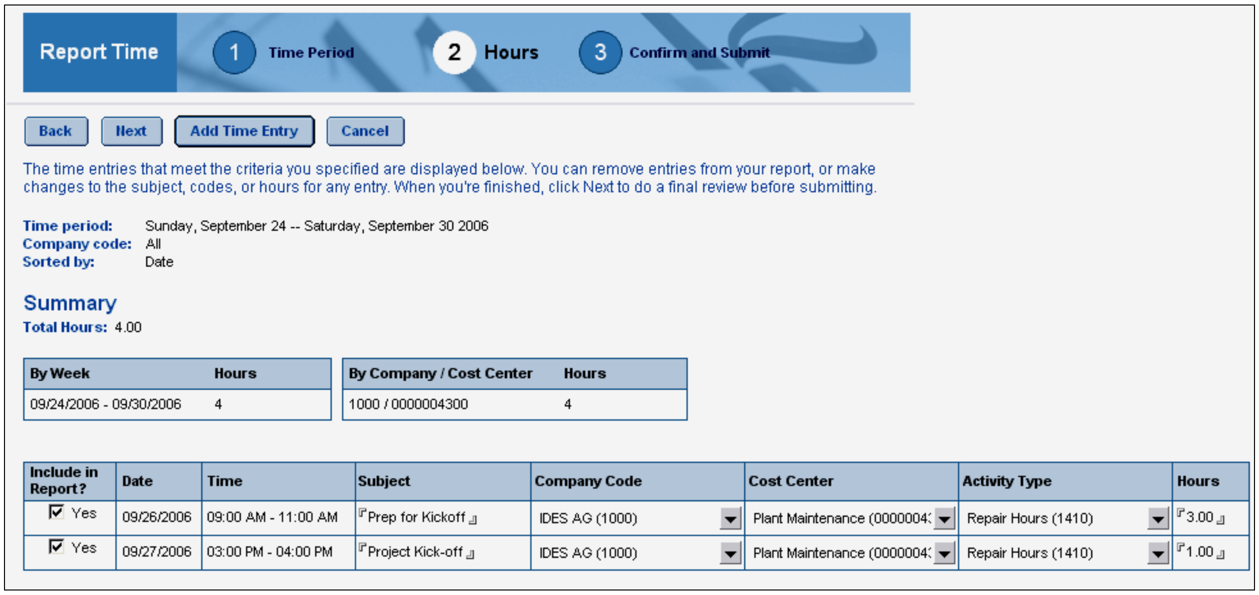
By Company / Cost Center	Hours
1000 / 0000004300	4

Include in Report?	Date	Time	Subject	Company Code	Cost Center	Activity Type	Hours
✓ Yes	9/26/2006	09:00 AM - 11:00 AM	Prep for Kickoff	IDES AG (1000)	Plant Maintenance (0000004:	Repair Hours (1410)	3.00
✓ Yes	9/27/2006	03:00 PM - 04:00 PM	Project Kick-off	IDES AG (1000)	Plant Maintenance (0000004:	Repair Hours (1410)	1.00

Figure 2-12 Step 2 of Report Time wizard

You can adjust the subject, time, and codes for the entries listed. (The changes you make here are saved back to the original calendar entries after the time has been successfully submitted to SAP.) You can add new time entries from here as well.

Step 3 of the wizard lets you confirm what will be reported (Figure 2-13). Click **Submit** to send the report to SAP. You will receive a confirmation e-mail that the time was successfully submitted.



The time entries that meet the criteria you specified are displayed below. You can remove entries from your report, or make changes to the subject, codes, or hours for any entry. When you're finished, click Next to do a final review before submitting.

Time period: Sunday, September 24 -- Saturday, September 30 2006
Company code: All
Sorted by: Date

Summary
Total Hours: 4.00

By Week	Hours
09/24/2006 - 09/30/2006	4

By Company / Cost Center	Hours
1000 / 0000004300	4

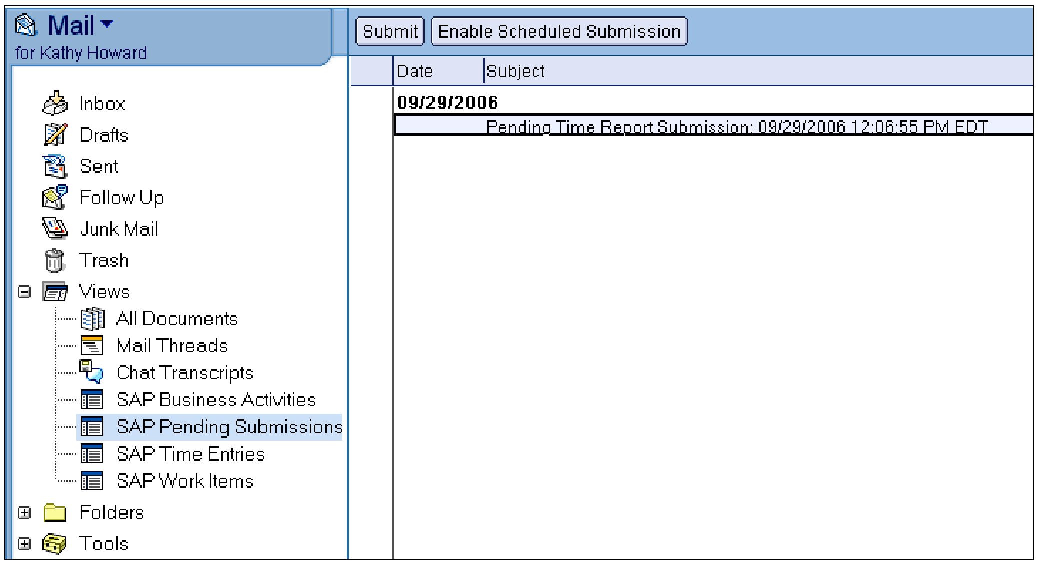
Include in Report?	Date	Time	Subject	Company Code	Cost Center	Activity Type	Hours
✓ Yes	09/26/2006	09:00 AM - 11:00 AM	Prep for Kickoff	IDES AG (1000)	Plant Maintenance (0000004:	Repair Hours (1410)	3.00
✓ Yes	09/27/2006	03:00 PM - 04:00 PM	Project Kick-off	IDES AG (1000)	Plant Maintenance (0000004:	Repair Hours (1410)	1.00

Figure 2-13 Step 3 of the Report Time wizard

Need to work offline? No problem—you can run the Report Time wizard when you are offline. After clicking Submit, the time report is saved in the Pending Submissions view so that you can send it later.

Pending Submissions view

The Pending Submissions view shows you the time reports that have not yet been submitted (Figure 2-14).



Date	Subject
09/29/2006	
	Pending Time Report Submission: 09/29/2006 12:06:55 PM EDT

Figure 2-14 SAP Pending Submissions view

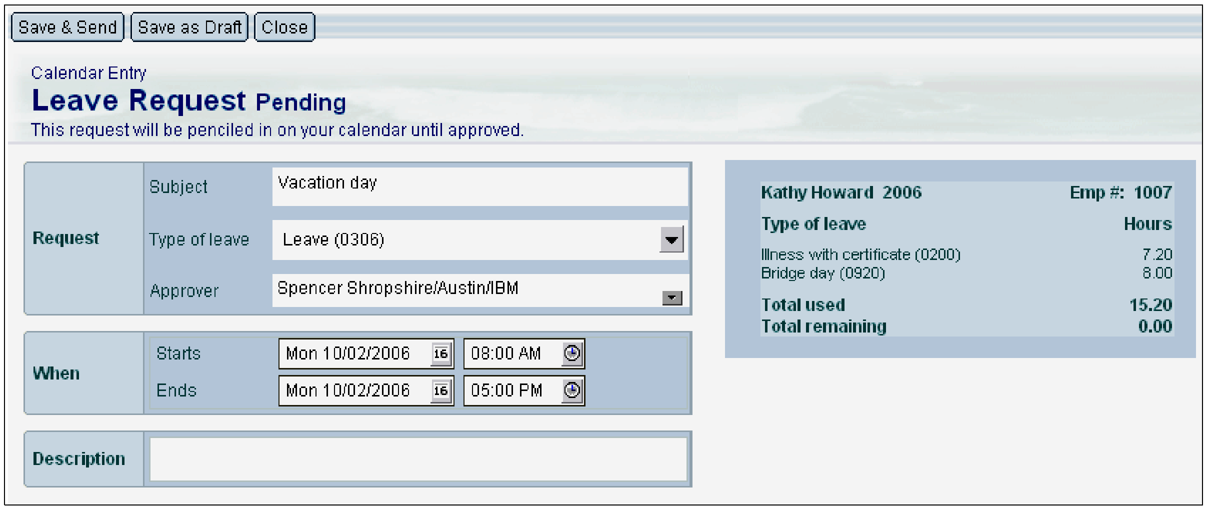
From this view, you can submit the report manually, or enable a scheduled agent that will automatically submit any pending reports.

Note: Clicking **Enable Scheduled Submission** also turns on the "Enable scheduled local agents" preference if necessary.

2.2.3 Vacation/leave request

Vacation/leave request lets you create requests for vacation or other types of leave by creating an entry in your Notes calendar and then sending it for approval. (It works much like scheduling a meeting.) You create a leave request calendar entry and send the request to an approver (probably your manager), who receives it in an e-mail. When your manager accepts the request, it is booked to SAP and you receive an acceptance e-mail.

To create a leave request, select **New** → **Leave Request** in your calendar. Notice that the summary area on the right shows leave you submitted for the year (Figure 2-15).



Calendar Entry
Leave Request Pending
This request will be penciled in on your calendar until approved.

Request	Subject	Vacation day
	Type of leave	Leave (0306)
	Approver	Spencer Shropshire/Austin/IBM
When	Starts	Mon 10/02/2006 08:00 AM
	Ends	Mon 10/02/2006 05:00 PM
Description		

Kathy Howard 2006	Emp #: 1007
Type of leave	Hours
Illness with certificate (0200)	7.20
Bridge day (0920)	8.00
Total used	15.20
Total remaining	0.00

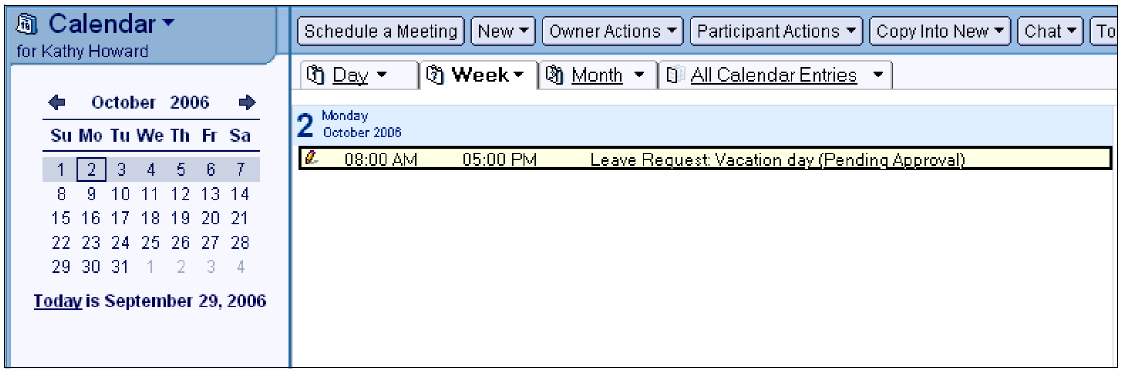
Figure 2-15 Creating a Leave Request

Enter a subject and then choose the type of leave from the list. Similar to the codes in time reporting, the list is initially blank but gets populated with the leave types you use.

Enter your manager's name. Then, specify the date and time for the leave. By default, the leave is one day. You can request part of a day by entering a start and end time. To request multiple full days, choose a different end date. (Multiple day requests must be for full days.)

Then click **Save & Send** to send it off for approval.

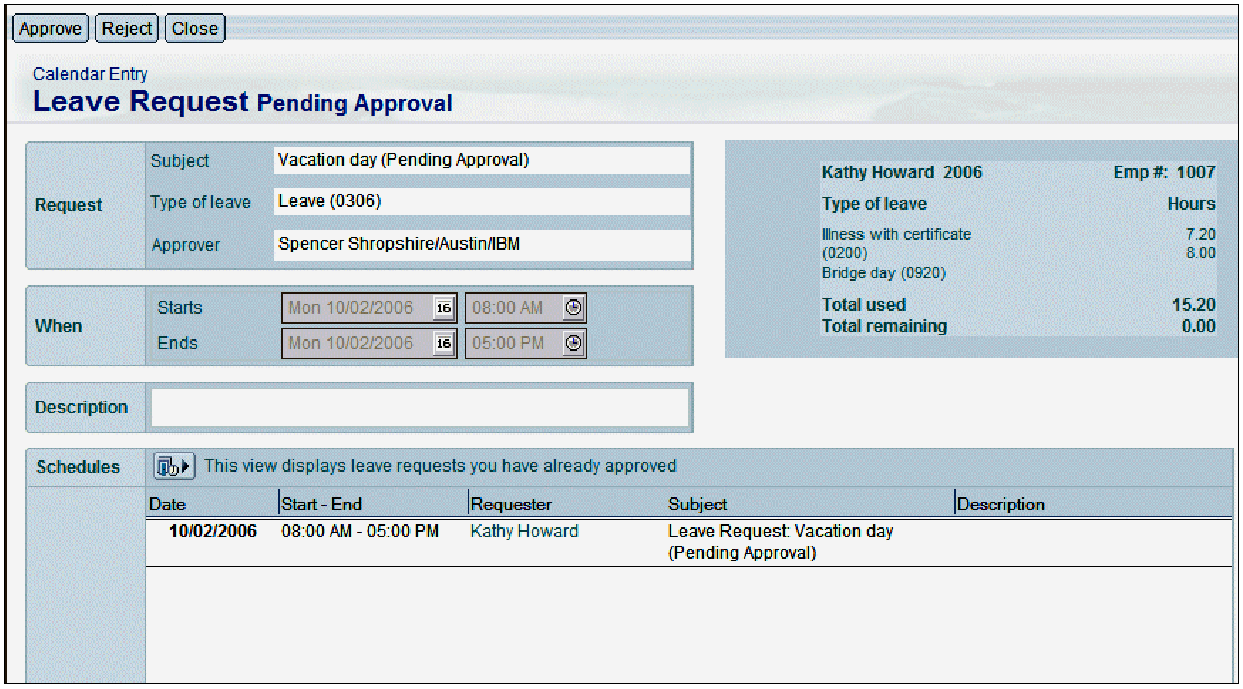
The request will appear on your calendar as penciled in until you receive approval, as shown in Figure 2-16.



08:00 AM 05:00 PM Leave Request: Vacation day (Pending Approval)

Figure 2-16 Leave Request awaiting approval

The manager receives an e-mail with the request. Managers who receive leave requests can use the Schedules section to check who else will be out. The Schedules section displays leave requests the manager has approved (Figure 2-17).



Calendar Entry

Leave Request Pending Approval

Request	Subject	Vacation day (Pending Approval)
	Type of leave	Leave (0306)
	Approver	Spencer Shropshire/Austin/IBM
When	Starts	Mon 10/02/2006 08:00 AM
	Ends	Mon 10/02/2006 05:00 PM
Description		

Kathy Howard 2006	Emp #: 1007
Type of leave	Hours
Illness with certificate (0200)	7.20
Bridge day (0920)	8.00
Total used	15.20
Total remaining	0.00

Schedules — This view displays leave requests you have already approved

Date	Start - End	Requester	Subject	Description
10/02/2006	08:00 AM - 05:00 PM	Kathy Howard	Leave Request: Vacation day (Pending Approval)	

Figure 2-17 Manager receives the Leave Request and can click Approve or Reject

When the manager clicks **Approve**, the request is submitted to SAP immediately. When the confirmation comes back from SAP, the Approval Comments window opens, allowing the manager to include comments.

When you open the acceptance e-mail, the request changes from being penciled in to an appointment (for single-day leave) or to an all-day event (for multi-day leave), as shown in Figure 2-18.

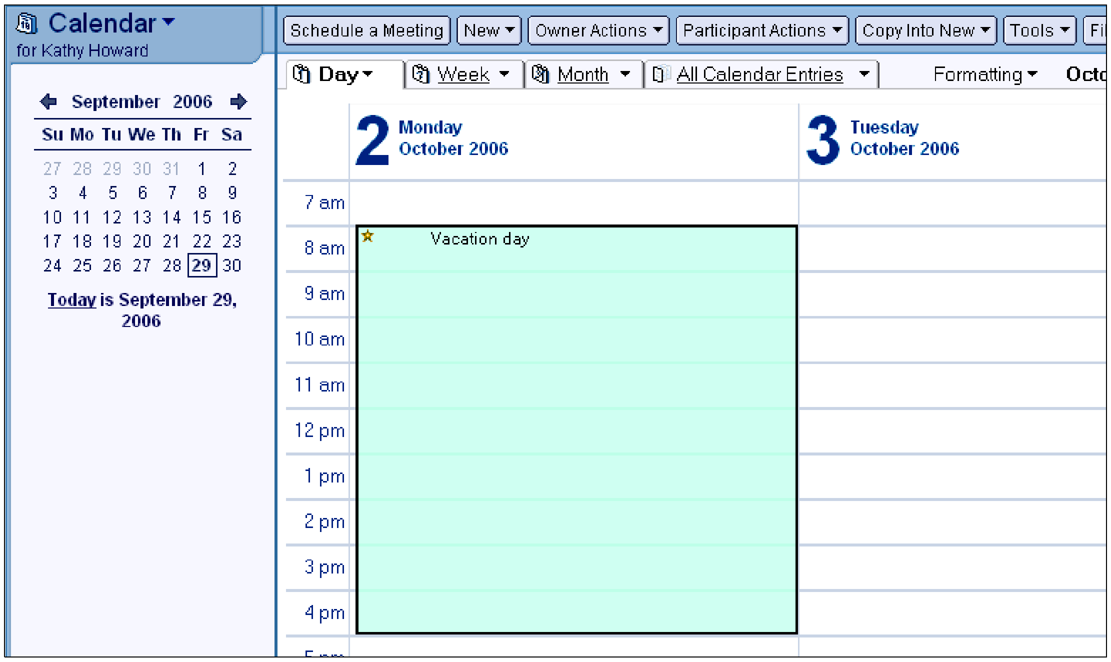
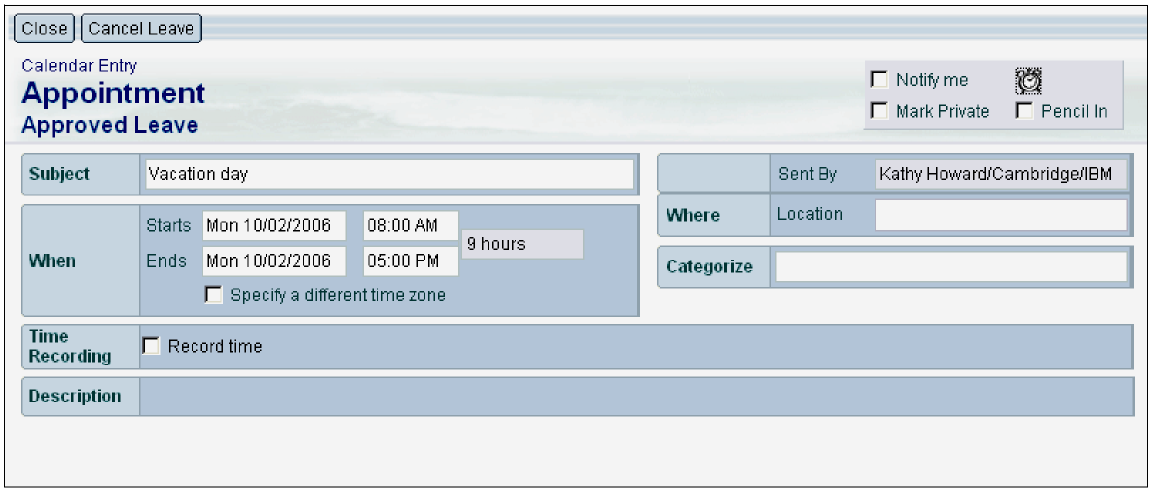


Figure 2-18 Accepted leave appears in the Calendar view just like other entries

When you open the entry, you can make changes, such as cancelling the leave or editing and resubmitting it (Figure 2-19).

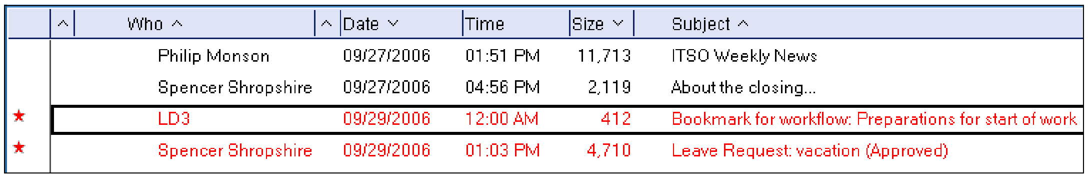


Subject	Vacation day
When	Starts Mon 10/02/2006 08:00 AM; Ends Mon 10/02/2006 05:00 PM; 9 hours; Specify a different time zone
Time Recording	Record time
Description	
Sent By	Kathy Howard/Cambridge/IBM
Where	Location
Categorize	

Figure 2-19 An Approved Leave request

2.2.4 Workflow integration

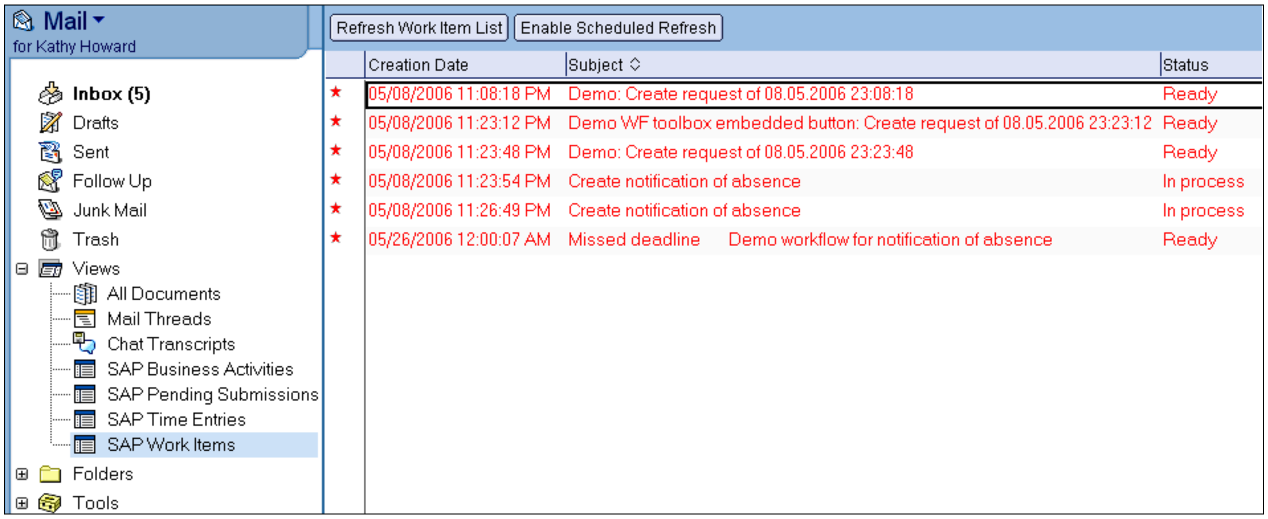
Workflow integration solves a common problem for people who spend most of their time in Notes, but need to process workflow items in SAP. It brings your SAP work item list into your Inbox so that you do not have to go into SAP to check it (Figure 2-20).



	Who	Date	Time	Size	Subject
	Philip Monson	09/27/2006	01:51 PM	11,713	ITSO Weekly News
	Spencer Shropshire	09/27/2006	04:56 PM	2,119	About the closing...
★	LD3	09/29/2006	12:00 AM	412	Bookmark for workflow: Preparations for start of work
★	Spencer Shropshire	09/29/2006	01:03 PM	4,710	Leave Request: vacation (Approved)

Figure 2-20 SAP work item in Inbox

Work items are also shown in the SAP Work Items view, as displayed in Figure 2-21, where you can enable an agent to periodically refresh the list. This SAP Work Items view shows the status and priority for each item.



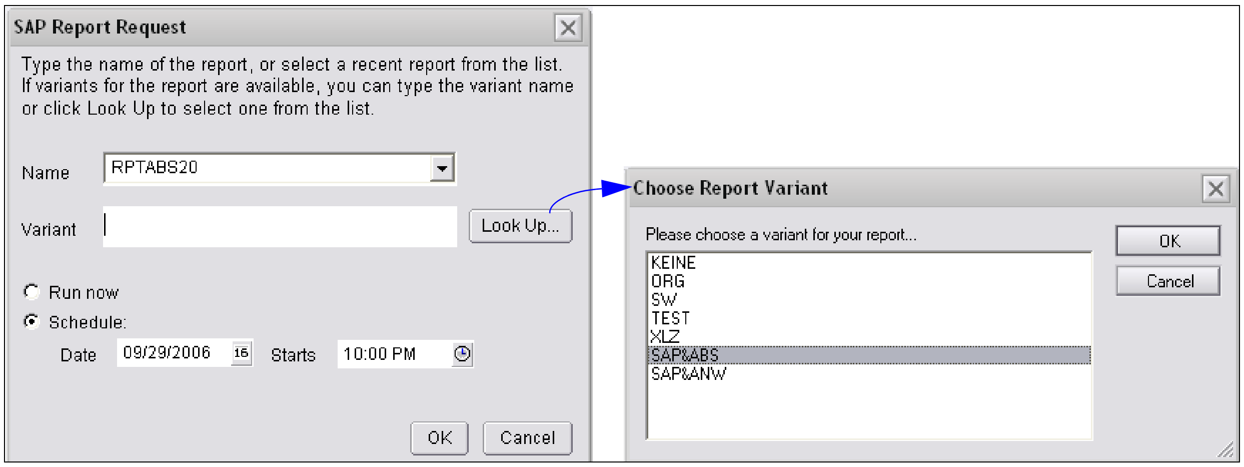
	Creation Date	Subject	Status
★	05/08/2006 11:08:18 PM	Demo: Create request of 08.05.2006 23:08:18	Ready
★	05/08/2006 11:23:12 PM	Demo WF toolbox embedded button: Create request of 08.05.2006 23:23:12	Ready
★	05/08/2006 11:23:48 PM	Demo: Create request of 08.05.2006 23:23:48	Ready
★	05/08/2006 11:23:54 PM	Create notification of absence	In process
★	05/08/2006 11:26:49 PM	Create notification of absence	In process
★	05/26/2006 12:00:07 AM	Missed deadline Demo workflow for notification of absence	Ready

Figure 2-21 SAP Work Items view in Mail

2.2.5 Report generation

Report generation is a simple yet powerful time-saving feature. Chances are, you run the same reports over and over. Report generation saves you a trip to the SAP GUI by providing a way for you to request a report right from within your Notes mail database.

Select **Request SAP Report** from the Tools menu in Mail or Calendar to open the SAP Report Request window (Figure 2-22).


SAP Report Request

Type the name of the report, or select a recent report from the list. If variants for the report are available, you can type the variant name or click Look Up to select one from the list.

Name: RPTABS20

Variant:

Run now

Schedule: Date 09/29/2006 Starts 10:00 PM

Choose Report Variant

Please choose a variant for your report...

KEINE
ORG
SW
TEST
XLZ
SAP&ABS
SAP&ANW

Figure 2-22 Report generation

First, you enter the report name. You can type in the report name or click the drop-down arrow to choose from reports you previously requested. The first time you use the feature, this drop-down list will be blank, but each time you request a report, it will be added to the list.

If the report you choose has variants associated with it, the Look Up button will appear and you can choose a variant from the list.

If your SAP server supports the scheduling of reports, you will see options for specifying a date and time to run the report. If these options do not appear, or if you select **Run now**, the request will be added to the report queue on the SAP server when you click **OK**.

You retrieve the report as you normally would. (The type of report determines how it is delivered, for example, in an e-mail or in the SAP GUI.)

2.2.6 Meeting scheduling

If you use the SAP Customer Relationship Management (CRM) system, you probably have business activities scheduled there, as well as meetings in your Notes calendar. Meeting scheduling pulls your meetings and other business activities from the SAP system into your Notes calendar, giving you just one place to monitor both.

To display SAP business activities on your Notes calendar, select **Tools** → **Refresh Business Activities from SAP (**Figure 2-23).

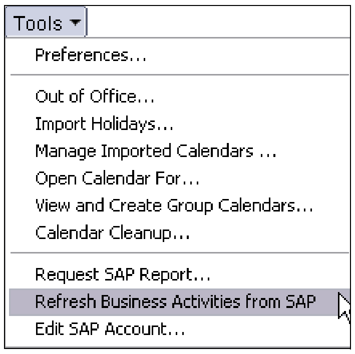


Figure 2-23 Adding or refreshing SAP business activities

In the Calendar view, shown in Figure 2-24, a special icon denotes which entries are from the SAP system.

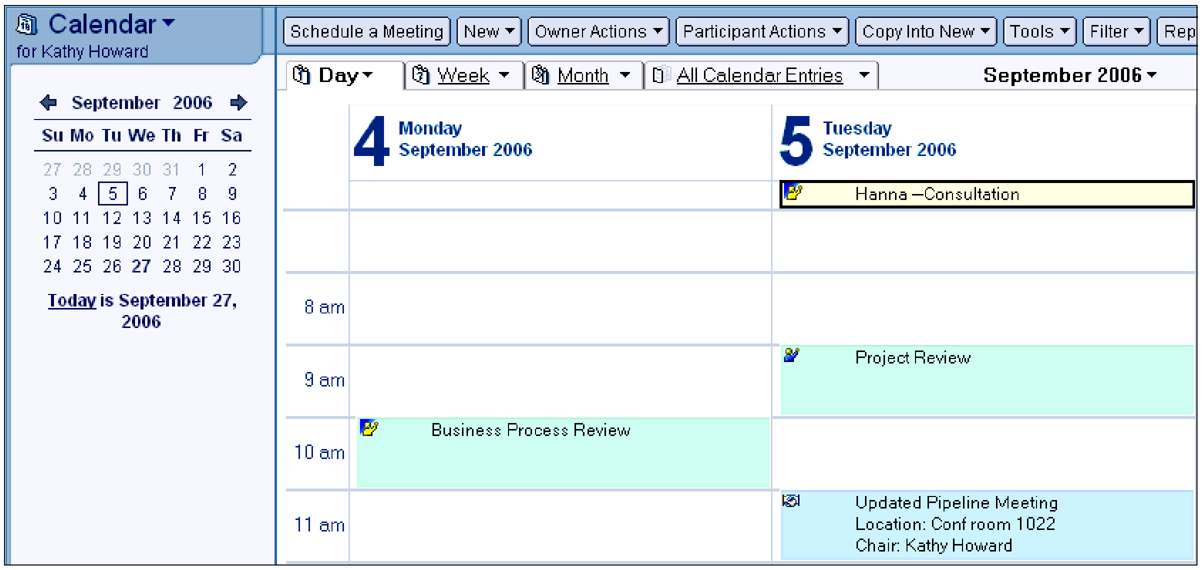
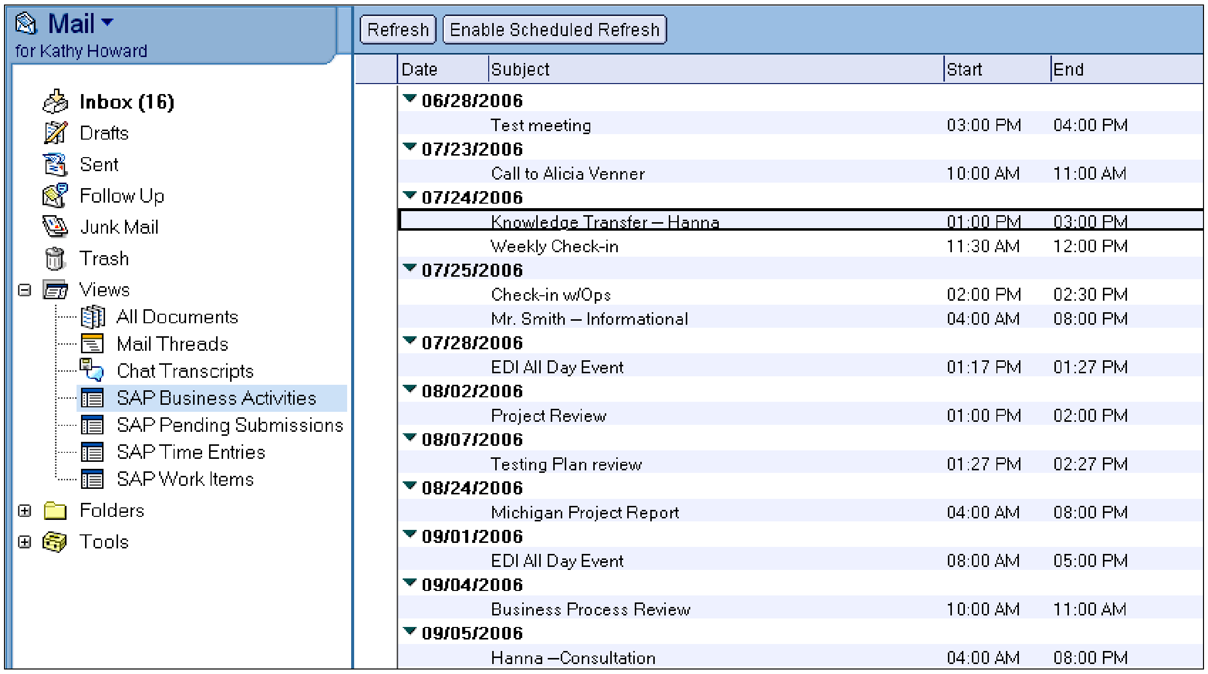


Figure 2-24 SAP business activities shown in the Calendar view

Business activities are always created on the SAP system and then copied to Notes; you cannot submit new or updated business activities to the SAP system from Notes.

The business activities you pull from SAP are also displayed in the SAP Business Activities view in Mail (Figure 2-25).



Date	Subject	Start	End
06/28/2006			
	Test meeting	03:00 PM	04:00 PM
07/23/2006			
	Call to Alicia Venner	10:00 AM	11:00 AM
07/24/2006			
	Knowledge Transfer – Hanna	01:00 PM	03:00 PM
	Weekly Check-in	11:30 AM	12:00 PM
07/25/2006			
	Check-in w/Ops	02:00 PM	02:30 PM
	Mr. Smith – Informational	04:00 AM	08:00 PM
07/28/2006			
	EDI All Day Event	01:17 PM	01:27 PM
08/02/2006			
	Project Review	01:00 PM	02:00 PM
08/07/2006			
	Testing Plan review	01:27 PM	02:27 PM
08/24/2006			
	Michigan Project Report	04:00 AM	08:00 PM
09/01/2006			
	EDI All Day Event	08:00 AM	05:00 PM
09/04/2006			
	Business Process Review	10:00 AM	11:00 AM
09/05/2006			
	Hanna –Consultation	04:00 AM	08:00 PM

Figure 2-25 SAP Business Activities view

This view provides a summary of all the business activities. You can refresh the list manually any time by clicking **Refresh**. To enable a scheduled agent to periodically refresh the list, click **Enable Scheduled Refresh**.

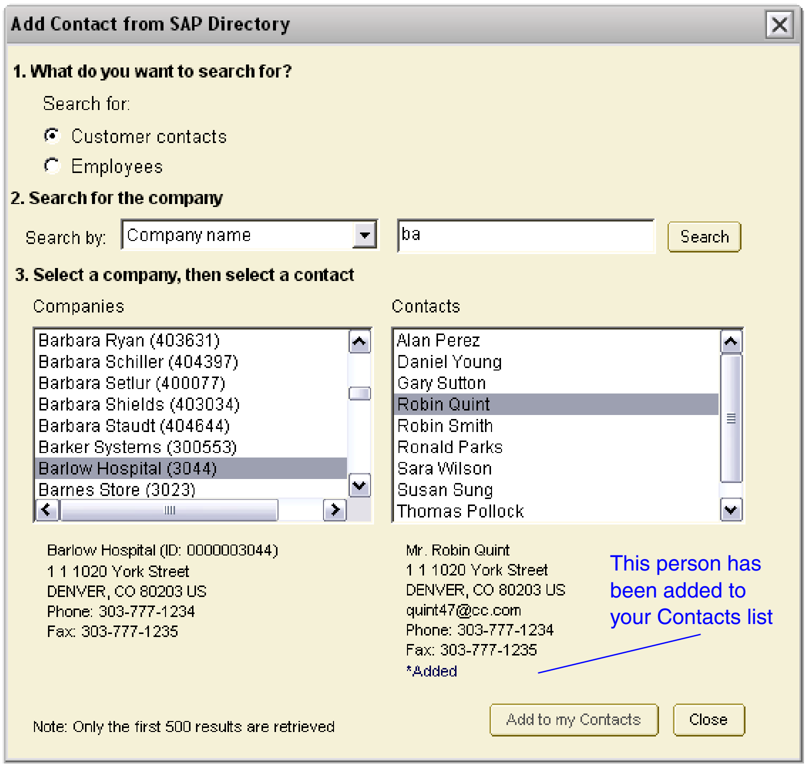
2.2.7 Contact management

Contact management enables you to copy contacts stored in the SAP system into your Notes personal name and address book. You can add customer contacts and employees.

In the address book, select **New** → **Contact from SAP Directory** in your personal address book.

For customer contacts, you first find the company by entering the company name (or other search string, such as city or state). You can enter just a few letters of the search string (a wildcard is automatically added to the string).

When you select a company in the list, the contacts for that company are displayed on the right (Figure 2-26).


Add Contact from SAP Directory
1. What do you want to search for?
Search for:
Customer contacts
Employees
2. Search for the company
Search by: Company name
ba
Search
3. Select a company, then select a contact
Companies
Contacts
This person has been added to your Contacts list
Add to my Contacts
Close

Barbara Ryan (403631)
Barbara Schiller (404397)
Barbara Setlur (400077)
Barbara Shields (403034)
Barbara Staudt (404644)
Barker Systems (300553)
Barlow Hospital (3044)
Barnes Store (3023)

Alan Perez
Daniel Young
Gary Sutton
Robin Quint
Robin Smith
Ronald Parks
Sara Wilson
Susan Sung
Thomas Pollock

Barlow Hospital (ID: 0000003044)
1 1 1020 York Street
DENVER, CO 80203 US
Phone: 303-777-1234
Fax: 303-777-1235

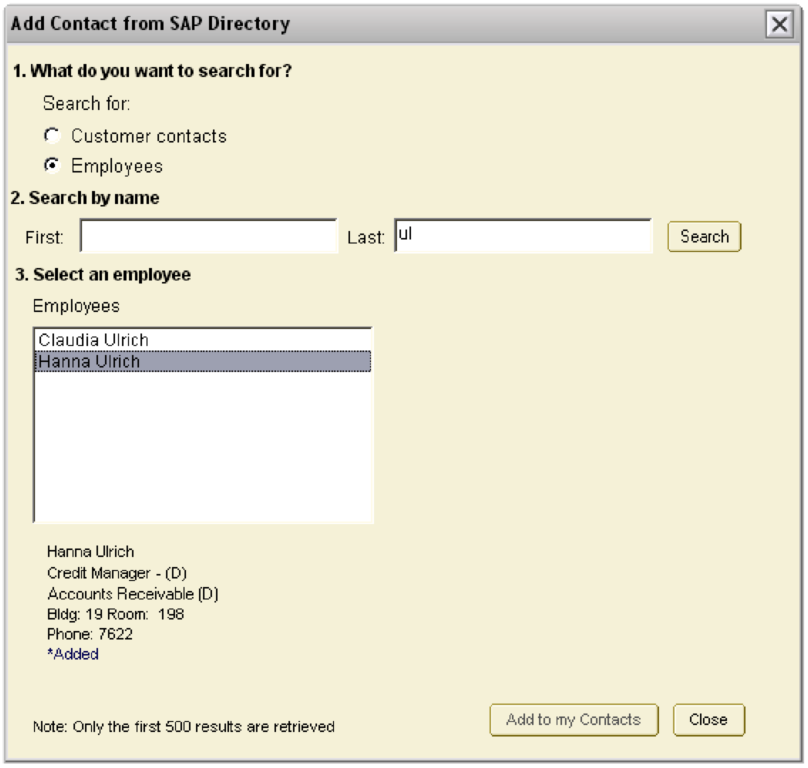
Mr. Robin Quint
1 1 1020 York Street
DENVER, CO 80203 US
quint47@cc.com
Phone: 303-777-1234
Fax: 303-777-1235
*Added

Note: Only the first 500 results are retrieved

Figure 2-26 Adding a client contact

Details about the company and the person appear under the list as you click through the lists. Click **Add to my Contacts** to add the selected contact. An indicator under the details lets you know the person has been added.

The window stays open so that you can search for additional companies, add more customer contacts, and add employees. To search for an employee to add, enter all or part of the first and last name, in any combination (Figure 2-27).

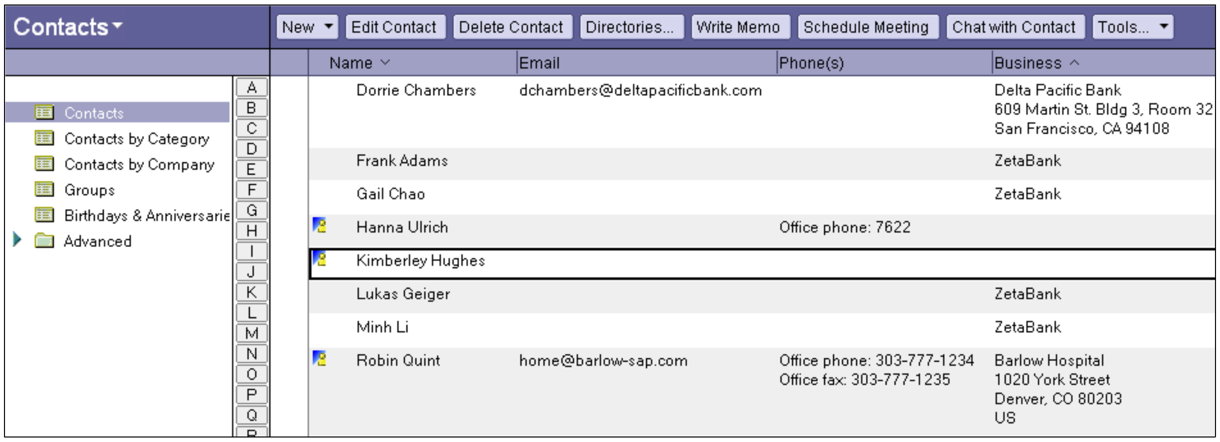


Hanna Ulrich
Credit Manager - (D)
Accounts Receivable (D)
Bldg: 19 Room: 198
Phone: 7622
*Added

Note: Only the first 500 results are retrieved

Figure 2-27 Adding an employee

When you click **Close**, the contacts you added are shown in your address book (Figure 2-28).



Name	Email	Phone(s)	Business
Dorrie Chambers	dchambers@deltapacificbank.com		Delta Pacific Bank 609 Martin St. Bldg 3, Room 32 San Francisco, CA 94108
Frank Adams			ZetaBank
Gail Chao			ZetaBank
Hanna Ulrich		Office phone: 7622	
Kimberley Hughes			
Lukas Geiger			ZetaBank
Minh Li			ZetaBank
Robin Quint	home@barlow-sap.com	Office phone: 303-777-1234 Office fax: 303-777-1235	Barlow Hospital 1020 York Street Denver, CO 80203 US

Figure 2-28 Contacts view in personal address book

You can keep the contacts you copied from SAP up to date by choosing one of the update commands on the Tools menu. You can also enable a scheduled agent to periodically refresh the contacts (Figure 2-29).

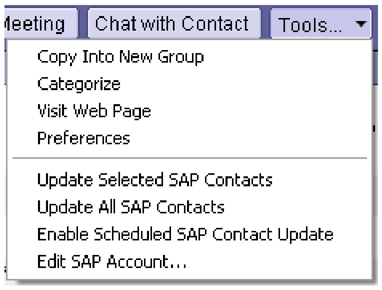


Figure 2-29 Tools menu in the personal name and address book

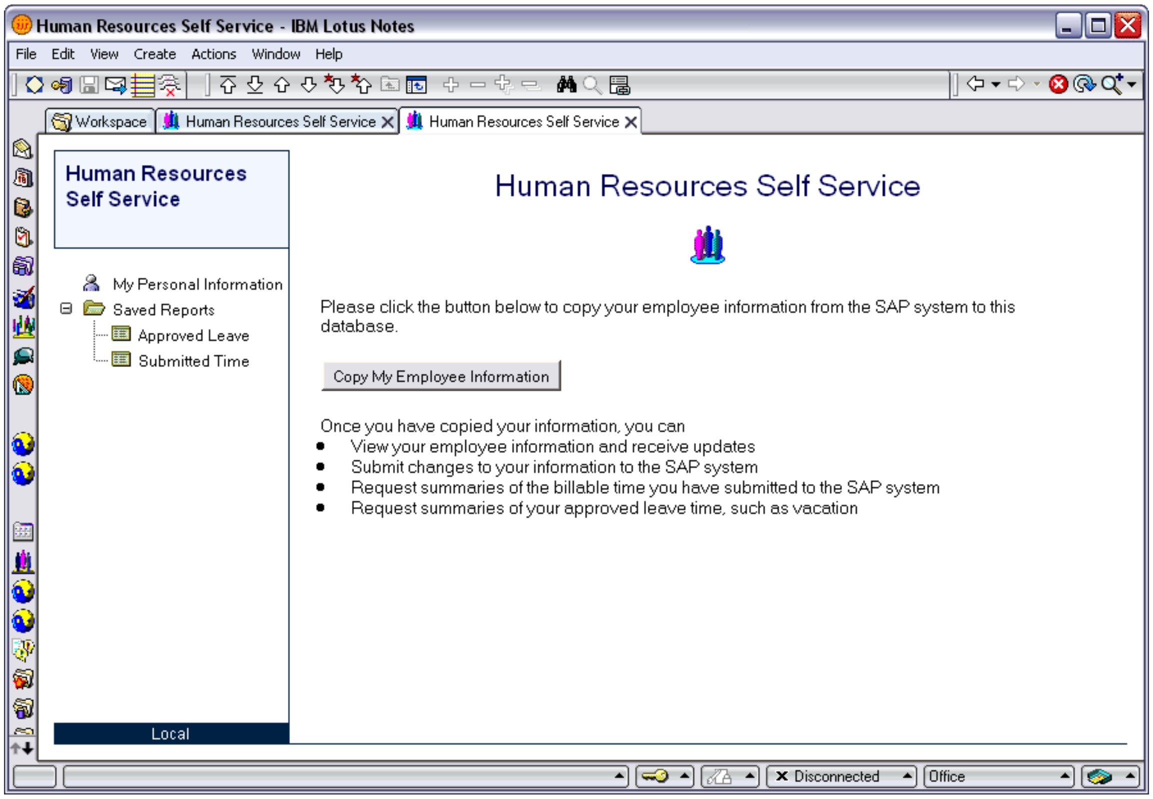
2.2.8 Employee and manager self service

Employee and manager self service provides a local Notes database in which you can view and update employee information stored in the SAP system. Managers can view their employees' records and get summary reports of vacation/leave time and billable time their employees have booked in SAP. Employees can view and update their personal information and get summary reports of their vacation/leave and billable time.

Employees and managers use different versions of the Human Resource Self Service database. Both versions include the ability pull your personal information from the SAP system and update it. The manager version adds the ability to pull the information of the manager's employees (for viewing but not updating). Both versions include the ability to get summary reports of your vacation/leave and billable time. The manager version allows managers to get summary reports for employees as well as themselves.

Working with employee information

When you first open the database, to get started you need to copy your personal information from SAP into the database by clicking **Copy My Employee Information** (Figure 2-30).


Human Resources Self Service - IBM Lotus Notes
File Edit View Create Actions Window Help
Workspace
Human Resources Self Service
Human Resources Self Service
Human Resources Self Service
My Personal Information
Saved Reports
Approved Leave
Submitted Time
Local
Disconnected
Office

Human Resources Self Service

Please click the button below to copy your employee information from the SAP system to this database.

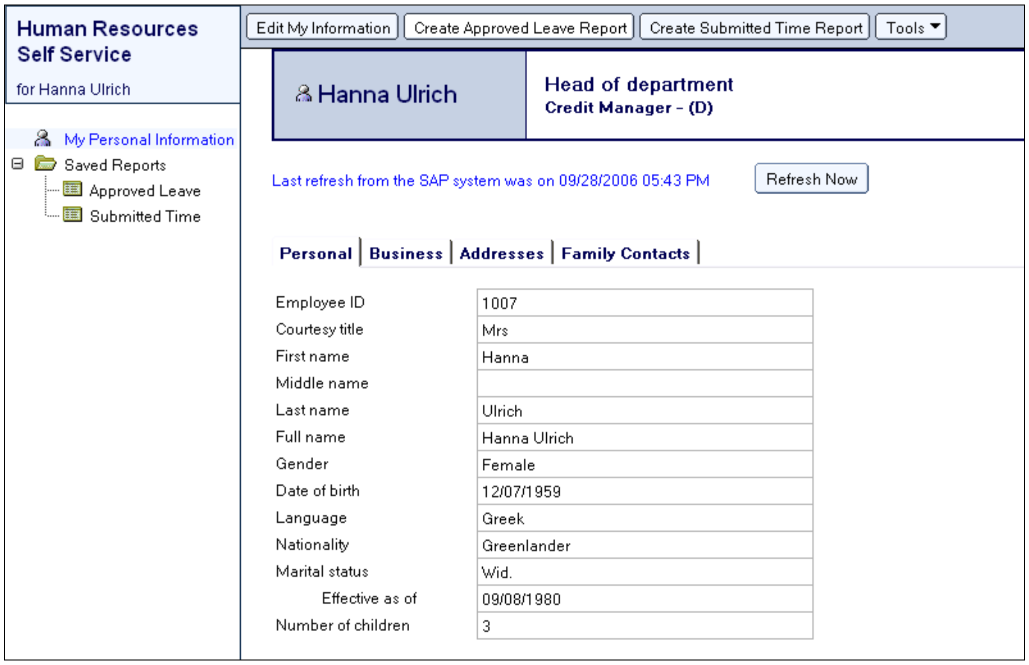
Copy My Employee Information

Once you have copied your information, you can

- View your employee information and receive updates
- Submit changes to your information to the SAP system
- Request summaries of the billable time you have submitted to the SAP system
- Request summaries of your approved leave time, such as vacation

Figure 2-30 Initial window for Human Resources Self Service database (employee version)

The employee information appears on the right (Figure 2-31).

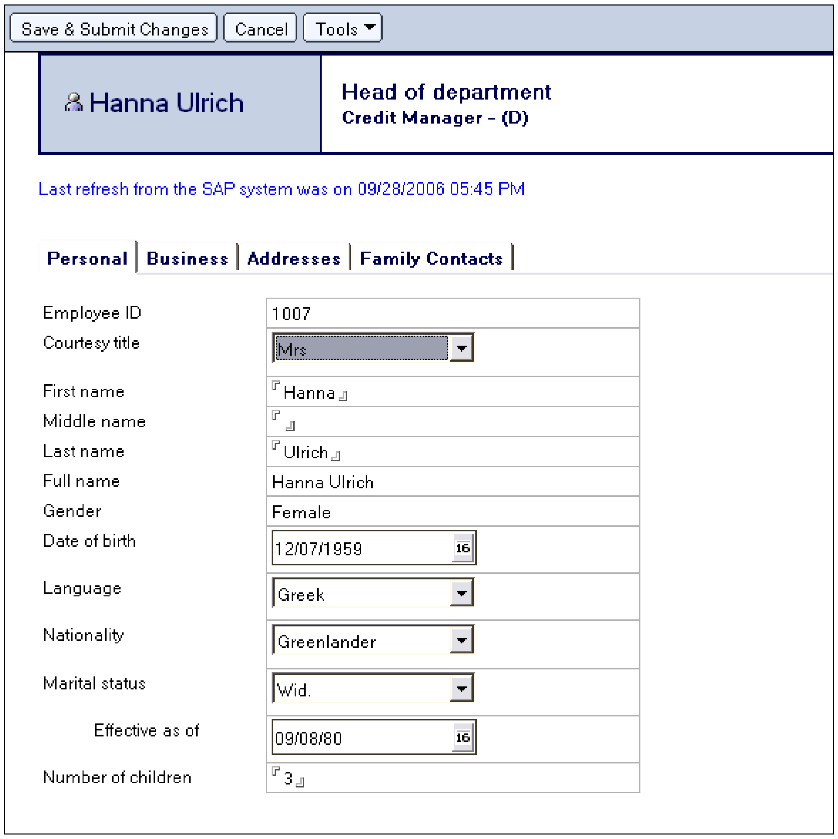


Last refresh from the SAP system was on 09/28/2006 05:43 PM

Employee ID	1007
Courtesy title	Mrs
First name	Hanna
Middle name	
Last name	Ulrich
Full name	Hanna Ulrich
Gender	Female
Date of birth	12/07/1959
Language	Greek
Nationality	Greenlander
Marital status	Wid.
Effective as of	09/08/1980
Number of children	3

Figure 2-31 Employee data in the Human Resources Self Service database (employee version)

To edit your employee information, click **Edit My Information**. You will be prompted to refresh the data from the SAP system if it has not been refreshed lately; this is to ensure that you are editing the most recent information. Figure 2-32 shows the Personal tab of the employee information. Changes you make here and on the Business tab are submitted to SAP when you click **Save & Submit Changes**.



Hanna Ulrich

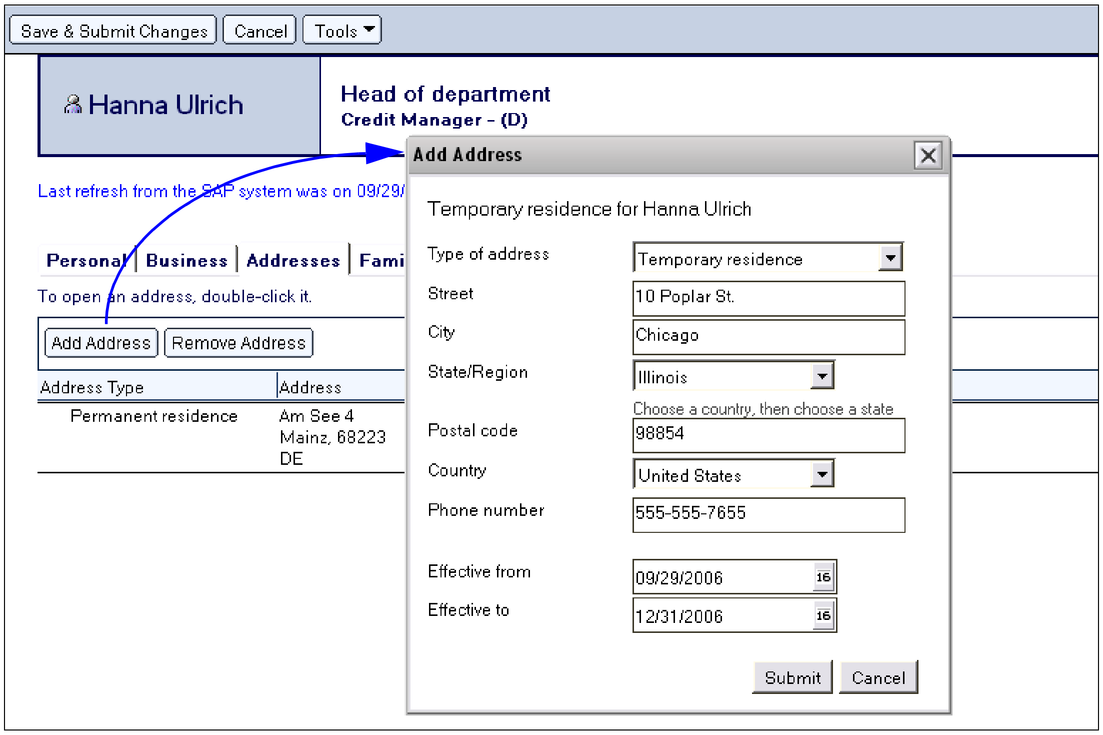
Head of department
Credit Manager - (D)

Last refresh from the SAP system was on 09/28/2006 05:45 PM

Employee ID	1007
Courtesy title	Mrs
First name	Hanna
Middle name	
Last name	Ulrich
Full name	Hanna Ulrich
Gender	Female
Date of birth	12/07/1959
Language	Greek
Nationality	Greenlander
Marital status	Wid.
Effective as of	09/08/80
Number of children	3

Figure 2-32 Editing employee information

Figure 2-33 shows adding an address on the Addresses tab. Changes you make to addresses and family contacts are submitted to SAP immediately when you click **Submit**.



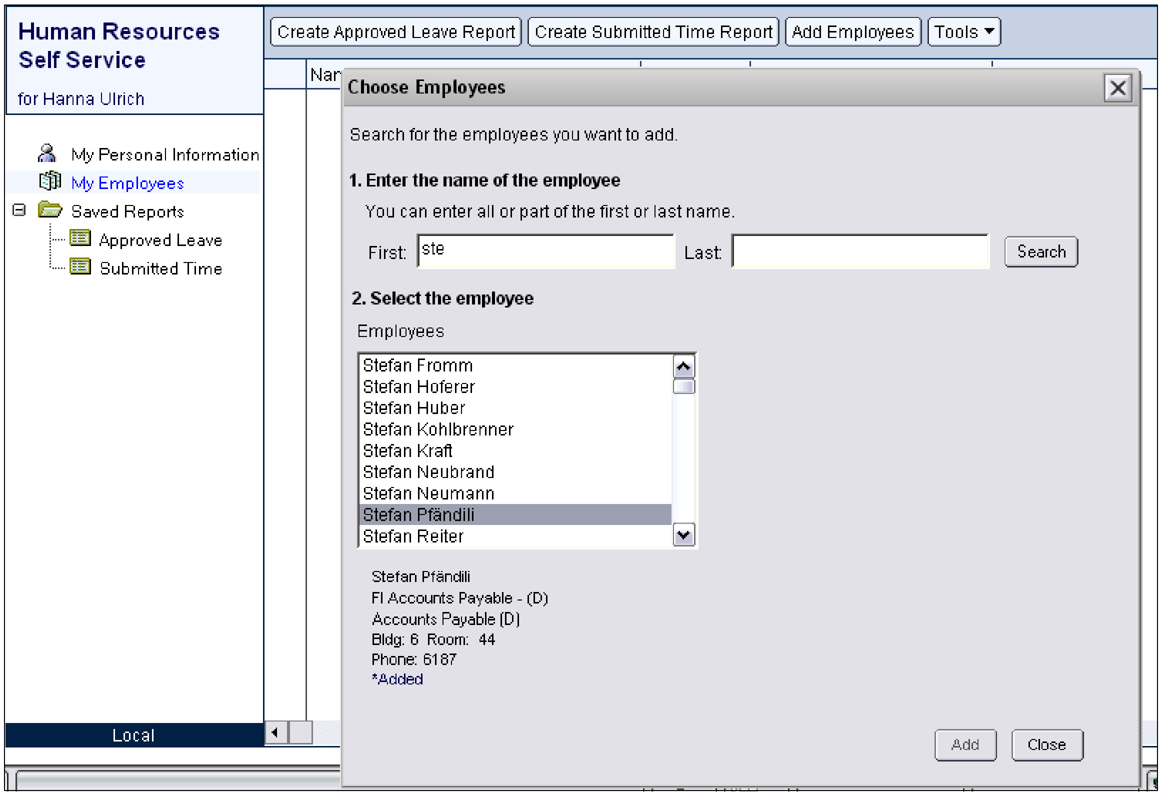
Address Type	Address
Permanent residence	Am See 4 Mainz, 68223 DE

Temporary residence for Hanna Ulrich

Type of address	Temporary residence
Street	10 Poplar St.
City	Chicago
State/Region	Illinois (Choose a country, then choose a state)
Postal code	98854
Country	United States
Phone number	555-555-7655
Effective from	09/29/2006
Effective to	12/31/2006

Figure 2-33 Adding an address

The manager version of the database includes a view for the employees' information. This My Employees view is initially blank; the first step is to add the employees whose information you want to see by clicking **Add Employees** (Figure 2-34).



Search for the employees you want to add.

1. Enter the name of the employee

You can enter all or part of the first or last name.

First: ste Last:

2. Select the employee

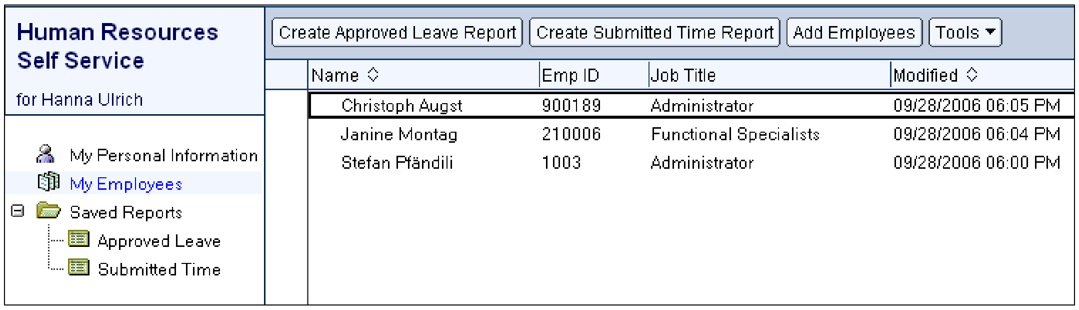
Employees

Stefan Fromm
Stefan Hoferer
Stefan Huber
Stefan Kohlbrenner
Stefan Kraft
Stefan Neubrand
Stefan Neumann
Stefan Pfändili
Stefan Reiter

Stefan Pfändili
FI Accounts Payable - (D)
Accounts Payable (D)
Bldg: 6 Room: 44
Phone: 6187
*Added

Figure 2-34 Adding employees to the Human Resources Self Service database (manager version)

Search by name for each employee and click **Add** (Figure 2-34). When you click **Close**, the employees are listed in the My Employees view (Figure 2-35).



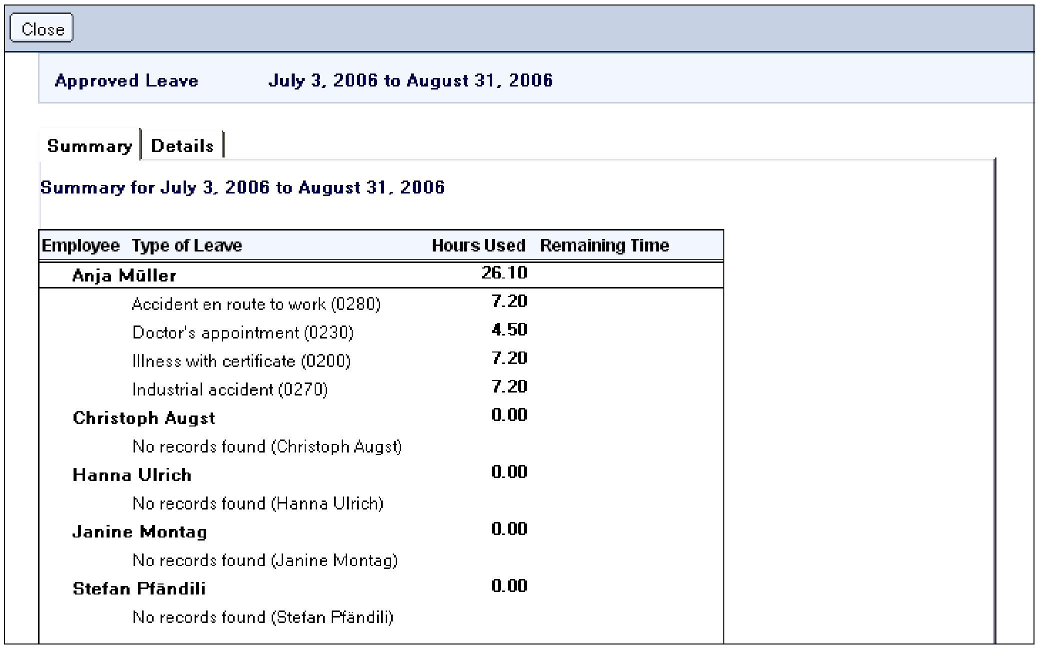
Name	Emp ID	Job Title	Modified
Christoph Augst	900189	Administrator	09/28/2006 06:05 PM
Janine Montag	210006	Functional Specialists	09/28/2006 06:04 PM
Stefan Pfändili	1003	Administrator	09/28/2006 06:00 PM

Figure 2-35 My Employees view

Summary reports

You can create summary reports of your approved leave time and submitted time. Managers can create summary reports for their employees' information as well as their own.

In Figure 2-36, the Summary page of an approved leave report shows the totals for each employee.

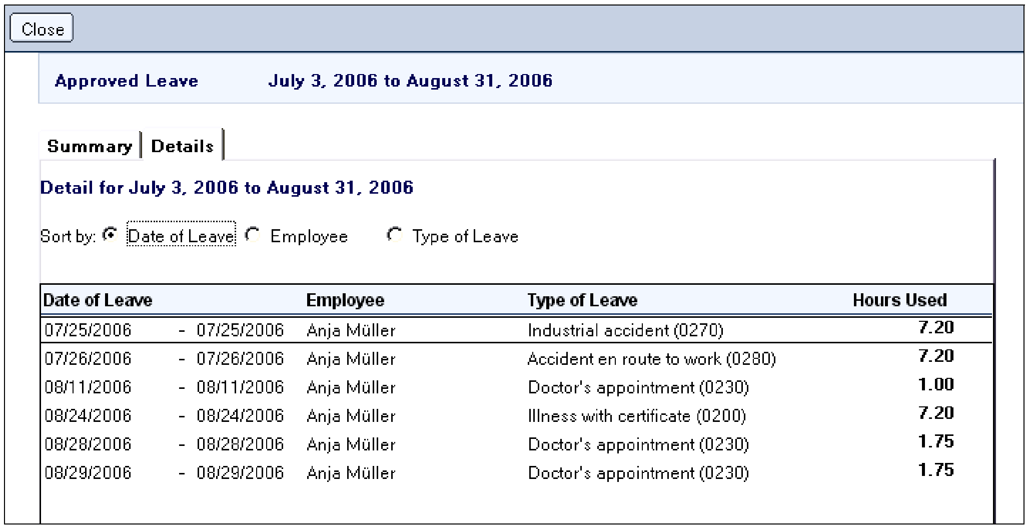


Summary for July 3, 2006 to August 31, 2006

Employee	Type of Leave	Hours Used	Remaining Time
Anja Müller		26.10	
	Accident en route to work (0280)	7.20	
	Doctor's appointment (0230)	4.50	
	Illness with certificate (0200)	7.20	
	Industrial accident (0270)	7.20	
Christoph Augst		0.00	
	No records found (Christoph Augst)		
Hanna Ulrich		0.00	
	No records found (Hanna Ulrich)		
Janine Montag		0.00	
	No records found (Janine Montag)		
Stefan Pfändili		0.00	
	No records found (Stefan Pfändili)		

Figure 2-36 Summary page of an Approved Leave report

The Details page of an approved leave report shows the individual leave entries and allows you to sort by date, employee, and type of leave (Figure 2-37).



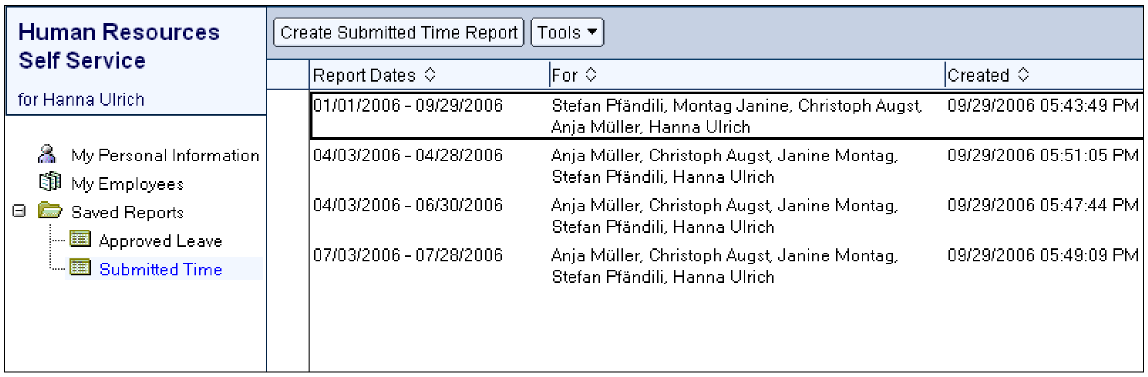
Detail for July 3, 2006 to August 31, 2006

Sort by: [✓ Date of Leave] Employee Type of Leave

Date of Leave	Employee	Type of Leave	Hours Used
07/25/2006 - 07/25/2006	Anja Müller	Industrial accident (0270)	7.20
07/26/2006 - 07/26/2006	Anja Müller	Accident en route to work (0280)	7.20
08/11/2006 - 08/11/2006	Anja Müller	Doctor's appointment (0230)	1.00
08/24/2006 - 08/24/2006	Anja Müller	Illness with certificate (0200)	7.20
08/28/2006 - 08/28/2006	Anja Müller	Doctor's appointment (0230)	1.75
08/29/2006 - 08/29/2006	Anja Müller	Doctor's appointment (0230)	1.75

Figure 2-37 Details page

The summary reports you create are saved in the Approved Leave and Submitted Time views. Figure 2-38 is an example of a Submitted Time view.



Report Dates	For	Created
01/01/2006 - 09/29/2006	Stefan Pfändili, Montag Janine, Christoph Augst, Anja Müller, Hanna Ulrich	09/29/2006 05:43:49 PM
04/03/2006 - 04/28/2006	Anja Müller, Christoph Augst, Janine Montag, Stefan Pfändili, Hanna Ulrich	09/29/2006 05:51:05 PM
04/03/2006 - 06/30/2006	Anja Müller, Christoph Augst, Janine Montag, Stefan Pfändili, Hanna Ulrich	09/29/2006 05:47:44 PM
07/03/2006 - 07/28/2006	Anja Müller, Christoph Augst, Janine Montag, Stefan Pfändili, Hanna Ulrich	09/29/2006 05:49:09 PM

Figure 2-38 Submitted Time view

Summary reports are snapshots of the data at the time of the request; you cannot update them over time. They are saved locally, so you can read them when offline.

2.3 Installing Notes access for SAP solutions

This section provides some basic information about installation. Refer to the Help file "Installing Notes access for SAP solutions Help" (nass_installhelp.nsf) for complete information.

When you install Lotus Domino 7.0.2 on the server, several Notes access for SAP solutions databases are automatically installed to the server data directory, as listed in Table 2-2.

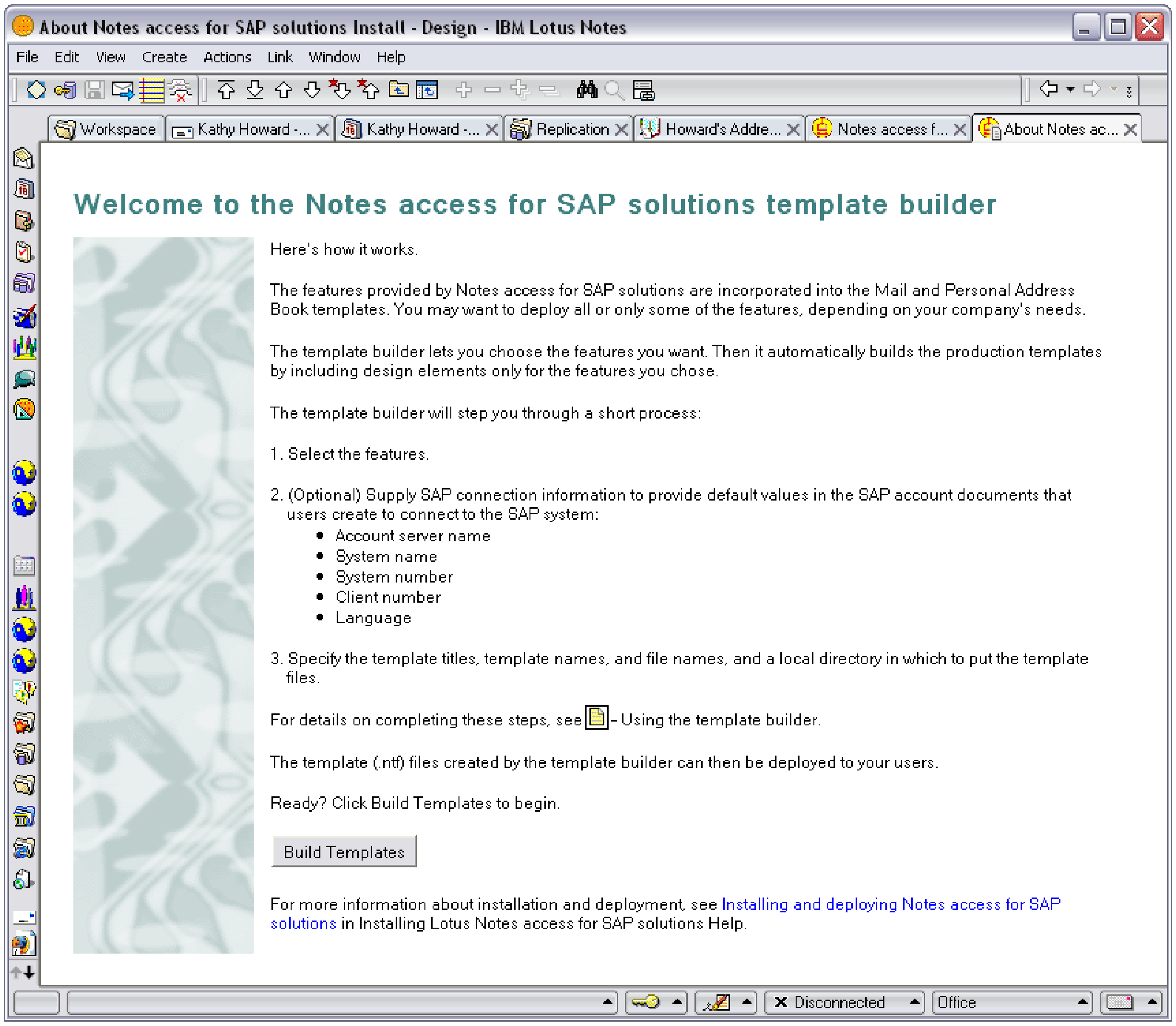
Table 2-2 Databases installed with Lotus Domino 7.0.2

Database title	Database file name	Description
Notes access for SAP solutions Install	NaSsInstall.nsf	Contains the template builder used to create the required templates. Contains file attachments of all the base templates (NTFs) from which the production templates are generated.
Installing Notes access for SAP solutions Help	nass_installhelp.nsf	Describes how to install and deploy the SAP integration features.
Notes access for SAP solutions Help	nass_userhelp.nsf	Describes how to use the SAP integration features.

For users, the SAP connector software and the Notes access for SAP solutions Help (nass_userhelp.nsf) are installed as part of the normal Notes 7.02 client installation process.

About the Notes access for SAP solutions Install database

Opening the About document in the Notes access for SAP solutions database provides the Notes administrator with easy instructions and links to create the templates for their environment to deploy to their organization's users. For complete details about using the template builder, refer to the Help topic "Using the template builder" in nass_installhelp.nsf.



Welcome to the Notes access for SAP solutions template builder

Here's how it works.

The features provided by Notes access for SAP solutions are incorporated into the Mail and Personal Address Book templates. You may want to deploy all or only some of the features, depending on your company's needs.

The template builder lets you choose the features you want. Then it automatically builds the production templates by including design elements only for the features you chose.

The template builder will step you through a short process:

1. Select the features.

2. (Optional) Supply SAP connection information to provide default values in the SAP account documents that users create to connect to the SAP system:
   - Account server name
   - System name
   - System number
   - Client number
   - Language

3. Specify the template titles, template names, and file names, and a local directory in which to put the template files.

For details on completing these steps, see - Using the template builder.

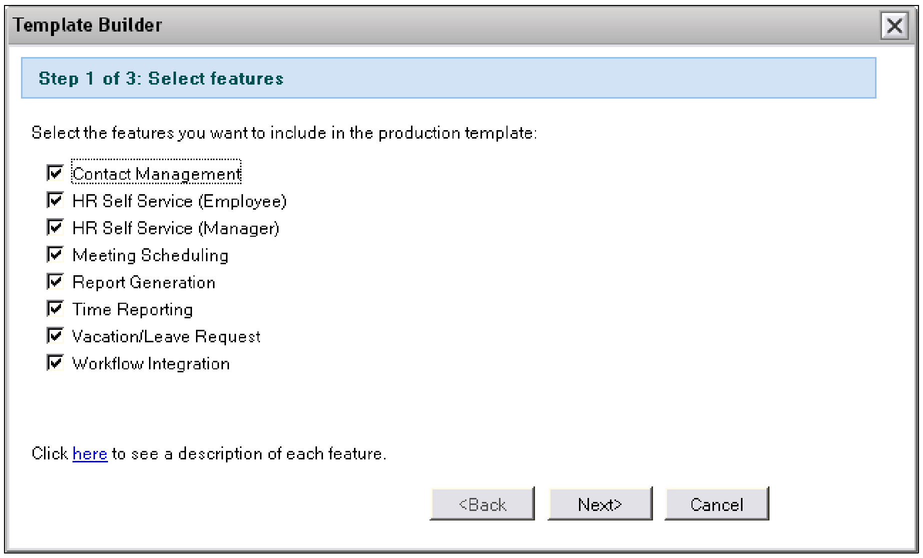
The template (.ntf) files created by the template builder can then be deployed to your users.

Ready? Click Build Templates to begin.

For more information about installation and deployment, see Installing and deploying Notes access for SAP solutions in Installing Lotus Notes access for SAP solutions Help.

Figure 2-39 About document for the template builder database

Step 1 of the template builder offers choices for which features/templates you want to deploy (Figure 2-40).


Template Builder

Step 1 of 3: Select features

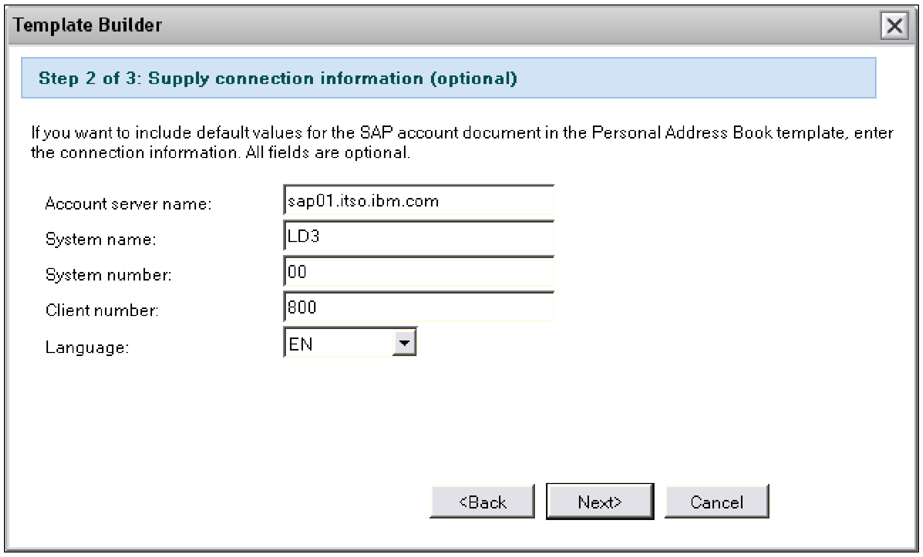
Select the features you want to include in the production template:

- [✓] Contact Management
- [✓] HR Self Service (Employee)
- [✓] HR Self Service (Manager)
- [✓] Meeting Scheduling
- [✓] Report Generation
- [✓] Time Reporting
- [✓] Vacation/Leave Request
- [✓] Workflow Integration

Click here to see a description of each feature.

Figure 2-40 Step 1 of the Template Builder

Step 2 of the template builder lets you optionally supply SAP connection information for the SAP account document in the Personal Address Book template (Figure 2-41).


Template Builder

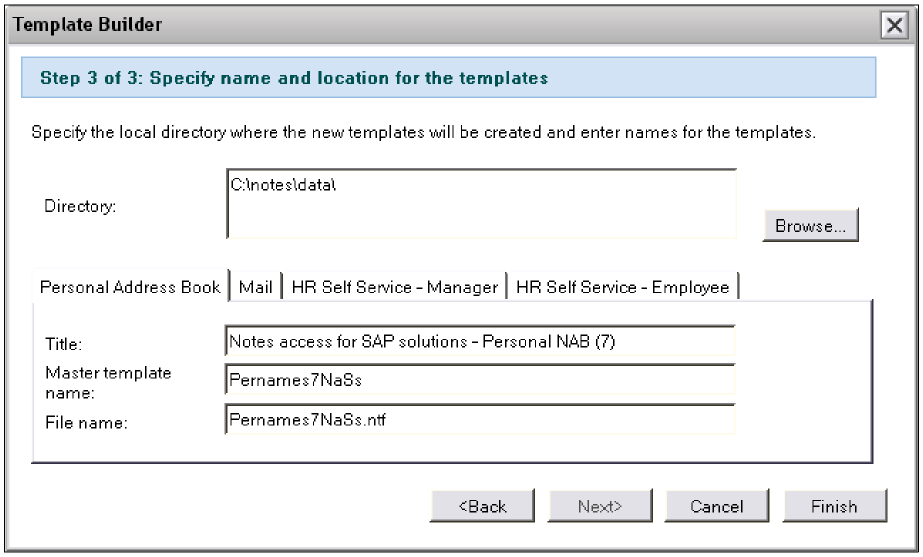
Step 2 of 3: Supply connection information (optional)

If you want to include default values for the SAP account document in the Personal Address Book template, enter the connection information. All fields are optional.

Account server name:	sap01.itso.ibm.com
System name:	LD3
System number:	00
Client number:	800
Language:	EN

Figure 2-41 Step 2 of the Template Builder

The final step enables you to specify the template names and locations (Figure 2-42).


Template Builder

Step 3 of 3: Specify name and location for the templates

Specify the local directory where the new templates will be created and enter names for the templates.

Directory: C:\notes\data\

Title: Notes access for SAP solutions - Personal NAB (7)

Master template name: Pernames7NaSs

File name: Pernames7NaSs.ntf

Figure 2-42 Step 3 of the Template Builder

After you click **Finish**, the templates required for the features you chose are created in the directory you specified in step 3 of the template builder. The status bar in Notes shows a success message when the templates have been created.

3 Customization

In this chapter, we describe how to tailor the supplied templates to your own needs in case your company plans to use them in a slightly different way. In contrast to Chapter 4, “Extending” on page 63, this chapter uses the Lotus Connector for LotusScript Extensions (LC LSX) classes instead of the prepared Lotus Notes access for SAP solutions libraries to help you develop a better understanding of the underlying LotusScript code. (If you do not want to go into details but just get going with Lotus Notes access for SAP solutions, you might want to skip this chapter.)

We also present a kind of road map to integrating Notes and SAP, using Lotus Notes access for SAP solutions along the way, which we follow in Chapter 4, “Extending” on page 63.

We discuss the following topics in this chapter:

- Scenario introduction: The ITSO Bank
- A road map for SAP access
- The SAP GUI client
- Customizing the Contacts template

3.1 Scenario introduction: ITSO Bank

To better illustrate all the concepts explained in this Redpaper and to tie them all together into a common scenario, we use the fictitious organization *ITSO Bank* throughout this paper.

ITSO Bank just finished upgrading all their Notes clients to version 7.0.2, as laid out in the Redpaper *Understanding Lotus Notes Smart Upgrade*, REDP-4180. Like many large companies, ITSO Bank uses SAP software for certain tasks, so integrating it with Notes through Lotus Notes access for SAP solutions makes perfect sense—not only from a technical, but also from a business point of view.

3.2 Developing a road map for accessing SAP

This section is an overview of the most important Lotus Connector LotusScript Extensions (LC LSX) classes. They form the building blocks for what we do throughout the whole integration scenario, so it is essential that you know how to use them effectively.

Important: The LC LSX classes will only be available with a full installation of the Lotus Notes client. See Chapter 5, "Troubleshooting" on page 91 for details.

We point out in this chapter that accessing SAP follows the same basic rules as any integration of Lotus Notes with any other external system. Figure 3-1 illustrates a generic road map for the Notes and SAP integration.

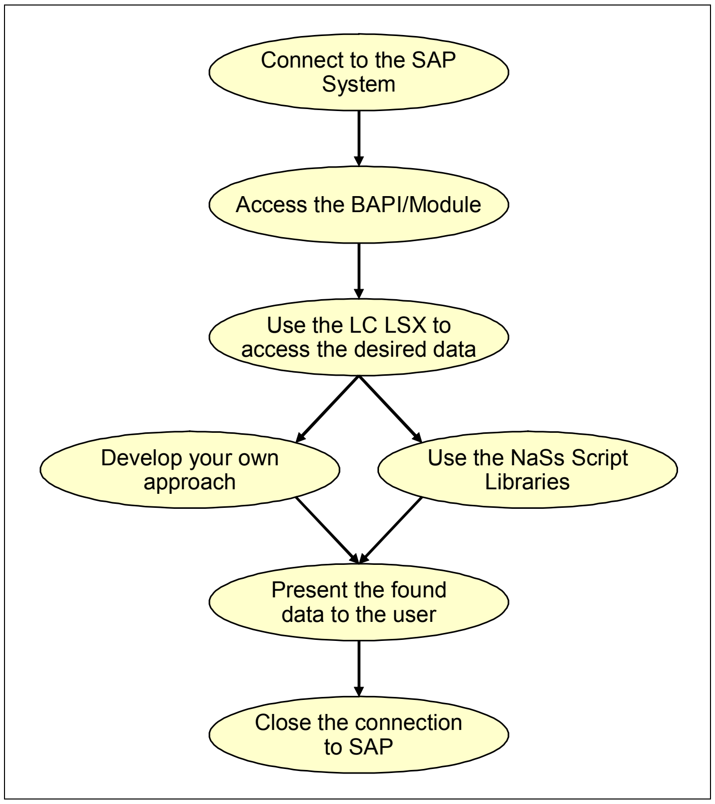

Connect to the SAP System
Access the BAPI/Module
Use the LC LSX to access the desired data
Develop your own approach
Use the NaSs Script Libraries
Present the found data to the user
Close the connection to SAP

Figure 3-1 Generic road map for Notes/SAP integration (draft version)

3.2.1 The LCSession class

This global helper class serves four main purposes, two of which might turn out especially important for our work:

- Output of error and status information, that is, for use in error handling code
- Control over the Connection Pooling features using a simple Boolean on/off switch

3.2.2 The LCConnection class

The LCConnection class represents your connection to a data source. Using this object, you can do about anything that involves reading or making changes to an external system.

To initialize a connection, just supply the name of the connector you want to use:

```
Dim SAPCon As New LCConnection("SAP")
```

The LCConnection class can contain information about:

- The type of connection (in our case, it is just “SAP” all the time).
- To which SAP system you are connecting (address, system, and client number).
- The user ID, password, and language used to access SAP.
- Which RFC module, BAPI®, or transaction you are using (optional).
- The debug level and character set (optional).

Before you can use the LCConnection object for any operations, you must call its Connect method to establish a connection to the SAP system.

You might have several simultaneous connections to the same database or even the same metadata. We explain this in more detail later.

3.2.3 The LCFieldlist class

The LCFieldlist class is the basic unit of information that you can read from or write to a database. It usually represents one or more rows of data. A field list has two types of indexes used to find information within it: a field index to select the column, and a row index to select the row. Most field lists in applications contain only one row; you control this when you write your code.

An LCFieldlist class is an ordered collection of LCField objects, including a name for each field. Multiple LCFieldlists can reference the same LCField object by different names (or by the same name). Understanding this is critical to writing efficient code.

3.2.4 The LCField class

The LCField object represents a single SAP or Notes field. It contains a data type, a value (or an array of values), and information about whether null values are allowed or whether it is a key field. It does not contain any field name, however, because the same LCField object can be known by different names in different contexts.

The methods of LCField allow data to be converted to and from different data types and different character sets. An LCField of an appropriate type can also contain binary data, including structured information such as Notes rich text fields and Notes multi-valued fields.

3.3 SAP software

This section acquaints you with the questions you need to ask your SAP administrator in order to connect to the SAP system.

Note: If you want to connect to SAP:

- As a user: You must obtain the file librfc32.dll from SAP. See “Installing the SAP client libraries” in *Installing Lotus Notes access for SAP solutions Help*.
- As a developer: We highly recommend that you install the SAP GUI client.

The user only needs to connect to SAP through the Notes client. Therefore, just librfc32.dll will do. But as a developer, you will be doing a lot more with SAP, so you will need a full-fledged proprietary client sooner or later.

The SAP GUI client software

In theory, you could develop all the necessary code on the Notes side only. But in real life, this is not what you will do. As soon as you run into problems with any SAP-related code, you will need a tool to help you discover what happened and check the SAP side. Notes cannot do that for you.

But the SAP GUI client can. It not only serves as an end user client, the software also features a set of tools that are useful when you need to examine modules, debug their output, examine the contents of certain tables, and so on.

Important: Make sure that you obtain a *dialog* user account with SAP rather than a mere *RFC* account. They will both work with Lotus Notes access for SAP solutions (except for the SAP workflow integration), but the latter will not allow you to use the SAP GUI client. See Chapter 5, “Troubleshooting” on page 91 for details.

3.4 Customizing the Contacts template

The ITSO Bank management board has decided that the integration of contacts data from SAP CRM is not enough for their planned business case. They want to modify their users’ personal address books to also integrate ITSO Bank’s vendors.

We now take a closer look on how to implement this feature by modifying the Lotus Notes access for SAP solutions Contacts form within the personal name and address book template.

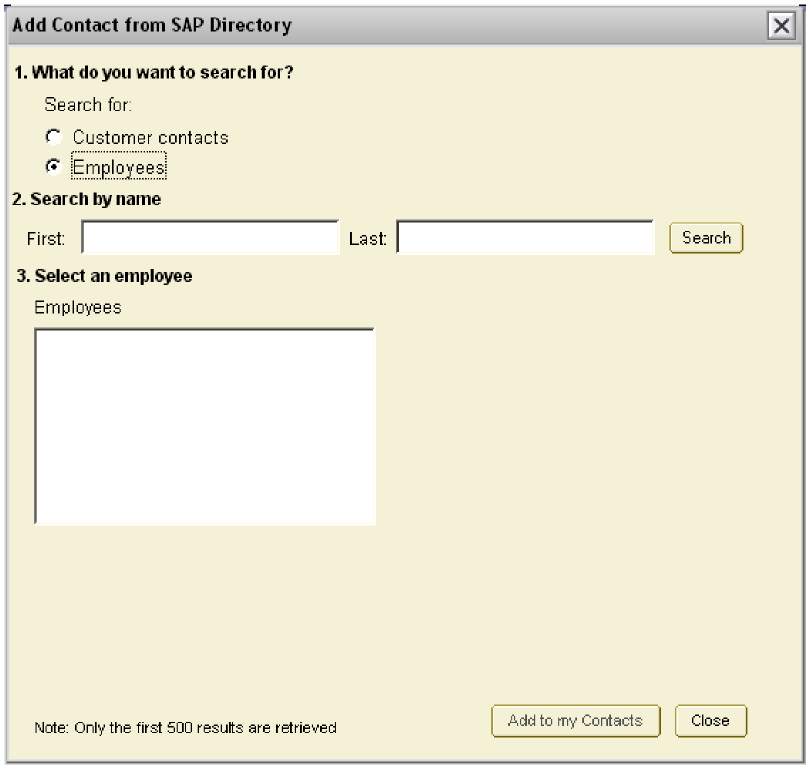
Customizing the personal name and address book template can be broken down into three major sections:

- The front-end Notes application (the user’s perspective)
- Finding the SAP module or modules that suit your purposes best
- The back-end LotusScript (the developer’s perspective)

3.4.1 Getting user input

One of the first steps is always to find out what the user wants to know from SAP. There is usually more than one way to determine this. In this specific case, most of the user interface work has already been done. The template contains a dialog box where users can select what kind of contact information they want to fetch from the SAP system: customer contacts or employees, as displayed in Figure 3-2.

This dialog box uses the Notes form (SAPContactDlg) from the personal address book template (pernab.ntf). Later, we customize the form to display the new controls for looking up vendors. For now, let's look at the current dialog box to understand what information we need to look up in the SAP system.


Add Contact from SAP Directory

1. What do you want to search for?

Search for:

( ) Customer contacts
(•) Employees

2. Search by name

First: Last:

3. Select an employee

Employees

Note: Only the first 500 results are retrieved

Figure 3-2 Dialog box for adding customers and employees to the personal address book

As shown in Figure 3-2:

1. The dialog box lets you choose what type of contact to look up, which does not require any information from the SAP system. (We add a choice for vendor later.)
2. The user enters a search string. The search string will need to be passed to the SAP system.
3. The list of vendors matching the search string will be shown. When a contact is selected in this list, the details about the contact are displayed, so the details also need to be fetched from the SAP system.
4. The user selects a specific vendor to be added to the personal address book. A new Notes document is created, and the data from SAP is inserted.

3.4.2 Connecting to SAP

First, the most important step in the process is to connect to the SAP system.

The following LotusScript agent can be used to test your connectivity to an SAP system (Example 3-1). This is the most basic way of connecting, so it contains no error checking whatsoever. Note the line:

```
Uselsx "*lsxlc"
```

This tells the script that we want to use the LC LSX.

Example 3-1 Agent to test SAP connectivity

```
[(Declarations) Section]
Option Public
Option Declare
Uselsx "*lsxlc"

[Initialize Section]
Sub Initialize

Dim src As New LCConnection ("SAP")

‘ Set the basic parameters, provide your own data here
src.Userid="muster"
src.Password="ides"
src.Client="800"
src.Destination="LD3"
src.SystemNo=0
src.Language="EN"
src.Server="sap01.itso.ibm.com"
src.Connect

‘ Do some stuff...
MessageBox(“Successfully connected to SAP!“)

‘ Never forget this last line:
src.Disconnect

End Sub
```

3.4.3 Finding the right module

Now that we have established a connection to SAP, it is time to find the right module to use in our business scenario. This is usually where your SAP staff comes into play. Because our fictitious ITSO Bank has several of these people, they are unlikely to run into many problems here. But in the real world, things often look a little different, so we will pretend that the ITSO Bank SAP staff is on holiday and we are left to our own devices.

The following sections assume that you are trying to implement the vendor integration without any help from your SAP staff. This is not an unlikely situation to encounter in smaller projects. But do not be intimidated by SAP—explore the system! It is usually quite helpful for your Notes development skills to be forced to hunt through SAP. It helps to develop an overview of SAP that you might not have had prior to this exercise and that is hard to get otherwise.

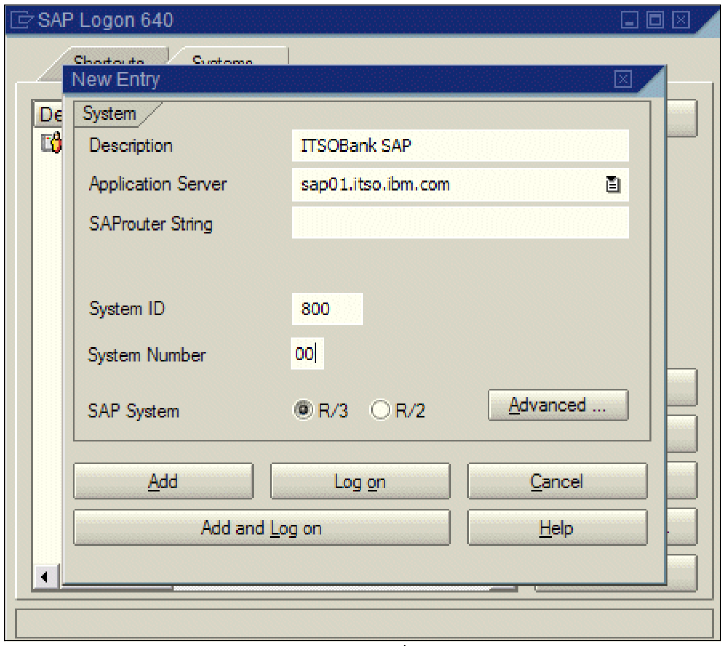
What exactly is our task? We need to find an SAP module that helps us to find vendor data. To check for modules that might be helpful, we use the SAP GUI client.

Important: Our approach to finding a suitable SAP module is not necessarily a systematic one. Usually, you first look for a module that shows the list of vendors and have the Notes user pick a vendor from that list (see 3.4.1, "Getting user input" on page 38). In the second step, you look for a module to look up this vendor's details and print them on the window.

We do not know where to get the list of vendors yet however, so we start our approach the other way around, hoping to find a reference to the vendor list somewhere in the get-me-the-details module. There are of course even more ways to go about this.

After installing the SAP GUI client with the default settings, we create a new SAP logon, as displayed in Figure 3-3. To do this, you need to set your logon properties, just as we did in the connect agent example in "Connecting the agent to SAP" on page 48. (Supply your own data here.)

We used the SAP GUI 640 client for the purposes of this paper, so you might need to modify your SAP GUI settings accordingly.



Description	ITSOBank SAP
Application Server	sap01.itso.ibm.com
SAProuter String	
System ID	800
System Number	00
SAP System	[✓ R/3] R/2

Figure 3-3 Create a new SAP logon[1]

Now we can connect to our SAP system using the SAP GUI client. We use the function builder module included in the client. This is the part of the SAP GUI client where the Advanced Business Application Programming, or ABAP™, functions are written, remotely comparable to Domino Designer on the Notes side. The fastest way to get there is to type the function's shortcut `se37` into to the quickfinder field as displayed in Figure 3-4 and press Enter. You can also select **Tools** → **ABAP Workbench** → **Development** → **Function Builder**.

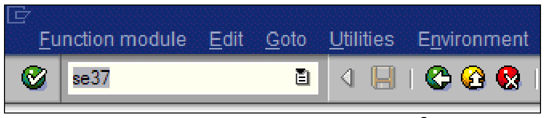
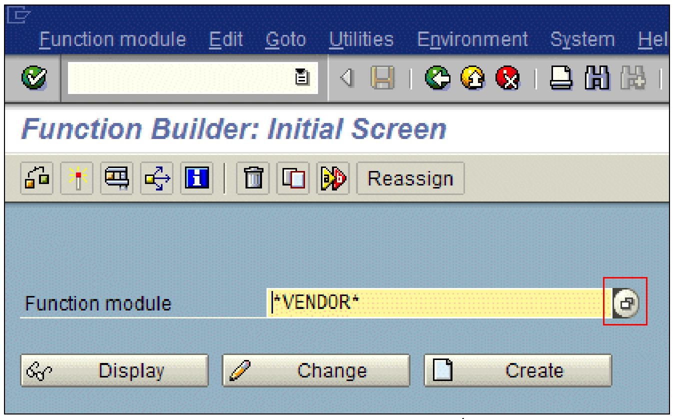


Figure 3-4 Calling the function builder[2]

[1] © SAP AG 2006. All rights reserved.
[2] © SAP AG 2006. All rights reserved.

Note: Usually, you can call any SAP function by typing its shortcut into the quickfinder field. But if you are changing from one module to another, you have to prefix your choice with `/N` to tell the SAP system that you want to change modules. Therefore, if you change from another module to `SE37`, call it `/NSE37`.

This takes us to the function builder window (Figure 3-5). Here, we are prompted to enter the name of the module we want to build or edit. This field also doubles as a search facility for existing function names. We look for vendors there by typing `*VENDOR*` into the box and clicking the search symbol (to the right of the box).

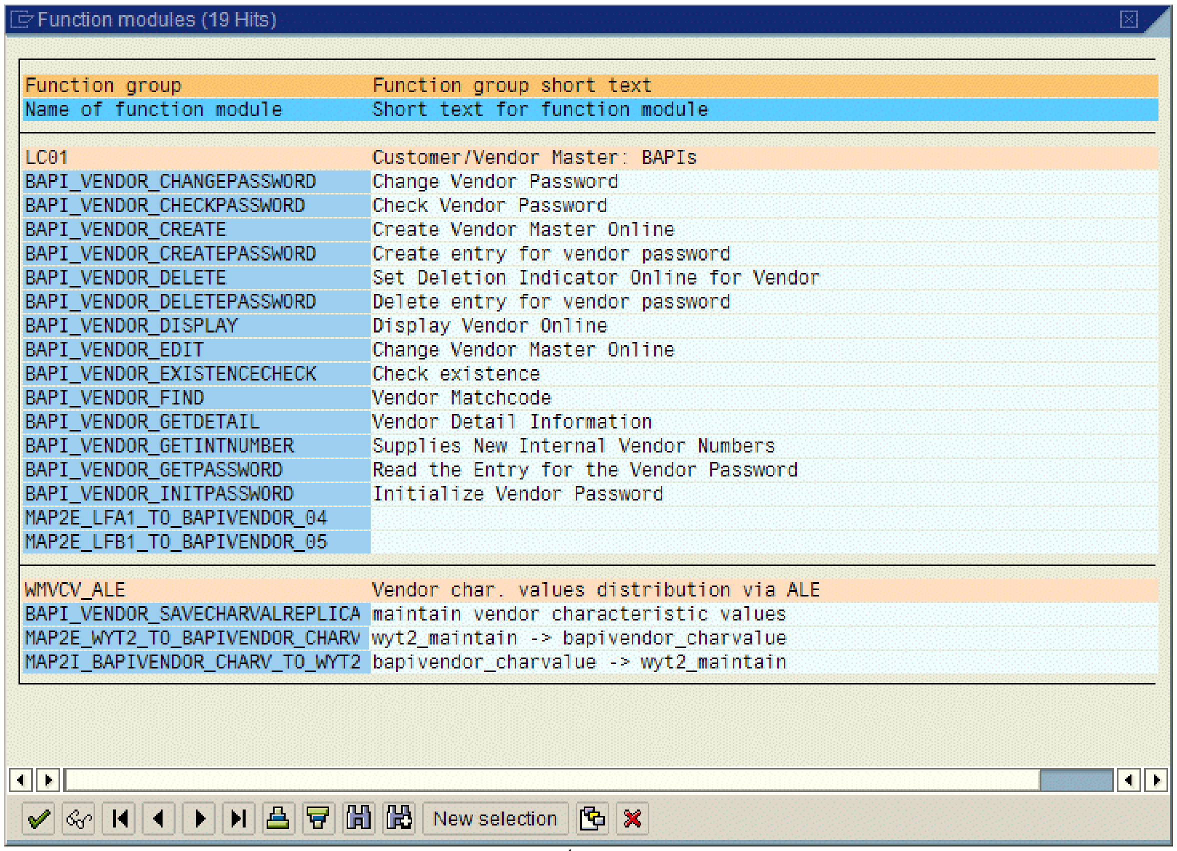


Function module: *VENDOR*

Figure 3-5 Searching for the right module[1]

[1] © SAP AG 2006. All rights reserved.

The search results in too many hits (172 on the system we used), so we have to narrow down our search term a little. We choose to integrate the term BAPI (business API) into our search, so we are now searching for the term *BAPI*VENDOR*. On the system we used, this returned 19 hits (Figure 3-6). Most of the results have some sort of comment associated; sometimes that is also helpful.



Function group / Name of function module	Function group short text / Short text for function module
LC01	Customer/Vendor Master: BAPIs
BAPI_VENDOR_CHANGEPASSWORD	Change Vendor Password
BAPI_VENDOR_CHECKPASSWORD	Check Vendor Password
BAPI_VENDOR_CREATE	Create Vendor Master Online
BAPI_VENDOR_CREATEPASSWORD	Create entry for vendor password
BAPI_VENDOR_DELETE	Set Deletion Indicator Online for Vendor
BAPI_VENDOR_DELETEPASSWORD	Delete entry for vendor password
BAPI_VENDOR_DISPLAY	Display Vendor Online
BAPI_VENDOR_EDIT	Change Vendor Master Online
BAPI_VENDOR_EXISTENCECHECK	Check existence
BAPI_VENDOR_FIND	Vendor Matchcode
BAPI_VENDOR_GETDETAIL	Vendor Detail Information
BAPI_VENDOR_GETINTNUMBER	Supplies New Internal Vendor Numbers
BAPI_VENDOR_GETPASSWORD	Read the Entry for the Vendor Password
BAPI_VENDOR_INITPASSWORD	Initialize Vendor Password
MAP2E_LFA1_TO_BAPIVENDOR_04	
MAP2E_LFB1_TO_BAPIVENDOR_05	
WMVCV_ALE	Vendor char. values distribution via ALE
BAPI_VENDOR_SAVECHARVALREPLICA	maintain vendor characteristic values
MAP2E_WYT2_TO_BAPIVENDOR_CHARV	wyt2_maintain -> bapivendor_charvalue
MAP2I_BAPIVENDOR_CHARV_TO_WYT2	bapivendor_charvalue -> wyt2_maintain

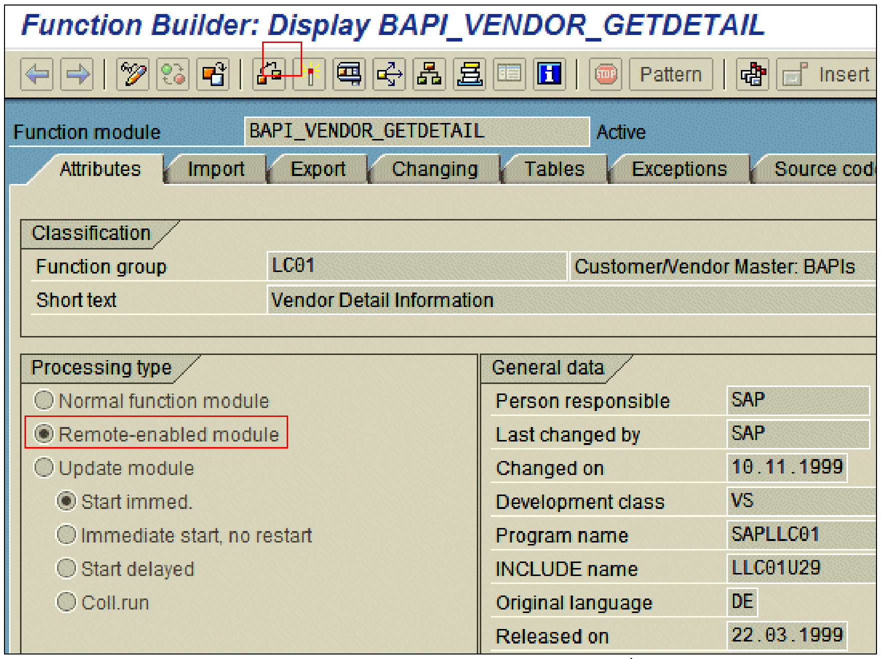
Figure 3-6 Result set of the function module search[1]

Of these 19 modules, the one called BAPI_VENDOR_GETDETAIL looks the most promising, so we double-click it. This brings us back to the search screen with the name of the module already filled in. Click **Display** to examine the module more closely.

[1] © SAP AG 2006. All rights reserved.

This brings us to the window displayed in Figure 3-7 that has a tabbed layout, initially showing the module's source code. We want to know whether this module can be called remotely, so we check the **Attributes** tab.

Important: If a module's Attributes tab indicates that the module cannot be called remotely, we cannot access it with Lotus Notes access for SAP solutions. Go find another one.



Function module: BAPI_VENDOR_GETDETAIL Active

Classification

Function group	LC01	Customer/Vendor Master: BAPIs
Short text	Vendor Detail Information	

Processing type

- ○ Normal function module
- ● Remote-enabled module
- ○ Update module
  - ● Start immed.
  - ○ Immediate start, no restart
  - ○ Start delayed
  - ○ Coll.run

General data

Person responsible	SAP
Last changed by	SAP
Changed on	10.11.1999
Development class	VS
Program name	SAPLLC01
INCLUDE name	LLC01U29
Original language	DE
Released on	22.03.1999

Figure 3-7 Attributes tab of BAPI_VENDOR_GETDETAIL[1]

This particular module is remote-enabled, so we can proceed.

3.4.4 Dissecting the module

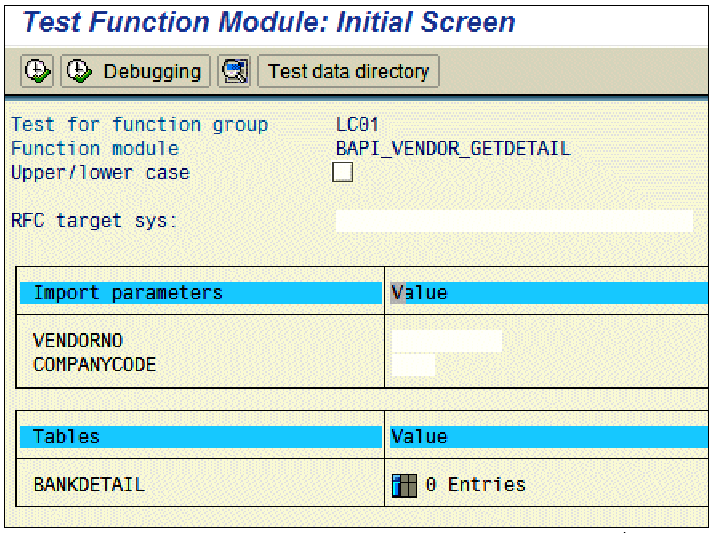
Now that we found an SAP module to return our vendor data, it is time to examine its requirements. Which parameters does it expect, and in which notation?

To find this out, we switch to the **Import** tab. This panel shows that this particular module expects two import parameters. Their names are:

- VENDORNO
- COMPANYCODE

[1] © SAP AG 2006. All rights reserved.

To examine these parameters more closely, we execute the module in the SAP GUI client by clicking the wrench symbol, resulting in the window displayed in Figure 3-8.



Test for function group	LC01
Function module	BAPI_VENDOR_GETDETAIL
Upper/lower case	☐
RFC target sys:	

Import parameters	Value
VENDORNO	
COMPANYCODE	

Tables	Value
BANKDETAIL	0 Entries

Figure 3-8 Initial screen of BAPI_VENDOR_GETDETAIL[1]

We can see that BAPI_VENDOR_GETDETAIL expects a vendor number or a company code, maybe even both at the same time, although one *or* the other sounds more reasonable. The parameters have different lengths, which is a concept that does not exist in Notes.

As Notes developers, we are used to fields of (almost) indefinite length. SAP, however, has a size property associated with every data item. In our example, the VENDORNO field is of type CHAR (meaning it accepts string values) and it has a length of 10. And unfortunately, that is exactly what the module expects.

Unfortunately? Well, usually the Lotus Notes access for SAP solutions software on our computer takes care of converting the data formats between Notes and SAP and we do not care, but sometimes there are situations that are next to impossible to anticipate. Most errors in SAP integration attempts are results of these seemingly insignificant details, like the one we just discovered. If you have the following LotusScript statement, it will fail:

```
selectstatement = |VENDORNO = "123456"|
```

You only fed the module six characters. The correct statement is:

```
selectstatement = |VENDORNO = "0000123456"|
```

Now why is that? Does the Lotus Notes access for SAP solutions software not correct this? Usually, yes. In this case, no. For some reason (that we do not know), the SAP developer who wrote the BAPI_VENDOR_GETDETAIL module chose CHAR as the data type for the VENDORNO field. If it had been NUMC instead, everything would be fine; conversion would be automatic, no intervention required. CHAR, however, is a string is a string is a string. This is not padded, clipped, or altered in any way, so we have to take care of this ourselves. Because this situation occurs from time to time, the Lotus Notes access for SAP solutions developers prepared a special function that you can use. It is in the script library SAPaccessUtilities, and its name is PadZerosLeft. For more information about this, see Chapter 5, “Troubleshooting” on page 91.

Now that we sorted this out, our next step is to look for vendor entries inside of SAP. It is useful to have a working vendor number to test our LotusScript. If you develop your code

[1] © SAP AG 2006. All rights reserved.

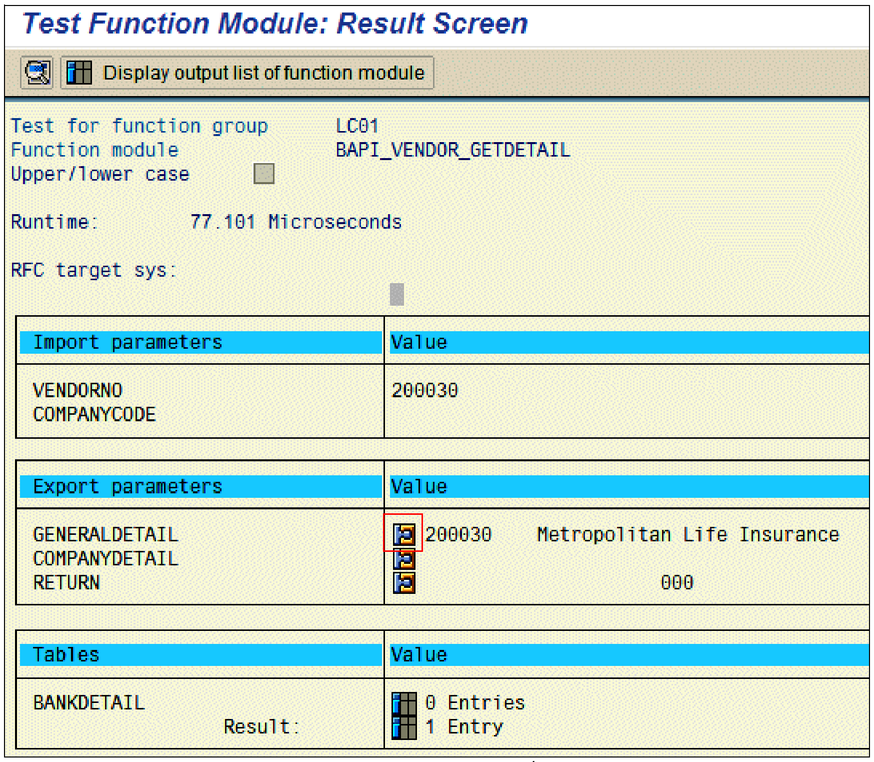
blindly, you might run into some sort of “xyz not found” error and think your code is faulty, while it is really a missing entry in some table.

We pick a random number, enter it into the VENDORNO field and click the run symbol (the clock with the green check mark, as shown in the top left corner of Figure 3-8 on page 45).

We choose the number 200030 and the resulting window shows us that it was a good choice; there is a company with that number in the SAP system. The Export Parameters section of the result window displayed in Figure 3-9 shows some company details. (Because we used the SAP GUI client, the padding from 200030 to 0000200030 was done automatically this time.)

Note: It is easy to get confused with the *import* and *export* terminology. Always consider SAP the leading system, so Notes imports are SAP exports and vice versa. While this is still a simple example, it might be a good thing to keep this advice in mind for more complex integration scenarios. This is even more important if tables/structures are used in the Imports/Exports sections instead of simple data fields, as we see later.

There is also a table called BANKDETAILS returned as part of the SAP result, but we do not go into details here because it is not necessary for the current task.


Test Function Module: Result Screen
Display output list of function module

```
Test for function group      LC01
Function module              BAPI_VENDOR_GETDETAIL
Upper/lower case

Runtime:        77.101 Microseconds

RFC target sys:
```

Import parameters	Value
VENDORNO	200030
COMPANYCODE	

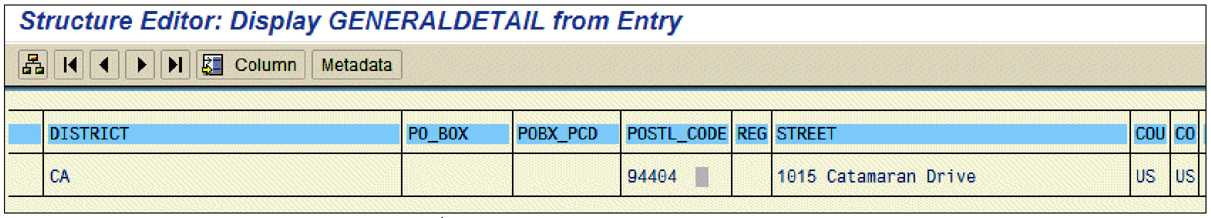
Export parameters	Value
GENERALDETAIL	200030 Metropolitan Life Insurance
COMPANYDETAIL	
RETURN	000

Tables	Value
BANKDETAIL	0 Entries
Result:	1 Entry

Figure 3-9 BAPI_VENDOR_GETDETAIL results[1]

[1] © SAP AG 2006. All rights reserved.

Clicking the **GENERALDETAIL** icon leads us to the complete data listing for vendor #200030. We might need to scroll a little to the right, but eventually we can see the vendor's address data that we want to write to our Notes contact entry (Figure 3-10).



DISTRICT	PO_BOX	POBX_PCD	POSTL_CODE	REG	STREET	COU	CO
CA			94404		1015 Catamaran Drive	US	US

Figure 3-10 Details for vendor #200030[1]

3.5 Using the LC LSX to access the BAPI

Our next step is to reproduce what we just did with the SAP GUI client and access BAPI_VENDOR_GETDETAIL in LotusScript. We use the classes described in 3.2, "Developing a road map for accessing SAP" on page 36.

We create a test agent and, always a good practice while testing with SAP, name it similarly to the remote function that it is supposed to call: BAPI_VENDOR_GETDETAILS.

Let's take a look at the *Initialize* event. We list the whole section first in Example 3-2 for your convenience and then discuss the parts next.

Example 3-2 Accessing BAPI_VENDOR_GETDETAILS in LotusScript

```
On Error Goto errorhandler
Dim session As New LCSession
Dim source As New LCConnection("SAP")
Dim sourcefieldlist As New LCFieldlist
Dim field1 As LCField
Dim field2 As LCField
Dim field3 As LCField
Dim counter As Integer
Dim selectstatement As String

source.Userid = "muster"
source.Password = "ides"
source.Client = "800"
source.SystemNo = 0
source.Language = "EN"
source.Server= "sap01.itso.ibm.com"
source.Destination = "LD3"
source.debuglevel = 0
source.Connect

source.Database = "BAPI_VENDOR_GETDETAIL"
source.Metadata = "*"
selectstatement = |VENDORNO = "0000200030"|
counter = source.Execute(selectstatement, sourcefieldlist)

If (counter = 0) Then
```

[1] © SAP AG 2006. All rights reserved.

```
      MessageBox ("No records returned by R/3")
      Exit Sub
   End If

   Dim index As Long
   While (source.Fetch(sourcefieldlist))
      Set field1 = sourcefieldlist.lookup("EXPORTSGENERALDETAILNAME",index)
      Set field2 = sourcefieldlist.lookup("EXPORTSGENERALDETAILSTREET",index)
      Set field3 = sourcefieldlist.lookup("EXPORTSGENERALDETAILCOUNTRYISO",index)
      Messagebox (field1.text(0) & field2.text(0) & field3.text(0))
   Wend

   source.Disconnect

   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

   End Sub
```

Connecting the agent to SAP

First, we define an error handler and the objects we need:

```
On Error Goto errorhandler
Dim session As New LCSession
Dim source As New LCConnection("SAP")
Dim sourcefieldlist As New LCFieldlist
Dim field1 As LCField
Dim field2 As LCField
Dim field3 As LCField
Dim counter As Integer
Dim selectstatement As String
```

Next, we connect to SAP:

```
source.Userid = "muster"
source.Password = "ides"
source.Client = "800"
source.SystemNo = 0
source.Language = "EN"
source.Server= "sap01.itso.ibm.com"
source.Destination = "LD3"
source.debuglevel = 0
source.Connect
```

The connection is made, so we tell SAP what module we want to use:

```
source.Database = "BAPI_VENDOR_GETDETAIL"
source.Metadata = "*"
```

Now that SAP knows what module we want to access, we tell it what record we are looking for:

```
selectstatement = |VENDORNO = "0000200030"|
counter = source.Execute(selectstatement, sourcefieldlist)
```

This returns a block of information. Anything there is about that specific vendor. If not, we just leave.

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

Integrating the data into Domino

If everything went as planned, we have a result set in our LCFieldlist object that contains the values we want to process. Next, we need to filter the results to isolate the data we want to show to the user and the data we need for background processing only:

```
Dim index As Long
While (source.Fetch(sourcefieldlist))
   Set field1 = sourcefieldlist.lookup("EXPORTSGENERALDETAILNAME",index)
   Set field2 = sourcefieldlist.lookup("EXPORTSGENERALDETAILSTREET",index)
   Set field3 = sourcefieldlist.lookup("EXPORTSGENERALDETAILCOUNTRYISO",index)
   Messagebox (field1.text(0) & field2.text(0) & field3.text(0))
Wend
```

Closing the connection

Never forget to close your connection to SAP:

```
source.Disconnect
```

Error handling

If the script did not do what you expected it to, it is time to do some error handling:

```
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

End Sub
```

Summary

As you can see, this was just LotusScript, no magic here. The complicated parts are just how to assemble the metadata for SAP and the appropriate filter to be applied to the result set. Let's take a look at this in more detail.

3.5.1 SAP metadata

When connecting to an RDBMS, you usually set the *Database* and *Metadata* properties of your LCConnection object to the names of the database and table you want to access. This is different when using SAP. The Database property corresponds to the module you want to call ("BAPI_VENDOR_GETDETAIL" in this example), and what is table information in an RDMBS call becomes a simple all-or-nothing switch in most cases. If you want your target module to return everything there is, you set the Metadata property to "*" (which is what you will be doing most of the time). If not, you set it to a meaningful value that depends on the particular module you are using. (Always set it to the respective table name when using RFC_READ_TABLE.)

3.5.2 Finding the right table to use

After having successfully accessed BAPI_VENDOR_GETDETAIL, we find that we can now fetch a vendor's data using its vendor number. But for our intended use, we will not have the vendor number right away; we will only have the vendor name. We need an intermediate step that helps us find the vendor number using the vendor name. As developers, we now immediately think of the terms *mapping* and *table*.

We need a table that holds both the vendor name and its number. To find out where this might be stored, we use the SAP GUI client again. SAP uses tables, so there must be a reference somewhere in the BAPI_VENDOR_GETDETAIL module. We just have to find it.

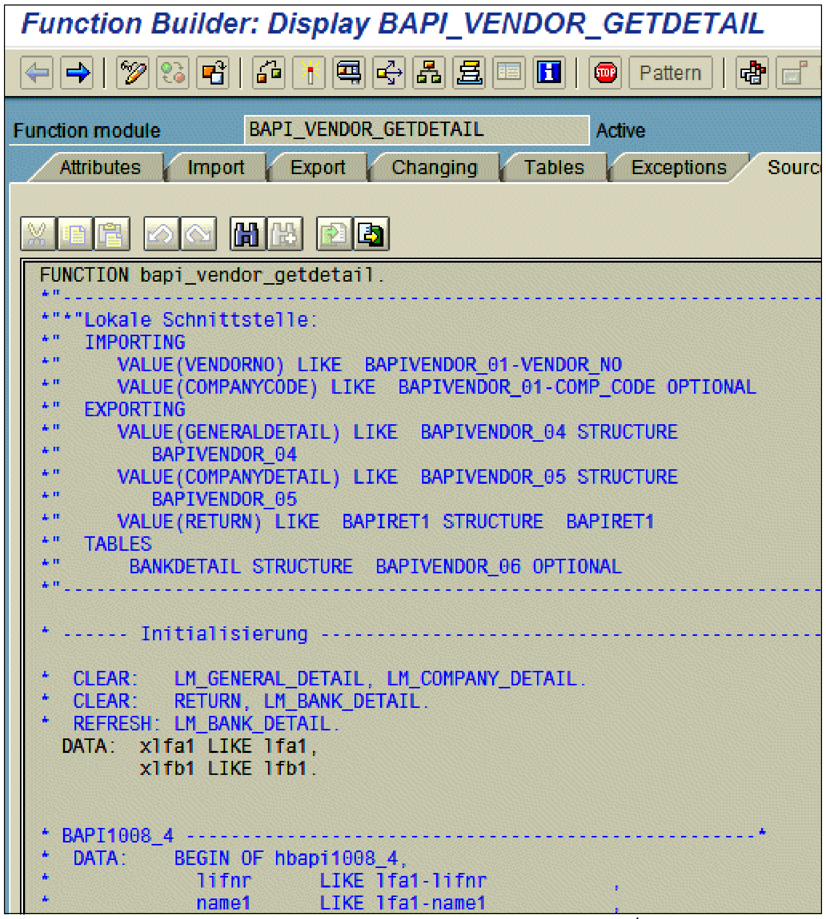
When we look at the module again in SE37, we notice a tab called Tables, but this only holds a reference to the BANKDETAILS table, which is a part of the result set.

Next, we check the function module documentation by clicking the respective button on the top right of the window. Again, this is not what we want.

It is time to drill down into the system a little more. One possible approach is to take a look at the source code. Another one is to explore the structures involved. We take a peek at both methods now.

Method A: Trying to interpret the source code

Let's switch to the **Source Code** tab and see what is written there (Figure 3-11 on page 51). Even if you are not an ABAP programmer, there might still be some hints there to help you find your way. Most of the time you do not have to look at the code itself—examining the comments is often sufficient. (Sometimes it also helps if you speak a little German. Many comments are written in that language, as well as some variables and table names. After all, SAP is a German company.)



```
FUNCTION bapi_vendor_getdetail.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(VENDORNO) LIKE  BAPIVENDOR_01-VENDOR_NO
*"     VALUE(COMPANYCODE) LIKE  BAPIVENDOR_01-COMP_CODE OPTIONAL
*"  EXPORTING
*"     VALUE(GENERALDETAIL) LIKE  BAPIVENDOR_04 STRUCTURE
*"        BAPIVENDOR_04
*"     VALUE(COMPANYDETAIL) LIKE  BAPIVENDOR_05 STRUCTURE
*"        BAPIVENDOR_05
*"     VALUE(RETURN) LIKE  BAPIRET1 STRUCTURE  BAPIRET1
*"  TABLES
*"      BANKDETAIL STRUCTURE  BAPIVENDOR_06 OPTIONAL
*"----------------------------------------------------------------------

* ------ Initialisierung -----------------------------------------------

*  CLEAR:   LM_GENERAL_DETAIL, LM_COMPANY_DETAIL.
*  CLEAR:   RETURN, LM_BANK_DETAIL.
*  REFRESH: LM_BANK_DETAIL.
  DATA:  xlfa1 LIKE lfa1,
         xlfb1 LIKE lfb1.


* BAPI1008_4 ---------------------------------------------------*
*  DATA:    BEGIN OF hbapi1008_4,
*             lifnr      LIKE lfa1-lifnr           ,
*             name1      LIKE lfa1-name1           ,
```

Figure 3-11 Source code of BAPI_VENDOR_GETDETAIL[1]

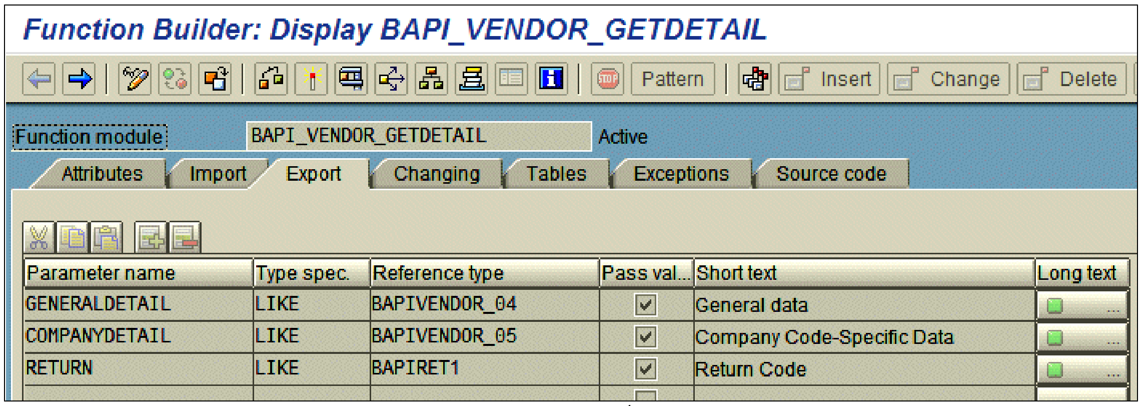
The first block of the source code is a comment that documents the local interface (“Lokale Schnittstelle” in German) of the code. Although it does not say anything about tables, at least it lists the names of some structures.

Structures are complex SAP data types that can hold virtually anything: tables, data fields, and even other structures. So although we might not have found the correct table itself, we might have found its container. The structures BAPIVENDOR_04 and BAPIVENDOR_05 look particularly interesting because they seem to hold values described as GENERALDETAIL (which contained the address data we found earlier) and COMPANYDETAIL. To explore these structures, we open the **Export** tab. From now on, follow the steps under method B in the following section.

[1] © SAP AG 2006. All rights reserved.

Method B: Explore the structures

We can also skip the scrutinizing of the module's source code by taking a click-by-click approach. There is a good chance that the missing information is to be found somewhere on the Export tab, because that is where our results come from. Therefore, we open the **Export** tab and see what is in there (Figure 3-12).



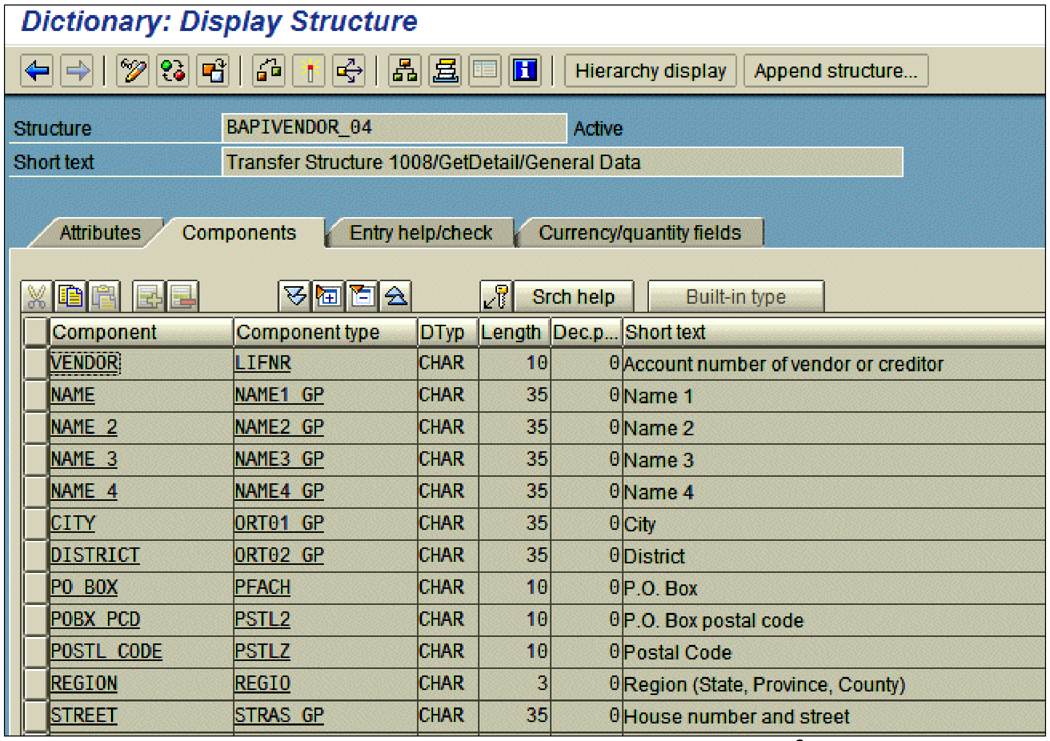
Function module BAPI_VENDOR_GETDETAIL Active

Parameter name	Type spec.	Reference type	Pass val...	Short text	Long text
GENERALDETAIL	LIKE	BAPIVENDOR_04	✓	General data	🟩 ...
COMPANYDETAIL	LIKE	BAPIVENDOR_05	✓	Company Code-Specific Data	🟩 ...
RETURN	LIKE	BAPIRET1	✓	Return Code	🟩 ...

Figure 3-12 Export tab of BAPI_VENDOR_GETDETAILS[1]

The first line tells us something along the lines of GENERALDETAIL is LIKE the structure BAPIVENDOR_04, so we double-click that structure's name.

This takes us to the SAP Data Dictionary (shortcut SE11), where all data structures are centrally defined and stored for any given SAP system. The selected structure opens on the Components tab (Figure 3-13).



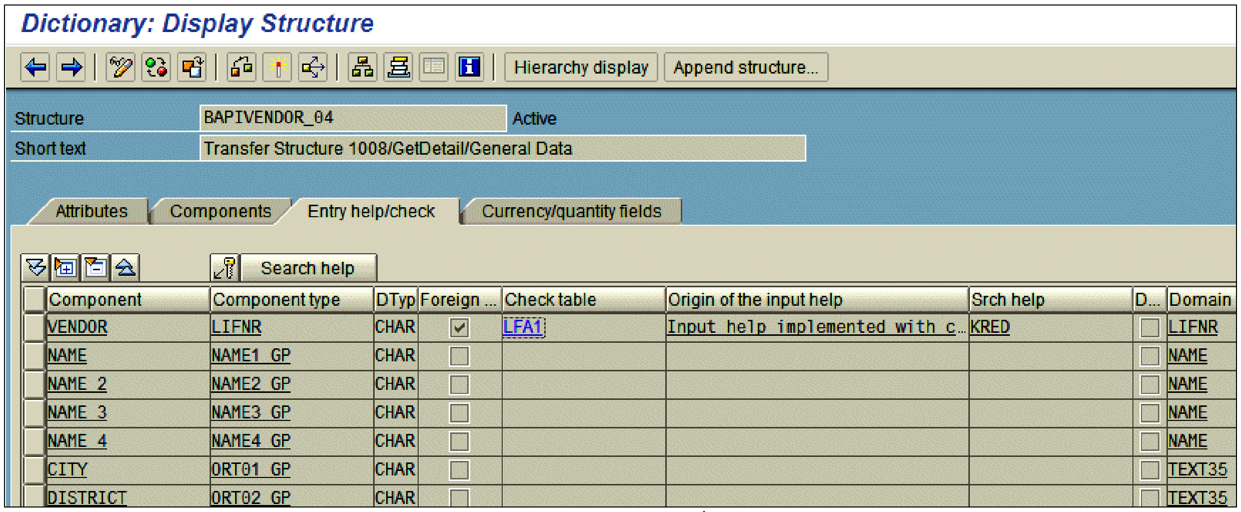
Structure BAPIVENDOR_04 Active
Short text Transfer Structure 1008/GetDetail/General Data

Component	Component type	DTyp	Length	Dec.p...	Short text
VENDOR	LIFNR	CHAR	10	0	Account number of vendor or creditor
NAME	NAME1_GP	CHAR	35	0	Name 1
NAME_2	NAME2_GP	CHAR	35	0	Name 2
NAME_3	NAME3_GP	CHAR	35	0	Name 3
NAME_4	NAME4_GP	CHAR	35	0	Name 4
CITY	ORT01_GP	CHAR	35	0	City
DISTRICT	ORT02_GP	CHAR	35	0	District
PO_BOX	PFACH	CHAR	10	0	P.O. Box
POBX_PCD	PSTL2	CHAR	10	0	P.O. Box postal code
POSTL_CODE	PSTLZ	CHAR	10	0	Postal Code
REGION	REGIO	CHAR	3	0	Region (State, Province, County)
STREET	STRAS_GP	CHAR	35	0	House number and street

Figure 3-13 Data Dictionary Components for structure BAPIVENDOR_04[2]

[1] © SAP AG 2006. All rights reserved.
[2] © SAP AG 2006. All rights reserved.

We seem to be on the right path. The components listed here look familiar; these are the table headings we saw before in the SAP GUI client. Therefore, we need to check the other tabs for table names. The Attributes and Currency tabs show no tables. But the Entry help/check tab has a column called Check table (Figure 3-14).

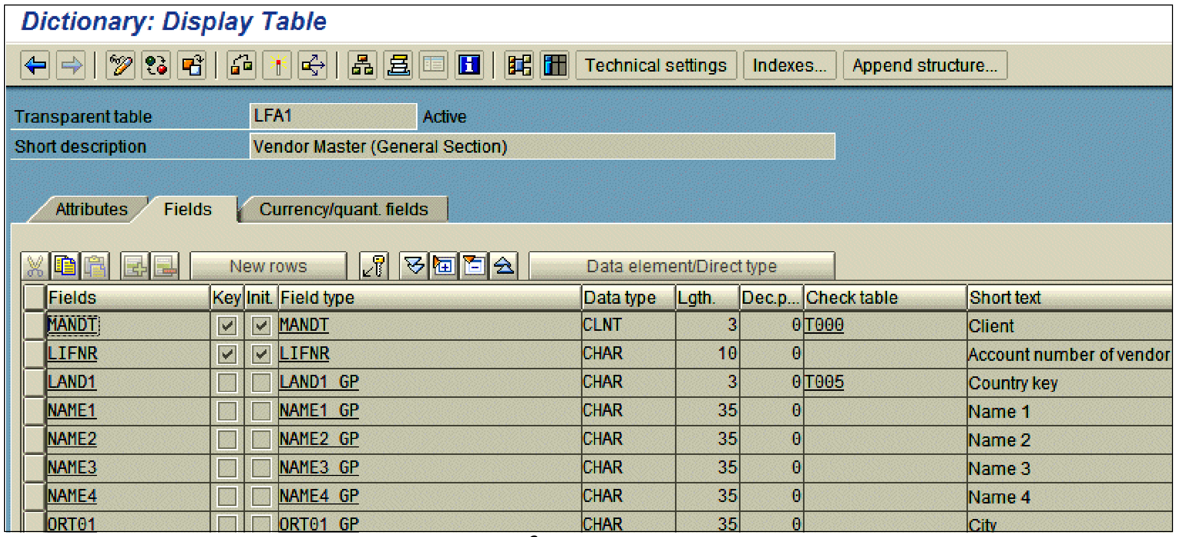


Structure BAPIVENDOR_04 Active
Short text Transfer Structure 1008/GetDetail/General Data

Component	Component type	DTyp	Foreign ...	Check table	Origin of the input help	Srch help	D...	Domain
VENDOR	LIFNR	CHAR	✓	LFA1	Input help implemented with c...	KRED	☐	LIFNR
NAME	NAME1_GP	CHAR	☐				☐	NAME
NAME_2	NAME2_GP	CHAR	☐				☐	NAME
NAME_3	NAME3_GP	CHAR	☐				☐	NAME
NAME_4	NAME4_GP	CHAR	☐				☐	NAME
CITY	ORT01_GP	CHAR	☐				☐	TEXT35
DISTRICT	ORT02_GP	CHAR	☐				☐	TEXT35

Figure 3-14 Entry help/check tab of structure BAPIVENDOR_04[1]

In the first line of that column, we find an entry called LFA1, which is highlighted in blue. Double-click this table name to go to the corresponding Data Dictionary entry (Figure 3-15). A check of the Fields tab tells us that this table contains the data we need, complete with the respective data types.



Transparent table LFA1 Active
Short description Vendor Master (General Section)

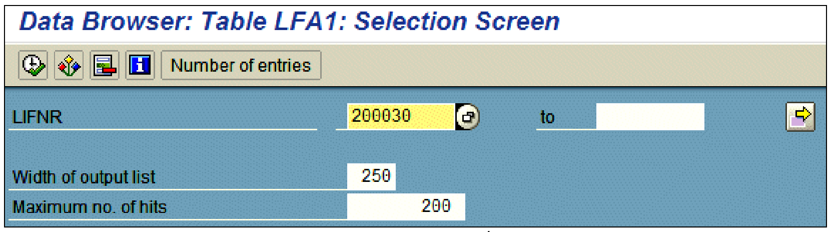
Fields	Key	Init.	Field type	Data type	Lgth.	Dec.p...	Check table	Short text
MANDT	✓	✓	MANDT	CLNT	3	0	T000	Client
LIFNR	✓	✓	LIFNR	CHAR	10	0		Account number of vendor
LAND1	☐	☐	LAND1_GP	CHAR	3	0	T005	Country key
NAME1	☐	☐	NAME1_GP	CHAR	35	0		Name 1
NAME2	☐	☐	NAME2_GP	CHAR	35	0		Name 2
NAME3	☐	☐	NAME3_GP	CHAR	35	0		Name 3
NAME4	☐	☐	NAME4_GP	CHAR	35	0		Name 4
ORT01	☐	☐	ORT01_GP	CHAR	35	0		City

Figure 3-15 Data Dictionary entry for table LFA1[2]

Now that we know how the vendor data is organized and where it is stored, we have to check if the table contains our example entry #200030. (So far, we only know the structure of the table, but not its contents.)

[1] © SAP AG 2006. All rights reserved.
[2] © SAP AG 2006. All rights reserved.

For this task, use the Data Browser (shortcut SE16) in the SAP GUI client (Figure 3-16). This tool (among others) enables you to peruse tables.

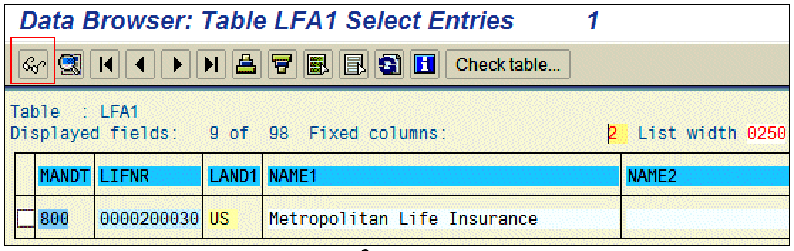

Data Browser: Table LFA1: Selection Screen

Number of entries

LIFNR	200030	to	
Width of output list	250		
Maximum no. of hits	200		

Figure 3-16 Data Browser Selection Screen[1]

We enter our vendor number into the LIFNR field (LIFNR is an abbreviation for Lieferantennummer, which is German for vendor number), click the clock symbol, and get the following result window (Figure 3-17).


Data Browser: Table LFA1 Select Entries 1

Check table...

```
Table   : LFA1
Displayed fields:   9 of  98  Fixed columns:              2  List width 0250
```

MANDT	LIFNR	LAND1	NAME1	NAME2
800	0000200030	US	Metropolitan Life Insurance	

Figure 3-17 Data Browser result[2]

We found vendor number 200030 in table LFA1. Scrolling to the right tells us that the other address information pertaining to this particular vendor is also there. To get all the details and field names, check the box to the left of the table entry and click the display icon (glasses). Now you can see all data about this vendor (Figure 3-18).

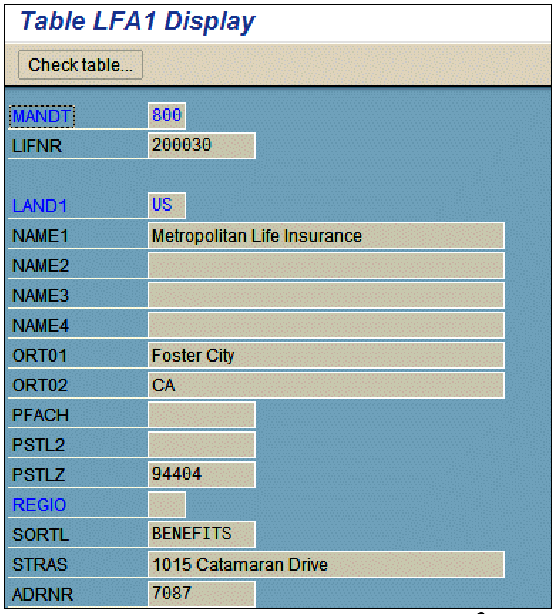


Table LFA1 Display

Check table...

MANDT	800
LIFNR	200030
LAND1	US
NAME1	Metropolitan Life Insurance
NAME2	
NAME3	
NAME4	
ORT01	Foster City
ORT02	CA
PFACH	
PSTL2	
PSTLZ	94404
REGIO	
SORTL	BENEFITS
STRAS	1015 Catamaran Drive
ADRNR	7087

Figure 3-18 Address detail from table LFA1[3]

[1] © SAP AG 2006. All rights reserved.
[2] © SAP AG 2006. All rights reserved.
[3] © SAP AG 2006. All rights reserved.

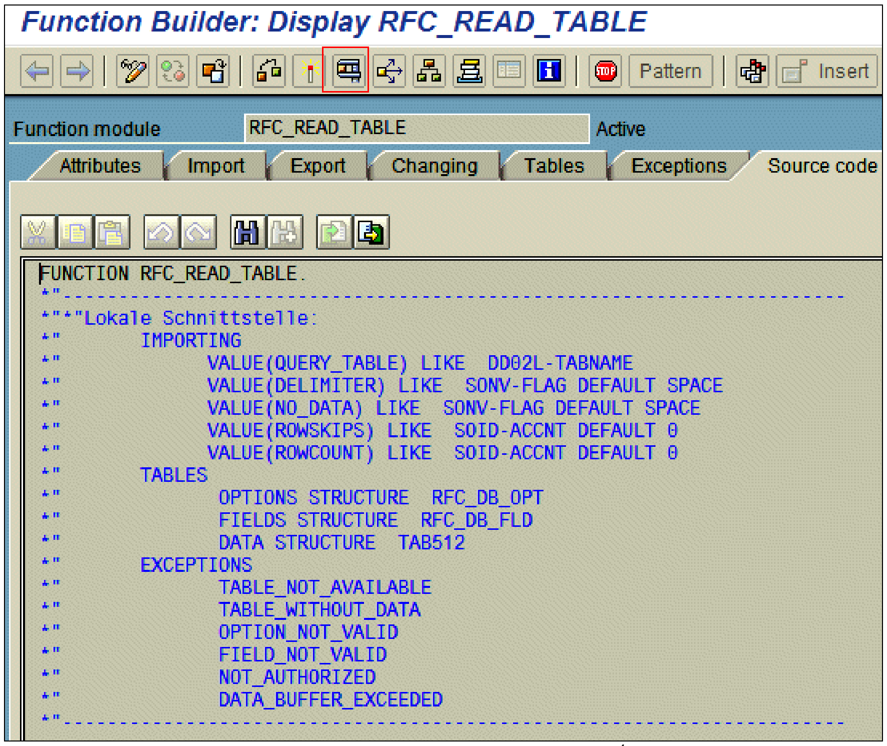
We found the right place to populate our list of vendors to present in the UI. The next step in the process for us is to read a few columns from table LFA1 in LotusScript and build a list of choices from them. This list is then presented to the users in a later step.

3.5.3 Searching SAP for vendors

To get the list of vendors for our UI, we have two promising options:

- We can use a module called BAPI_VENDOR_FIND to find a list of vendors along with their associated vendor numbers. When the user clicks one of them in the list, we call BAPI_VENDOR_GETDETAIL to retrieve further data such as street address and city.
- We can also use a standard function module provided by SAP called RFC_READ_TABLE, which is one of the most commonly used modules in SAP integration. It is well-known and well-documented; you can find anything about it by searching for its name in any major Web search engine. With RFC_READ_TABLE, we can read vendor number, vendor name, and the address data all at once, thus saving the second lookup with BAPI_VENDOR_GETDETAIL, so we opt for this approach.

Let's take a look at RFC_READ_TABLE in the ABAP function builder of the SAP GUI client. We open up the module in SE37 and examine its features (Figure 3-19).



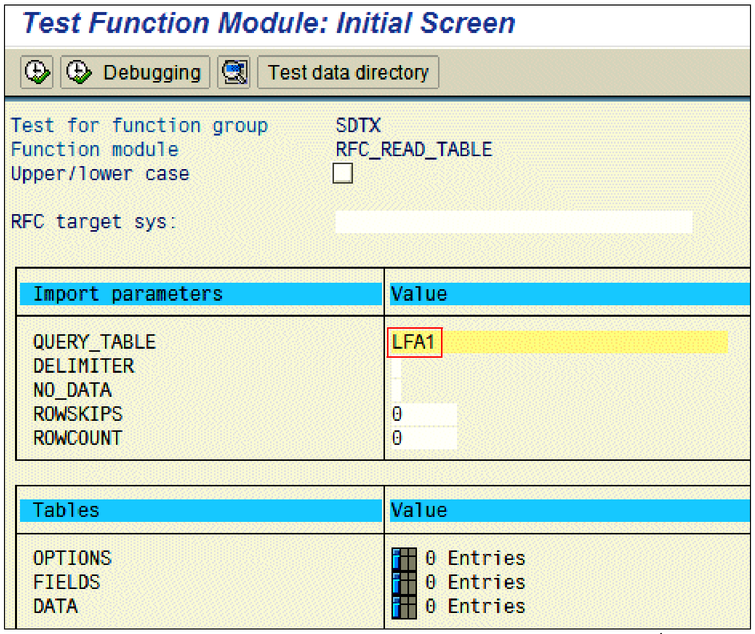
```
FUNCTION RFC_READ_TABLE.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"             VALUE(QUERY_TABLE) LIKE  DD02L-TABNAME
*"             VALUE(DELIMITER) LIKE  SONV-FLAG DEFAULT SPACE
*"             VALUE(NO_DATA) LIKE  SONV-FLAG DEFAULT SPACE
*"             VALUE(ROWSKIPS) LIKE  SOID-ACCNT DEFAULT 0
*"             VALUE(ROWCOUNT) LIKE  SOID-ACCNT DEFAULT 0
*"       TABLES
*"              OPTIONS STRUCTURE  RFC_DB_OPT
*"              FIELDS STRUCTURE  RFC_DB_FLD
*"              DATA STRUCTURE  TAB512
*"       EXCEPTIONS
*"              TABLE_NOT_AVAILABLE
*"              TABLE_WITHOUT_DATA
*"              OPTION_NOT_VALID
*"              FIELD_NOT_VALID
*"              NOT_AUTHORIZED
*"              DATA_BUFFER_EXCEEDED
*"----------------------------------------------------------------------
```

Figure 3-19 Source code snippet of RFC_READ_TABLE[1]

We find three headings in the starting comments block of RFC_READ_TABLE's source code: IMPORTING, TABLES, and EXCEPTIONS. There seem to be no EXPORTS.

[1] © SAP AG 2006. All rights reserved.

This is because SAP tables can work both ways. This module is a very good example of how this two-way approach works. But to save a lengthy explanation, let's just run RFC_READ_TABLE now by clicking the wrench symbol once again, resulting in the window shown in Figure 3-20.


Test Function Module: Initial Screen

Test for function group	SDTX
Function module	RFC_READ_TABLE
Upper/lower case	☐
RFC target sys:	

Import parameters	Value
QUERY_TABLE	LFA1
DELIMITER	
NO_DATA	
ROWSKIPS	0
ROWCOUNT	0

Tables	Value
OPTIONS	0 Entries
FIELDS	0 Entries
DATA	0 Entries

Figure 3-20 Running RFC_READ_TABLE in function builder[1]

The import parameter QUERY_TABLE determines which table we want to access with our module, so we enter `LFA1` (and ignore the rest of the imports for our example.) If we ran this query now, we would receive a DATA_BUFFER_EXCEEDED exception from SAP because it would try to return the entire table LFA1, and that is a lot more than the DATA table can hold at once (512 bytes).

This is not what we want anyway, so we need to find a way to set the level of detail we want from SAP first. First, we limit the number of fields to be contained in our result set; just the vendor name and a few address details will do. In SE16, the column headings represent the field names, so we use these. See Figure 3-18 on page 54 for the exact names.

The best place to restrict the number of fields returned seems to be the FIELDS table shown in the lower section of the window. We click the table symbol and get to the details window for the selected table. Here, we enter the field names we want to make part of the result set. For this example, we use four fields (click the **New Line** button after entering each one).

Now, we click the green **Back** Icon to get back to the previous page. We can see that the FIELDS table contains four entries now. After we execute the module by clicking the clock symbol, we notice that the DATA table suddenly contains a lot of entries. When we click the DATA's table symbol, we are shown a list of vendor records.

SAP has obviously written the results of our query to the DATA table, while we have entered our field filter in the FIELDS table. This is a big difference in concepts compared to what we have seen so far. The tables existed before and after query execution, while regular export parameters only come to life as part of the result set. The FIELDS table was even modified during query execution if we take a closer look. SAP has added the field descriptions of the four fields we selected for the result set.

[1] © SAP AG 2006. All rights reserved.

Now that we know the way RFC_READ_TABLE works, let's do everything again in LotusScript. As usual, we first show you the code in Example 3-3 and then walk it through in detail.

Example 3-3 Example function to call RFC_READ_TABLE in LotusScript

```
Sub searchVendors (thisdoc As NotesDocument)

   Dim session As New LCSession
   Dim source As New LCConnection("SAP")
   Dim sourcefieldlist As New LCFieldlist
   Dim counter As Integer
   Dim selectstatement As String
   Dim retval As Integer

   source.Userid = "muster"
   source.Password = "ides"
   source.Client = "800"
   source.SystemNo = 0
   source.Language = "EN"
   source.Server= "sap01.itso.ibm.com"
   source.Destination = "LD3"

   source.Connect

   thisdoc.T_Vendor = thisdoc.GetItemValue("SearchVendorName")
   thisdoc.TL_VENDORID = ""
   thisdoc.TL_VENDORNAME = ""
   thisdoc.TL_COUNTRY = ""
   thisdoc.TL_CITY = ""

   source.Database = "RFC_READ_TABLELFA1"
   source.Metadata = "LFA1"

   selectstatement = +_
   |FIELDS.1.FIELDNAME="LIFNR",| +_
   |FIELDS.2.FIELDNAME="NAME1",| +_
   |FIELDS.3.FIELDNAME="ORT01",| +_
   |FIELDS.4.FIELDNAME="LAND1",| +_
   |OPTIONS.1.TEXT="NAME1 LIKE '| + thisdoc.searchvendorname(0) +|%'"|

   counter = source.Execute(selectstatement, sourcefieldlist)

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

   Dim field1 As LCField
   Dim field2 As LCField
   Dim field3 As LCField
   Dim field4 As LCField

   Set field1 = sourcefieldlist.lookup("LFA1LIFNR", 1)
   Set field2 = sourcefieldlist.lookup("LFA1NAME1", 1)
   Set field3 = sourcefieldlist.lookup("LFA1ORT01", 1)
   Set field4 = sourcefieldlist.lookup("LFA1LAND1", 1)

   Dim i As Long
   i = 1
   While (source.Fetch(sourcefieldlist))
      thisdoc.TL_VENDORID = thisdoc.TL_VENDORID(0) + "$" +  field1.text(0)
      thisdoc.TL_VENDORNAME = thisdoc.TL_VENDORNAME(0) + "$" +  field2.text(0)
      thisdoc.TL_COUNTRY = thisdoc.TL_COUNTRY(0) + "$" +  field3.text(0)
      thisdoc.TL_CITY = thisdoc.TL_CITY(0) + "$" +  field4.text(0)
      i = i + 1
   Wend

      thisdoc.Vendors = ""
   thisdoc.NoResultTextE = "<No results found>"

End Sub
```

We call this piece of code from the Search button on the Domino form if the Search radio button is set to vendors. First, we define the objects and variables needed and connect to the SAP system (Example 3-4).

Example 3-4 Defining objects and variables to connect to our SAP system

```
Dim session As New LCSession
   Dim source As New LCConnection("SAP")
   Dim sourcefieldlist As New LCFieldlist
   Dim counter As Integer
   Dim selectstatement As String
   Dim retval As Integer

   source.Userid = "muster"
   source.Password = "ides"
   source.Client = "800"
   source.SystemNo = 0
   source.Language = "EN"
   source.Server= "sap01.itso.ibm.com"
   source.Destination = "LD3"

   source.Connect
```

Next, we create fields in our Notes document that we use to store the fields from SAP. The document we use for this purpose is the back-end NotesDocument object connected to the currently open NotesUIDocument.

How you do your own implementation of this is up to you. As we see in Chapter 4, “Extending” on page 63 when we go into the details of the Notes access for SAP solutions script libraries, the Lotus Notes access for SAP solutions template developers have their own way of handling the data.

Example 3-5 Creating fields in our Notes document

```
thisdoc.T_Vendor = thisdoc.GetItemValue("SearchVendorName")
thisdoc.TL_VENDORID = ""
thisdoc.TL_VENDORNAME = ""
thisdoc.TL_COUNTRY = ""
thisdoc.TL_CITY = ""
```

One word about naming conventions used here. We use prefixes to indicate the type of data stored in a field. See Table 3-1 for details. (Do not feel bound by them, however; they are completely arbitrary.)

Table 3-1 Data type prefixes used in the examples

Prefix	**Data type**
T_	Text Value
TL_	Text List
NL_	Number List
DL_	Date List

Now we set our metadata details and fetch the data from SAP (Example 3-6).

Example 3-6 Moving values to Notes

```
source.Database = "RFC_READ_TABLELFA1"
source.Metadata = "LFA1"

selectstatement = +_
|FIELDS.1.FIELDNAME="LIFNR",| +_
|FIELDS.2.FIELDNAME="NAME1",| +_
|FIELDS.3.FIELDNAME="ORT01",| +_
|FIELDS.4.FIELDNAME="LAND1",| +_
|OPTIONS.1.TEXT="NAME1 LIKE '| + thisdoc.searchvendorname(0) +|%'"|

counter = source.Execute(selectstatement, sourcefieldlist)

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

Important: Note that the LCConnection.Database and LCConnection.Metadata properties are different from our other examples. The former equals the name of the module directly followed by the name of the table that is to be accessed, the latter is set to the name of the table in question.

This is a special case due to the architecture of the SAP connector software that Lotus Notes access for SAP solutions uses. Usually when accessing an SAP system, you call RFC modules, BAPIs, or transactions. These are usually only very remotely comparable to SQL calls.

Structured Query Language (SQL), as the name implies, is a standard that behaves in known ways. If you try to INSERT new data into a table, and there is a problem, the insert action fails. It is as simple as that.

RFCs, however, are written by thousands of different people, each going their own way (not only) regarding error handling. If you try to send new data to SAP (INSERT), and there is a problem, then maybe:

- The call of the RFC throws an exception.
- The call of the RFC return san EXPORT with failure/error information. This is more common than exceptions.

If you call an RFC and there is a problem, most times the call itself succeeds. It is up to your application to examine the result set to determine what really happened on the way. There is no standardized way to do this.

RFC_READ_TABLE is a little different because you can use it almost like an SQL call. Append the table name to the RFC name, set your fields and options, and then call the SAP system. You basically do something similar to this wrapped in a Lotus Connector call:

```
SELECT <your FIELDS> FROM <SAP tablename> WHERE <your OPTIONS>
```

This makes for a good multipurpose tool in SAP integration, if all else fails. Just do not get into the habit. Tables alone sometimes are not very meaningful without the SAP application logic on top.

The selectstatement variable holds the values of the FIELDS and OPTIONS tables that are part of the RFC module to filter the result set. FIELDS holds the four fields that we want to be included in the result set, and OPTIONS brings in our search term, further limiting the result set to matching records.

After the statement's execution, we have the results in our LCFieldlist. Now we need to get the values over to Notes (Example 3-7).

Example 3-7

```
   Dim field1 As LCField
   Dim field2 As LCField
   Dim field3 As LCField
   Dim field4 As LCField

   Set field1 = sourcefieldlist.lookup("LFA1LIFNR", 1)
   Set field2 = sourcefieldlist.lookup("LFA1NAME1", 1)
   Set field3 = sourcefieldlist.lookup("LFA1ORT01", 1)
   Set field4 = sourcefieldlist.lookup("LFA1LAND1", 1)

   Dim i As Long
   i = 1
   While (source.Fetch(sourcefieldlist))
      thisdoc.TL_VENDORID = thisdoc.TL_VENDORID(0) + "," +  field1.text(0)
      thisdoc.TL_VENDORNAME =  thisdoc.TL_VENDORNAME(0) + "," +  field2.text(0)
      thisdoc.TL_CITY = thisdoc.TL_CITY(0) + "," +  field3.text(0)
      thisdoc.TL_COUNTRY = thisdoc.TL_COUNTRY(0) + "," +  field4.text(0)
      i = i + 1
   Wend

      thisdoc.Vendors = ""
   thisdoc.NoResultTextE = "<No results found>"

End Sub
```

As usual, we loop through the result set and write the SAP values to the different fields in our Notes document. After the loop is through, we have the following situation, as shown in Table 3-2.

Table 3-2 Contents of Notes fields after SAP import

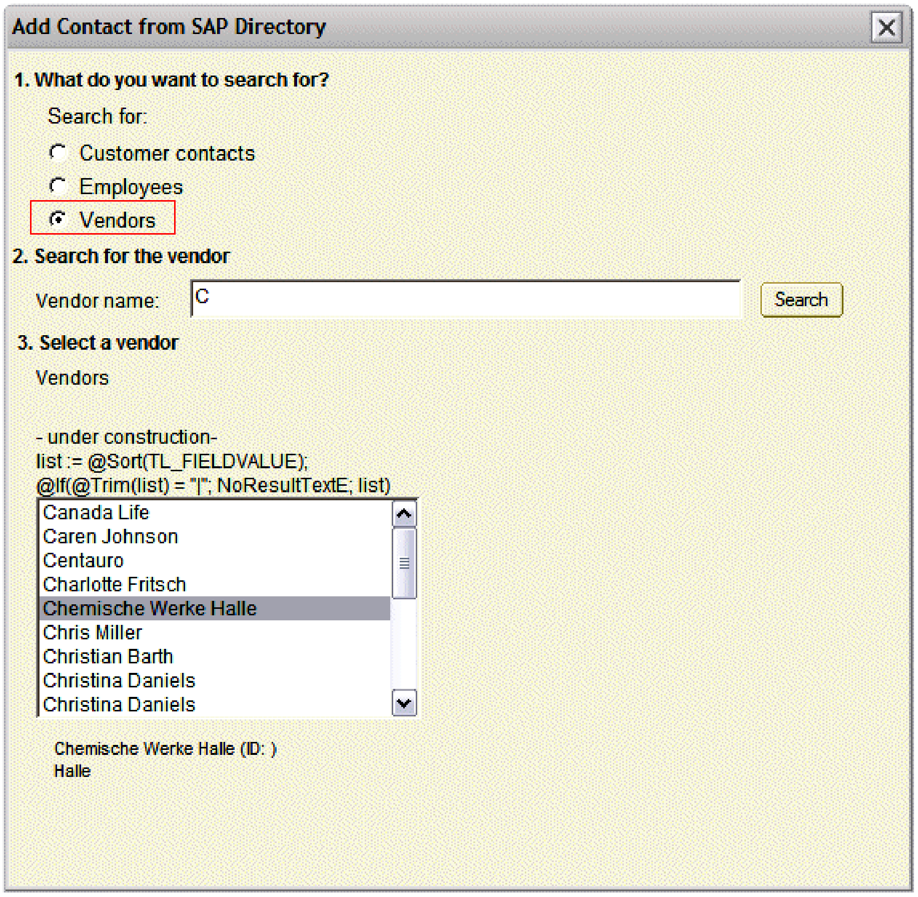
Field name	Contents	Example
TL_VENDORID	Vendor codes, separated by commas (string)	00000100, 000200030
TL_VENDORNAME	Vendor names, separated by commas (string)	IBM, ITSO
TL_CITY	City names, separated by commas (string)	Austin, Hamburg
TL_COUNTRY	Country codes, separated by commas (string)	us, de

Now we have all our important vendor data sorted into Notes multi-value fields. We can access a single set of vendor data by its position inside these fields, so the fields necessary to display one vendor's details on the Vendors tab of the Notes form will look similar to those in Table 3-3.

Table 3-3 Fields necessary for selecting/displaying vendors

Field name	Type	Formula
Vendors	List box → Use formula for choices, single-value	list := @Sort(@Explode(TL_VENDORNAME; "$")); @If(@Trim(list) = "I"; NoResultTextE; list)
VendorName	Computed for display, single-value	idx := @Member(Vendors; @Explode(TL_VENDORNAME; "$")); @If(@IsError((@Explode(TL_VENDORNAME; "$")) [idx]); "";(@Explode(TL_VENDORNAME; "$")) [idx])
VendorNumber	Computed for display, single-value	idx := @Member(Vendors; @Explode(TL_VENDORNAME; ",")); @If(@IsError((@Explode(TL_VENDORID; ",")) [idx]); "";(@Explode(TL_VENDORID; ",")) [idx])
VendorCountry	Computed for display, single-value	idx := @Member(Vendors; @Explode(TL_VENDORNAME; ",")); @If(@IsError((@Explode(TL_COUNTRY; ",")) [idx]); "";(@Explode(TL_COUNTRY; ",")) [idx])

Lets take a brief look at how these fields work together (Figure 3-21). When we click one of the entries in the list box, the index value (idx) of that entry gets calculated. This index value is then used to return the elements in the idx positions from the comma-separated text strings. This process is repeated each time you click one of the vendor names in the list box.



Add Contact from SAP Directory

1. What do you want to search for?

Search for:

- Customer contacts
- Employees
- [✓ Vendors]

2. Search for the vendor

Vendor name: C

3. Select a vendor

Vendors

- under construction-
list := @Sort(TL_FIELDVALUE);
@If(@Trim(list) = "|"; NoResultTextE; list)

Canada Life
Caren Johnson
Centauro
Charlotte Fritsch
Chemische Werke Halle
Chris Miller
Christian Barth
Christina Daniels
Christina Daniels

Chemische Werke Halle (ID:)
Halle

Figure 3-21 Vendor list in Notes client (draft)

If you have made it up to here, it should not be a problem to develop the code that is needed to add the vendor's data to the user's personal address book.

We developed all this code as a pure exercise. In general, the vendor document that needs to be created needs more information. This information is relatively easy to obtain. You have the vendor's ID that you can feed to BAPI_VENDOR_GETDETAIL. The result set from that query needs to be split onto the fields of the (Contact | Person) form in the user's personal address book. We leave that up to the you though, because it is a repetition of what we already did in this chapter. You can always download the full code from the Redbooks FTP site, as explained in Appendix B, "Additional material" on page 113.

4

Extending

In Chapter 3, “Customization” on page 35, we customized the Lotus Notes access for SAP solutions template for the personal address book completely from scratch for ITSO Bank. We had three main goals in mind:

- Developing a road map (a systematic approach) to simplify SAP/Notes integration
- Determining how customizable the Lotus Notes access for SAP solutions templates are
- Learning the basics of how to use the LC LSX classes of LotusScript

We found several steps to integrate SAP data into Notes, and it can be easy to lose direction at first without any guide or road map to follow during development. We also found that the template is easily customizable. Our third conclusion was that, for the most part, all we need to know is how to use the LCConnection, LCFieldlist, and LCField classes of the LC LSX architecture to write LotusScript code for accessing the SAP data.

In this chapter, we focus on how to use the script libraries that ship with the Lotus Notes access for SAP solutions templates. And then, with the functions available in these libraries, we show how easily you can develop your own SAP/Notes applications independently of the Lotus Notes access for SAP solutions templates by simply copying the required script libraries and using them in your own applications.

We rebuild the ITSO Bank vendor example as an exercise to help you understand the basic structure and layout of the script libraries in the Lotus Notes access for SAP solutions templates. We want to demonstrate how easy it is to extend these libraries in your own applications using our developmental road map as an overall guide for access and integrating your Notes database templates and databases with SAP. Therefore, our big goal is to show that with these templates we now have the tools to easily build any SAP/Notes integrated application.

4.1 Understanding Notes access for SAP solutions script libraries

Because we assume you are somewhat familiar with the vendors example developed in Chapter 3, “Customization” on page 35, we use it as an initial guide to walk through the functions available in the script libraries of the Lotus Notes access for SAP solutions templates. By focusing on this example, we can stress the elements of the script libraries without having to go into the details of describing a different application. Our first goal in this chapter is to convey the basic building blocks of the libraries and how capable they are for developing most of what would ever be needed for a Notes/SAP integration based on BAPI calls.

Reviewing our road map, we see that we have four basic steps:

1. Connect to SAP. However, this time, we do not have to develop this on our own. Instead, we use the function calls already available in the Notes access for SAP solutions script libraries.
2. Determine our SAP BAPIs or tables (we already did this in Chapter 3, “Customization” on page 35, so this part is already done).
3. Describe how to call the BAPIs through the functions available in the script libraries rather than directly through the LCConnection, LCFieldist, and LCField classes, as in Chapter 3, “Customization” on page 35.
4. Link the data retrieved from SAP into our UI. Our focus here is describing the structure in which the data is returned from the script libraries and how to handle that in your UI elements.

We follow these steps again in developing our application. At some points, we also demonstrate how these libraries can easily be copied into another non-Lotus Notes access for SAP solutions database and used in any SAP/Notes application.

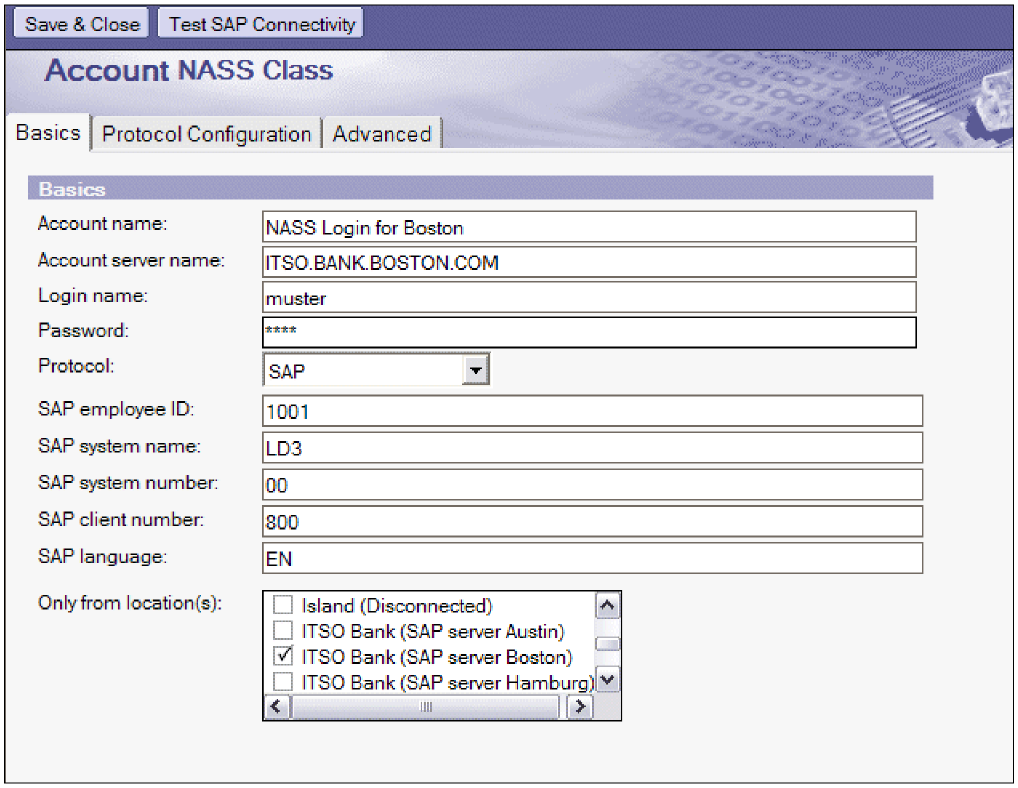
4.2 Step 1: Connecting to SAP

We need to connect to our SAP system. In Chapter 3, “Customization” on page 35, we reviewed the information that we need to get from our SAP administrator. However, instead of hardcoding this into our script, which is not good style, we now begin to take advantage of the existing functions available in the Notes access for SAP solutions script libraries.

The Notes access for SAP solutions script libraries contain several useful functions in the SAPaccessUtilities script library. These functions call into the client personal address book. The SAPlogin function is extremely convenient for our needs, because it enables users to manage their SAP account documents independently of any SAP/Notes application development.

The users receive their SAP login information from the SAP administrator, create location and account documents for each SAP server to which they connect, associate the account documents with the respective location documents a (Figure 4-1 on page 65). They can then switch to the proper location prior to running the back-end scripts that call SAPlogin and then can access the various SAP servers running their applications independently of each other. Therefore, we develop our SAP/Notes application calling SAPlogin, and as long as the users created their location and SAP account documents as described later, there is no need to remember the specifics.

In our case, ITSO Bank has divisions in Austin, Boston, and Hamburg. Therefore, we have the users create three location documents titled ITSO Bank (SAP server Austin, SAP Server Boston, and so on). When they create their SAP account documents, they will need to select the **Only for location(s)** box specific to their SAP server login information (Figure 4-1).



Account name:	NASS Login for Boston
Account server name:	ITSO.BANK.BOSTON.COM
Login name:	muster
Password:	****
Protocol:	SAP
SAP employee ID:	1001
SAP system name:	LD3
SAP system number:	00
SAP client number:	800
SAP language:	EN
Only from location(s):	☐ Island (Disconnected) ☐ ITSO Bank (SAP server Austin) ☑ ITSO Bank (SAP server Boston) ☐ ITSO Bank (SAP server Hamburg)

Figure 4-1 Create a different location for each SAP Account document

Because connecting is such a critical step and will be a part of the development process that the user needs to step through, let’s review in detail what needs to be done when creating an SAP account document:

1. From the **Advanced** → **Accounts** view of your personal address book, select **New** → **Account** (Figure 4-2).

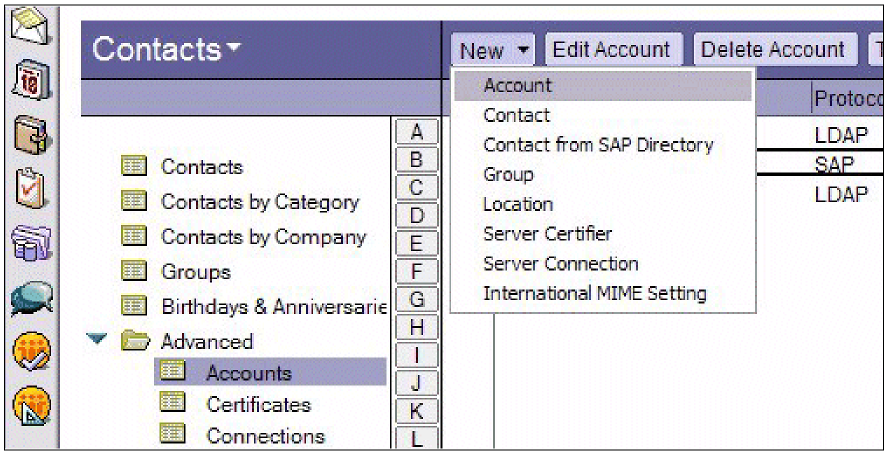
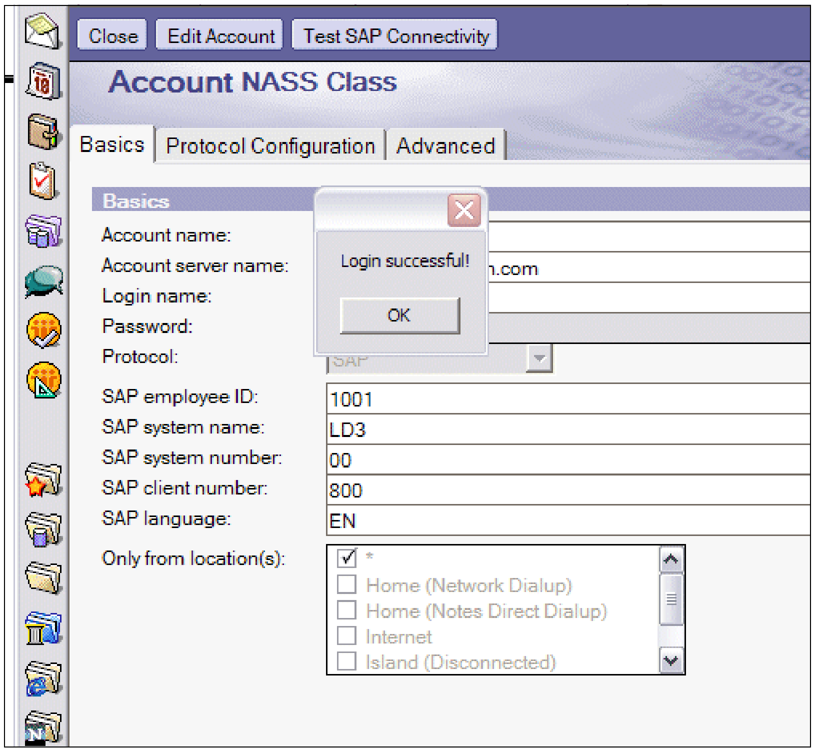


Figure 4-2 SAP Account document

2. After talking to the SAP administrator, the user will have all the information needed to connect. We select **SAP** as the protocol type, enter the SAP login information, and click **Test SAP Connectivity**. If all the information is correct, a Login successful pop-up window opens (Figure 4-3), followed by a list of details about the connected SAP system.

   This is our first call into the Notes access for SAP solutions libraries. The agent behind the Test SAP Connectivity button assembles the login data we have provided in our account document and gives it to a function called SAPLogin. This function passes the data to another function called SAPLoginWithOptions, which performs the login. In Chapter 5, "Troubleshooting" on page 91, we discuss some troubleshooting tips if any problems arise.



Login successful!

OK

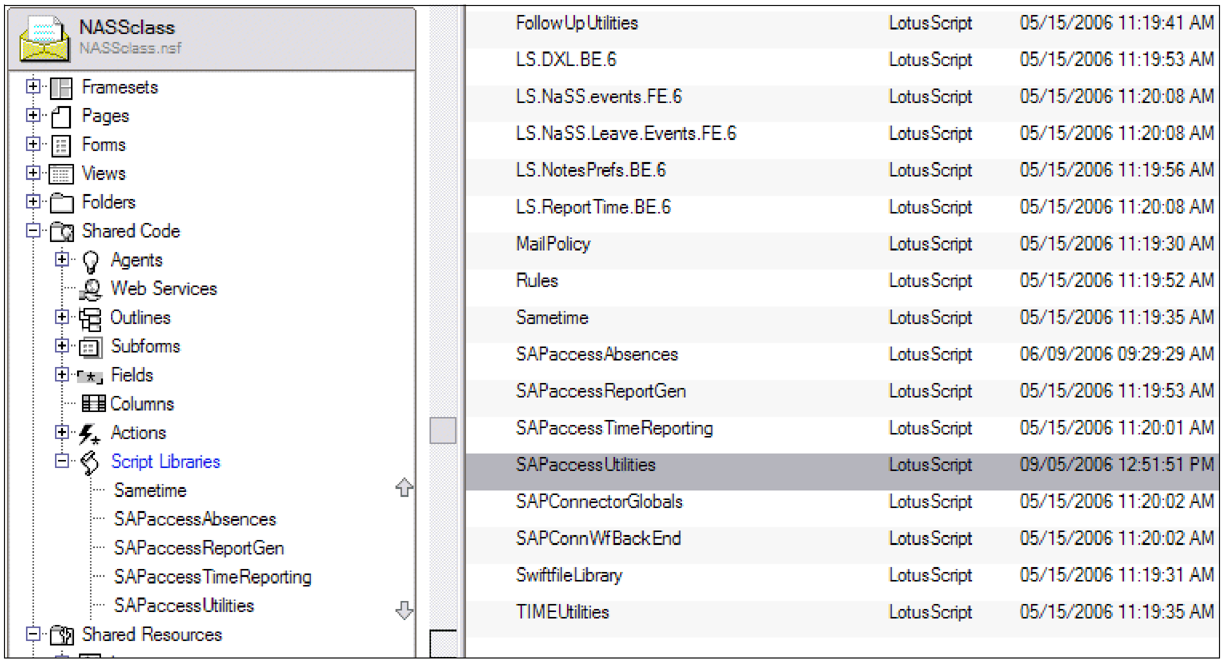
Account name:	
Account server name:	.com
Login name:	
Password:	
Protocol:	SAP
SAP employee ID:	1001
SAP system name:	LD3
SAP system number:	00
SAP client number:	800
SAP language:	EN
Only from location(s):	[✓ *] Home (Network Dialup) Home (Notes Direct Dialup) Internet Island (Disconnected)

Figure 4-3 Account document with Login successful message

Note: In Figure 4-3, we have not selected a location for our account document as detailed earlier. This is fine if the user has only one SAP account. However, it causes a problem if they have more. Therefore, make sure that in your client rollout, users with more than one SAP account have multiple locations and must to set this option in their account documents.

3. Now let's see how complicated it is going to be to call the functions in the Notes access for SAP solutions script library. To really test the usability, let's put together a login application completely independent of Lotus Notes access for SAP solutions.

   We copy the entire SAPaccessUtilities library into the new Notes database or template (Figure 4-4).

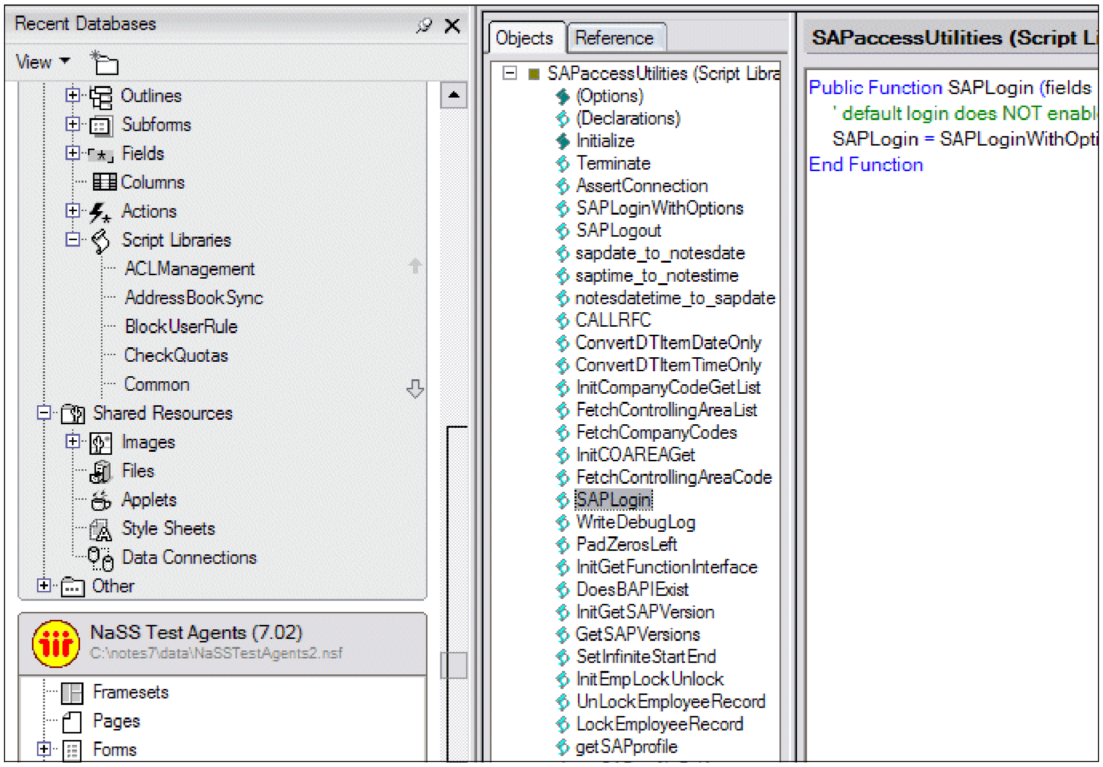


FollowUpUtilities	LotusScript	05/15/2006 11:19:41 AM
LS.DXL.BE.6	LotusScript	05/15/2006 11:19:53 AM
LS.NaSS.events.FE.6	LotusScript	05/15/2006 11:20:08 AM
LS.NaSS.Leave.Events.FE.6	LotusScript	05/15/2006 11:20:08 AM
LS.NotesPrefs.BE.6	LotusScript	05/15/2006 11:19:56 AM
LS.ReportTime.BE.6	LotusScript	05/15/2006 11:20:08 AM
MailPolicy	LotusScript	05/15/2006 11:19:30 AM
Rules	LotusScript	05/15/2006 11:19:52 AM
Sametime	LotusScript	05/15/2006 11:19:35 AM
SAPaccessAbsences	LotusScript	06/09/2006 09:29:29 AM
SAPaccessReportGen	LotusScript	05/15/2006 11:19:53 AM
SAPaccessTimeReporting	LotusScript	05/15/2006 11:20:01 AM
SAPaccessUtilities	LotusScript	09/05/2006 12:51:51 PM
SAPConnectorGlobals	LotusScript	05/15/2006 11:20:02 AM
SAPConnWfBackEnd	LotusScript	05/15/2006 11:20:02 AM
SwiftfileLibrary	LotusScript	05/15/2006 11:19:31 AM
TIMEUtilities	LotusScript	05/15/2006 11:19:35 AM

Figure 4-4 Notes access for SAP solutions Script Libraries

4. We then build a simple agent to call the SAPLogin function (Figure 4-5).

> **Note:** In Appendix B, "Additional material" on page 113, we include a test database that was used to develop many of the Lotus Notes access for SAP solutions templates. These test agents are generally very helpful for initial testing prior to working them into the GUI.



```
Public Function SAPLogin (fields
  ' default login does NOT enabl
  SAPLogin = SAPLoginWithOpti
End Function
```

Figure 4-5 The SAPlogin function

5. We use the simple login script included in the test agents database to test if we can connect to the SAP server specified. This time, we do not have to set it in our LotusScript, as needed in Chapter 3, "Customization" on page 35, but in the Account document in our personal address book. See the LotusScript code in Example 4-1.

Example 4-1 Code for TestConfiguredSAPConnection agent

```
Option Public
Option Declare
Use "SAPaccessUtilities"

Sub Initialize
   Dim profile List As Variant
   On Error Goto ehandler

   getSAPProfile profile
   If SAPLogin (profile) = 0 Then
      Msgbox "Login successful!"
      SAPLogout
   Else
      Msgbox "Login did not succeed"
   End If
   Exit Sub

ehandler:
   Msgbox "Error logging in to SAP: " & Error$
   Exit Sub
End Sub
```

By simply copying the script library and adding a few lines of code, we already finished step 1 of our road map.Within 30 minutes, you can add connectivity to any local Notes database that is enabled for using your existing SAP account documents. For your rollout, you are not tied to a user by user case; you roll out this template and each user can automatically start with their own SAP account documents.

The good thing about the script libraries developed for Lotus Notes access for SAP solutions is that they work with the user's personal address book. Therefore, after we create the SAP Account documents there, any local database with the appropriate script libraries copied into them can connect, because they just need to have access to the personal address book to get the needed SAP information. We copy the SAPaccessUtilities script library from either the Lotus Notes access for SAP solutions Name and Address Book template or the Lotus Notes access for SAP solutions mail template. Select the SAPaccessUtilities as seen in Figure 4-5 on page 68 to copy and paste it into our new template.

4.3 Step 2: Getting SAP details

Now that we can connect, we must find the BAPIs that we need. In Chapter 3, "Customization" on page 35, we discussed how we had to experiment and search for the correct BAPI and tables. We used transaction SE37 to look at BAPI_VENDOR_GETDETAIL and RFC_READ_TABLE. In this chapter, we do not need to find these; we did that already. Your SAP administrator usually will give you the correct BAPI and details about how to call it. You now need to access it through Notes. This is probably the easiest step for the Notes development team if all goes well.

To summarize, there are three key pieces of information that the Notes developer needs:

- The list of BAPIs (or table names if RFC_READ_TABLE is involved) that are needed to access the SAP information that our Notes application will need.
- A list of the SAP input and output parameters for the BAPI that is going to be used. (For a table, the input will be the syntax of the WHERE clause and the output will be a list of the column names of the table. We describe this in an example later.)
- How to run the BAPI in the SAP GUI interface. (Although this is not completely necessary, this step is usually helpful in many troubleshooting situations.)

Tip: We recommend that you always go into the SAP GUI and run the BAPI in the function builder. This enables you to get an overall idea of what is feasible with this particular module, because you will see its defaults and the necessary steps to call and run it. This was always helpful to our team. (See Chapter 5, "Troubleshooting" on page 91 for more details.)

4.4 Step 3: Getting SAP data with Notes access for SAP solutions library function calls

In Chapter 3, “Customization” on page 35, we customized the (SAPContactSearchDlg) form and added an option for vendors. We wrote our script completely from scratch. We broke our agent into four basic sections:

1. Setting the BAPI or table we need to call.
2. Creating a SELECT statement for use in the Execute method of the LCConnection class to generate the SAP result set (the structure of the SAP data returned) and its metadata definition.
3. Executing the SELECT statement to create our result set and return the data values.
4. Using the LCFieldlist and LCField classes to look up and format the data for a transfer to the UI.

Let’s go through in detail rebuilding our vendor example from Chapter 3, “Customization” on page 35 using the function calls available in the Notes access for SAP solutions script libraries instead of calling the LCConnection, LCFieldlist, or LCField classes directly. In our development stage, we found it extremely helpful to develop our agent first, independent of our UI. In each case, we started with a very simplified agent. You can download a database containing all the simplified agents that were used to build the Lotus Notes access for SAP solutions application in the downloads associated with this paper. We highly recommend playing with these agents to get a better feel for how to use the Notes access for SAP solutions script libraries in general. As mentioned earlier, we go through two sample agents:

- Calling a BAPI
- Calling an SAP table (RFC_READ_TABLE)

By stepping through the details of these two agents, one calling the BAPI BAPI_VENDOR_FIND and one calling table LFA1 via RFC_READ_TABLE, we cover the major details needed to effectively use the Lotus Notes access for SAP solutions functions calls to return SAP data.

It was very helpful to go through this exercise to get a very basic idea of what is entailed in getting data from SAP. But it would be better if we had a set of functions already built and all we had to do was set up a map of our Notes data to our SAP data. This is exactly what the Notes access for SAP solutions script libraries give us. Let’s look at the details of an agent that calls these libraries and compare it to our development in Chapter 3, “Customization” on page 35.

Let’s step through the code of the AA-TestVendorSearch agent of the testing DB downloadable with this paper (see Appendix B, “Additional material” on page 113). From the code in Example 4-2, we can see that we call the SearchVendors function within the SAPaccessContactMgmt script library.

Example 4-2 Agent AA-TestVendorSearch

```
Option Public
Option Declare
Use "SAPaccessContactMgmt"
Sub Initialize
   Dim profile List As Variant
   Dim session As New notessession
   Dim db As NotesDatabase
   Dim doc As notesdocument
   Dim count As Integer
   Dim i As Integer
   Dim cnames As Variant
   Dim cids As Variant
   Dim msgs As Variant

   getSAPProfile profile
   SAPLogin profile
   Set db = session.currentdatabase
   Set doc = db.CreateDocument

   ' set up a search by Vendorname, all vendors
   doc.T_FIELDVALUE = "C*"

   count = SearchVendors(doc)

   Msgbox "Found " & count & " matching vendors"

   cnames = doc.TL_FIELDVALUE
   cids = doc.TL_VENDORID

   For i = 0 To count - 1
      Msgbox  Cstr(i) & ": Id = " & cids(i) & ", " & cnames(i)
      'If Len(msgs(i)) > 0 Then Msgbox msgs(i)
      If i > 20 Then
         Msgbox "Too many to display"
         Exit For
      End If
   Next

End Sub
```

A review of the SearchVendors function in Example 4-3 show us that this function calls the InitVendorFind function in Example 4-4.

Example 4-3 Function SearchVendors

```
Public Function SearchVendors(doc As NotesDocument) As Integer
   ' search for vendors based on 1 or more fields

   InitVendorFind
   SearchVendors = CallRFC(doc, gMD_BAPI_VENDOR_FIND)

End Function
```

Example 4-4 Sub InitVendorFind

```
Public Sub InitVendorFind
      'Dim gMD_BAPI_VENDOR_FIND As SAPMetadata

   If gMD_BAPI_VENDOR_FIND Is Nothing Then
      Set gMD_BAPI_VENDOR_FIND = New SAPMetaData(5, 3)
      gMD_BAPI_VENDOR_FIND.IsInitialized = False
   End If

   If Not gMD_BAPI_VENDOR_FIND.IsInitialized Then
```

```
      gMD_BAPI_VENDOR_FIND.BAPI_NAME = "BAPI_VENDOR_FIND"
      gMD_BAPI_VENDOR_FIND.Fetch1Row = False

      ' Inputs: max-count (default 0), "place holder" (default X), table name
(default LFA1)
      ' field name (default NAME1, but could be others), field value (default *,
could be anything)
      gMD_BAPI_VENDOR_FIND.InitInputItem 0, "N_MAXCOUNT", "", "MAX_CNT", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 0, 0

      gMD_BAPI_VENDOR_FIND.InitInputItem 1, "T_PLHOLD", "", "PL_HOLD", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 1, "X"

      gMD_BAPI_VENDOR_FIND.InitInputItem 2, "T_TABLENAME", "SELOPT_TAB",
"TABNAME", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 2, "LFA1"

      gMD_BAPI_VENDOR_FIND.InitInputItem 3, "T_FIELDNAME", "SELOPT_TAB",
"FIELDNAME", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 3, "NAME1"

      gMD_BAPI_VENDOR_FIND.InitInputItem 4, "T_FIELDVALUE", "SELOPT_TAB",
"FIELDVALUE", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 4, "*"

      ' Outputs: value of selected field, VENDOR number, errmsg
      gMD_BAPI_VENDOR_FIND.InitOutputItem 0, "TL_FIELDVALUE", "RESULT_TAB",
"FIELDVALUE", False
      gMD_BAPI_VENDOR_FIND.InitOutputItem 1, "TL_VENDORID", "RESULT_TAB",
"VENDOR_NO", False
      gMD_BAPI_VENDOR_FIND.InitOutputItem 2, "TL_RETURNMSG", "RESULT_TAB",
"MESSAGE", False

      gMD_BAPI_VENDOR_FIND.IsInitialized = True
   End If

End Sub
```

And finally, we are at the base of how all this works. If we set up this initialization function properly, all the other function calls available to us will work automatically. After we understand how to put together the pieces of the InitVendorFind function above, we are basically done with our back-end development. Because as we see in the function SearchVendors, after the InitVendorFind is done, all we need to do to get our data into our document output fields is call CallRFC.

In other words, all we really need to do to develop our Notes/SAP integration is carefully determine our:

- Notes: Input and output variables
- SAP: Input and output variables

And then, we map (point) them to each other.

How to set up this mapping is the trickiest part to understand in the Lotus Notes access for SAP solutions architecture. However, after you become familiar with its setup, customizing the

Lotus Notes access for SAP solutions templates or developing your own application will become fairly simple. To completely explain how the Lotus Notes access for SAP solutions architecture works, let's go through a few examples of how to call both a BAPI and an SAP table with the RFC_READ_TABLE module. These two cases are a little bit different, but fortunately they are the only two cases we need to understand to take full advantage of Lotus Notes access for SAP solutions.

In the other simplified examples in the test agent database, you will find that all the BAPIs can now be called in the exact same manner. The most complex part of this entire process is putting together the initialization code found in Example 4-4 on page 71. To understand the exact details of the SAPMetadata object, go through the details of the code found in the Declarations section of SAPaccessUtilities. However, for development purposes, all we need to know is how to use it. We can just copy this library into all our Notes/SAP applications and we are set.

All we need to know to use CallRFC is the following information:

- The name of the BAPI
- How many input and output parameters are needed
- The default values, if any
- SAP input and output variable names
- Notes input and output variable names

Important: Remember that most data returned from the CallRFC function is returned in arrays. This is important when we discuss hooking our back-end LotusScript code with our UI interface.

4.4.1 Section 1: Setting the selected BAPI/table

In section one of the code, we declare the BAPI or table we are going to call by setting the database and metadata properties of the LCConnection class (Example 4-5).

Example 4-5 Calling BAPIs and tables

```
‘ code for calling a BAPI
   source.Database = "BAPI_VENDOR_GETDETAIL"
   source.Metadata = "*"

‘ code for calling a table
   source.Database = "RFC_READ_TABLELFA1"
   source.Metadata = "LFA1"
```

Our goal in using the Lotus Notes access for SAP solutions templates is to separate ourself from the details of the LC LSX classes (in this case, having to know that there are two properties in the LCConnection class: database and metadata, as shown in Example 4-5). At the base of Lotus Notes access for SAP solutions is a function CallRFC in the script library SAPaccessUtilities. As we can see from the code snippet in Example 4-6 on page 74, after we have the name of our BAPI or table, we prepend `gMD_` (standing for global metadata) to the name and we are done with section 1 of our code. Therefore, we only need to know the name of our BAPI or table and not the properties of the LCConnection class.

Example 4-6 Calls to the CallRFC function

```
‘ code for calling a BAPI
   SearchVendors = CallRFC(doc, gMD_BAPI_VENDOR_FIND)
‘ code for calling a table
   FetchVendorNames = CallRFC(doc, gMD_RFC_READ_TABLELFA1)
```

4.4.2 Section 2: Mapping Notes and SAP data

In this section, we go through the initialization done in Example 4-4 on page 71. In general, the developers set the dimensions for all their BAPIs in the Declarations section of the SAPaccessUtilities library. You can go there to see a list of all the BAPIs and tables that were used to create the Lotus Notes access for SAP solutions templates.

Next, we need to compute the size of our array of SAPMetaData objects. The size is determined by the number of input parameters and the number of output parameters. In our case, we needed five inputs to run the BAPI and we wanted back three output parameters. Each BAPI is going to be different, and even calling the same BAPI can result in a different number of parameters depending on what specific output is required by the UI of your application.

Now that we have set the BAPI or table, let’s briefly look at the CallRFC function. As we can see in Example 4-7, it takes as its arguments a Notes document and a custom data type called SAPMetaData, which was especially designed for Lotus Notes access for SAP solutions.

Example 4-7 General calling syntax for CallRFC function

```
Function CallRFC(source As NotesDocument, metadata As SAPMetaData) As Integer
```

Therefore, all we really need to understand is that CallRFC maps the fields on our Notes document to the array elements contained in our SAPMetaData object. Therefore, when thinking about CallRFC, we really should think of Notes input/output fields and SAP input/output data. The details of how we map them together makes up the majority of the Init function.

Understanding the SAPMetaData data type is central to how to use the Notes access for SAP solutions libraries. In fact, as we will see by the end of this discussion, it is almost all of the development that will be needed for most of our Notes/SAP applications. Therefore, this is probably the most important section of this paper, so set aside some time to really focus on the next few pages; it will be well worth it.

Building the function needed to initialize the SAPMetaData data type is usually the most time-consuming part of the application development. It requires a good overall picture of the application layout. We need to look at the application from a big picture perspective. What do we want?

From our Notes perspective, we want to be able to input a string (for example, C*), click a button, and get back a list of all our vendors whose names begin with the letter C. In addition, we also want to get their vendor number. Therefore, at the first pass, we have the following input and outputs (Table 4-1).

Table 4-1 Input and output values

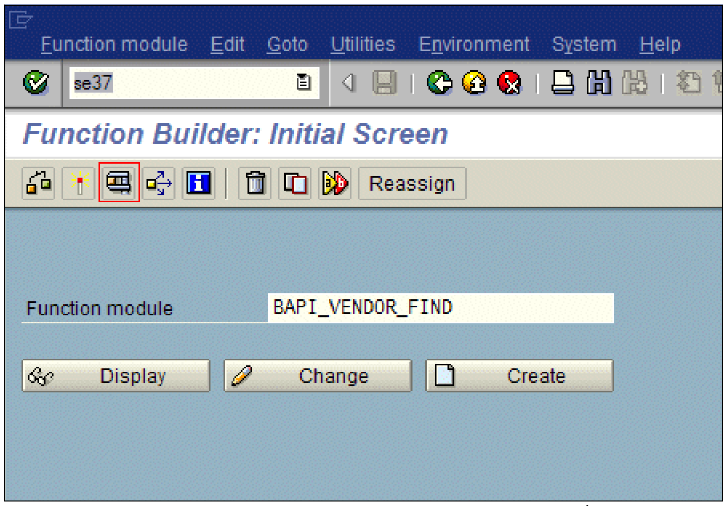
Inputs	Outputs
Search string	Vendor name
	Vendor ID number

Because in this example we are only looking for a relationship between the vendor name and number, as well as maybe an error message, three output parameters are sufficient. For input parameters, we required some input from our SAP developers. In this particular case, our SAP developer explained that we need the following information to run our BAPI:

- Set MAX_CNT default to `0`.
- Set PL_HOLD to `X`.
- For SELOPT_TAB, set the parameters of this table as follows:
  - TABNAME = `LFA1`
  - FIELDNAME = `NAME1`
  - FIELDVALUE = “`C*`”

These instructions from our SAP developer seemed cryptic at first, so was helpful to run the BAPI again. The following figures correspond to the steps needed to run the BAPI_VENDOR_FIND in the SAP GUI client:

1. First, run transaction `se37` and open BAPI_VENDOR_FIND (Figure 4-6).

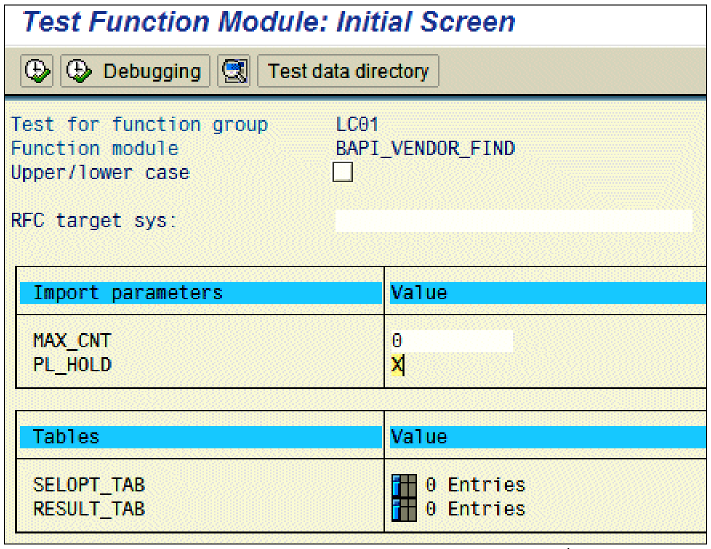

Function module Edit Goto Utilities Environment System Help
se37
Function Builder: Initial Screen
Reassign
Display
Change
Create

Function module: BAPI_VENDOR_FIND

Figure 4-6 BAPI_VENDOR_FIND in Function Builder[1]

[1] © SAP AG 2006. All rights reserved.

2. In Figure 4-6 on page 75, click the wrench icon (or press F8) to run a single test of the BAPI. Enter `0` for MAX_CNT and `X` for PL_HOLD (Figure 4-7).



Test Function Module: Initial Screen

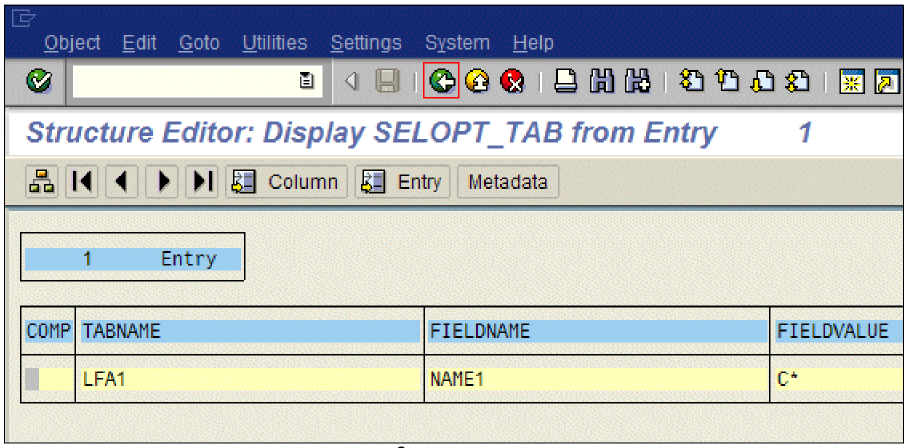
Test for function group	LC01
Function module	BAPI_VENDOR_FIND
Upper/lower case	☐
RFC target sys:	

Import parameters	Value
MAX_CNT	0
PL_HOLD	X

Tables	Value
SELOPT_TAB	0 Entries
RESULT_TAB	0 Entries

Figure 4-7 BAPI_VENDOR_FIND input parameters[1]

3. Entering the MAX_CNT and PL_HOLD values is fairly straightforward after we knew the necessary default values. Entering the data for the SELOPT_TAB table took a little experimentation however. We found this struggle invaluable for later troubleshooting though (see further details in Chapter 5, “Troubleshooting” on page 91). To enter the data for the SELOPT_TAB table, first click the icon in the Value column corresponding to **SELOPT_OPT** (Figure 4-8) After entering `LFA1` for TABNAME, `NAME1` for FIELDNAME, and `C*` for FIELDVALUE, click the green circle with the white arrow to go back to the previous window (or press F3).



Structure Editor: Display SELOPT_TAB from Entry 1

1 Entry

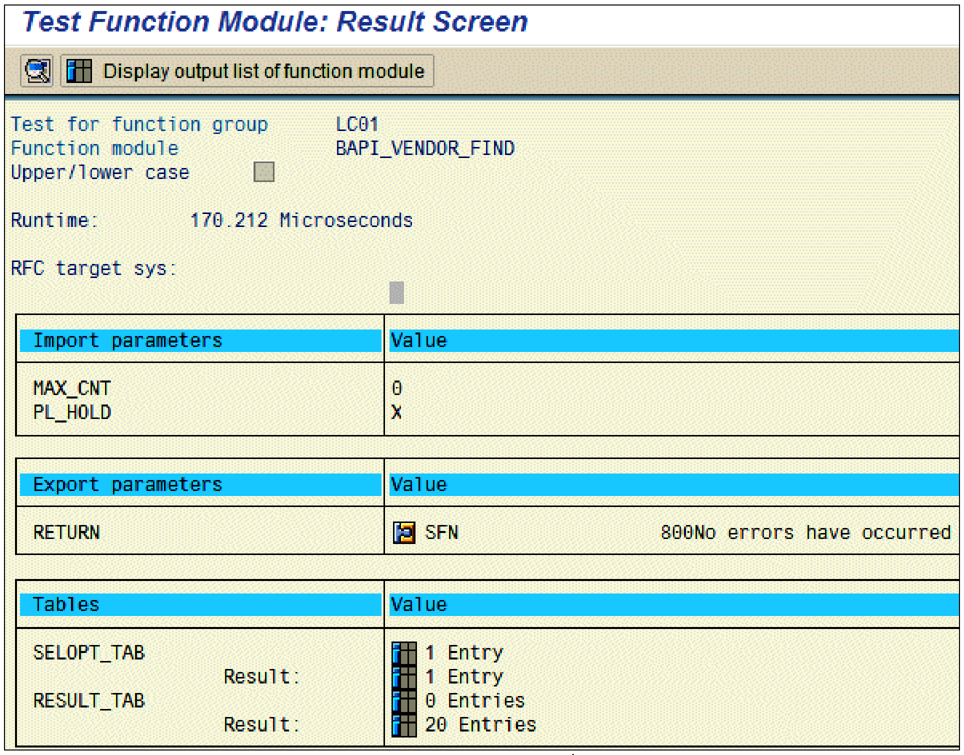
COMP	TABNAME	FIELDNAME	FIELDVALUE
	LFA1	NAME1	C*

Figure 4-8 Setting selection options[2]

[1] © SAP AG 2006. All rights reserved.
[2] © SAP AG 2006. All rights reserved.

Restriction: For FIELDVALUE our administrator told us to use “C*”, but this did not work. So we tried ‘C*’, which also failed. We finally realized that just C* was needed. In other BAPIs, ‘C%’ was needed for what seemed to be the same data. Therefore, SAP has no standard; the value and requirements are completely arbitrary. They are completely dependent on what the person who developed the BAPI decided. So to reiterate, it is always advisable to experiment with your BAPIs during development.

4. You now return to the window shown in Figure 4-7 on page 76. We are ready to run the BAPI and return all the vendors whose names start with a C. Click the clock icon to execute the BAPI (or press F8).


Test Function Module: Result Screen
Display output list of function module

```
Test for function group      LC01
Function module              BAPI_VENDOR_FIND
Upper/lower case

Runtime:        170.212 Microseconds

RFC target sys:
```

Import parameters	Value
MAX_CNT	0
PL_HOLD	X

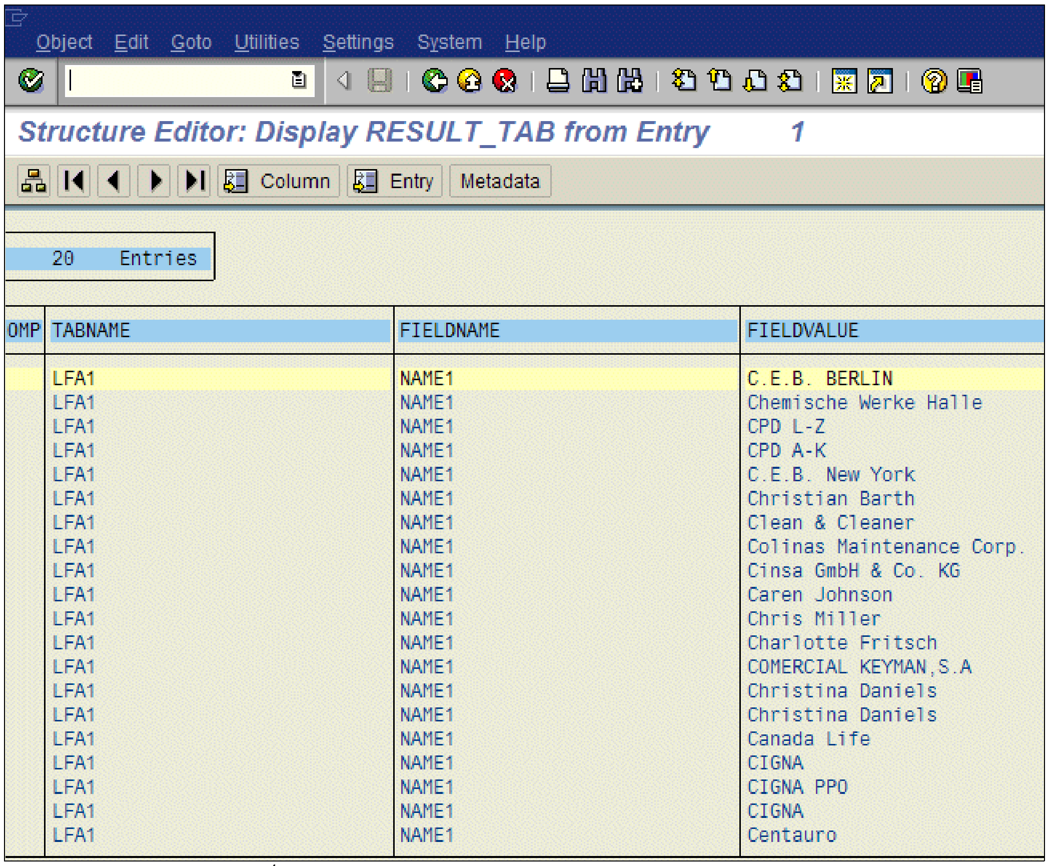
Export parameters	Value
RETURN	SFN 800No errors have occurred

Tables		Value
SELOPT_TAB		1 Entry
	Result:	1 Entry
RESULT_TAB		0 Entries
	Result:	20 Entries

Figure 4-9 Result Screen for BAPI_VENDOR_FIND[1]

[1] © SAP AG 2006. All rights reserved.

a. We can see in Figure 4-10 that we have 20 entries in the RESULT_TAB table. Click the icon next to the 20 entries to see all vendors beginning with the letter C.



Structure Editor: Display RESULT_TAB from Entry 1

20 Entries

OMP	TABNAME	FIELDNAME	FIELDVALUE
	LFA1	NAME1	C.E.B. BERLIN
	LFA1	NAME1	Chemische Werke Halle
	LFA1	NAME1	CPD L-Z
	LFA1	NAME1	CPD A-K
	LFA1	NAME1	C.E.B. New York
	LFA1	NAME1	Christian Barth
	LFA1	NAME1	Clean & Cleaner
	LFA1	NAME1	Colinas Maintenance Corp.
	LFA1	NAME1	Cinsa GmbH & Co. KG
	LFA1	NAME1	Caren Johnson
	LFA1	NAME1	Chris Miller
	LFA1	NAME1	Charlotte Fritsch
	LFA1	NAME1	COMERCIAL KEYMAN,S.A
	LFA1	NAME1	Christina Daniels
	LFA1	NAME1	Christina Daniels
	LFA1	NAME1	Canada Life
	LFA1	NAME1	CIGNA
	LFA1	NAME1	CIGNA PPO
	LFA1	NAME1	CIGNA
	LFA1	NAME1	Centauro

Figure 4-10 Table LFA1[1]

We now have what we want in the SAP GUI and we can see from where the values come that we enter into our initializing functions (Table 4-2).

Table 4-2 BAPI_VENDOR_FIND arguments

Inputs	**Outputs**
MAX_CNT default to 0	RESULT_TAB.1.FIELDVALUE
PL_HOLD default to X	RESULT_TAB.2.VENDOR_NO
SELOPT_TAB.1.TABNAME = LFA1	RESULT_TAB.3.MESSAGE
SELOPT_TAB.2.FIELDNAME = NAME1	
SELOPT_TAB.3.FIELDVALUE = C*	

[1] © SAP AG 2006. All rights reserved.

Now that we know our inputs and outputs, it is clear that the dimensions of our array of SAPMetaData objects 5 and 3 correspond to our input and output, respectively (Example 4-8).

Example 4-8 Setting SAP metadata

```
If gMD_BAPI_VENDOR_FIND Is Nothing Then
      Set gMD_BAPI_VENDOR_FIND = New SAPMetaData(5, 3)
      gMD_BAPI_VENDOR_FIND.IsInitialized = False
   End If
```

Now that we set the size of gMD_BAPI_VENDOR_FIND, we need to fill it with the mappings of our input and output variables. Let's look in greater detail at the syntax of the statements in Example 4-9.

Example 4-9 InitVenforFind subroutine

```
' Inputs: max-count (default 0), "place holder" (default X), table name (default
LFA1)
      ' field name (default NAME1, but could be others), field value (default *,
could be anything)
      gMD_BAPI_VENDOR_FIND.InitInputItem 0, "N_MAXCOUNT", "", "MAX_CNT", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 0, 0

      gMD_BAPI_VENDOR_FIND.InitInputItem 1, "T_PLHOLD", "", "PL_HOLD", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 1, "X"

      gMD_BAPI_VENDOR_FIND.InitInputItem 2, "T_TABLENAME", "SELOPT_TAB",
"TABNAME", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 2, "LFA1"

      gMD_BAPI_VENDOR_FIND.InitInputItem 3, "T_FIELDNAME", "SELOPT_TAB",
"FIELDNAME", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 3, "NAME1"

      gMD_BAPI_VENDOR_FIND.InitInputItem 4, "T_FIELDVALUE", "SELOPT_TAB",
"FIELDVALUE", False
      gMD_BAPI_VENDOR_FIND.SetItemDefault 4, "*"

      ' Outputs: value of selected field, VENDOR number, errmsg
      gMD_BAPI_VENDOR_FIND.InitOutputItem 0, "TL_FIELDVALUE", "RESULT_TAB",
"FIELDVALUE", False
      gMD_BAPI_VENDOR_FIND.InitOutputItem 1, "TL_VENDORID", "RESULT_TAB",
"VENDOR_NO", False
      gMD_BAPI_VENDOR_FIND.InitOutputItem 2, "TL_RETURNMSG", "RESULT_TAB",
"MESSAGE", False

gMD_BAPI_VENDOR_FIND.IsInitialized = True
```

There are four subroutines: InitInputItem, SetItemDefault, InitOutputItem, and IsInitialized. The details of these subroutines are included in the Declarations section of the SAPaccessUtilities script library. And again, we stress that the details are not as important as how to use them properly and in what script library they are located. If we know how to properly write the syntax of the statements in Example 4-9, to use them, wo only need to copy the SAPaccessUtilities library into our application, set up our init file, and call CallRFC.

To use InitInputItem, we need to know four things: the name of the BAPI, the name of the notes field, the SAP table or structure name, and the SAP field:

```
gMD_YOUR_BAPI_HERE.InitInputItem 4, “Notes field”, “table or structure name if any”, “SAP field”, false
```

From the AA-TestVendorSearch agent shown in Example 4-2 on page 70, the Notes input field name is T_FIELDVALUE, the table, as we discussed earlier, is SELOPT_TAB, and the SAP field is FIELDVALUE:

```
gMD_BAPI_VENDOR_FIND.InitInputItem 4, “T_FIELDVALUE”, “SELOPT_TAB”, “FIELDVALUE”, false
```

Therefore, this statement is our fourth input item and maps the T_FIELDVALUE field on our Notes document to the SAP FIELDVALUE column in the SELOPT_TAB.

Table 4-3 shows the Notes field names that are directly mapped to the SAP fields.

Table 4-3 SAP versus Notes input arguments

SAP inputs	Notes inputs
SELOPT_TAB.1.TABNAME = LFA1	T_TABLENAME
SELOPT_TAB.2.FIELDNAME = NAME1	T_FIELDNAME
SELOPT_TAB.3.FIELDVALUE = C*	T_FIELDVALUE

InitItemDefault sets basic default values. For the following statement, it sets a value that is constant:

```
gMD_BAPI_VENDOR_FIND.SetItemDefault 0, 0
```

For the following statement, it sets a basic value that gets overwritten by the value in the T_FIELDVALUE field on the Notes document:

```
gMD_BAPI_VENDOR_FIND.SetItemDefault 4, “*”
```

InitOutputItem works exactly like InitInputItem except it is used to map the outputs (Table 4-4).

Table 4-4 SAP versus Notes output arguments

SAP outputs	Notes outputs
RESULT_TAB.1.FIELDVALUE	TL_FIELDVALUE
RESULT_TAB.2.VENDOR_NO	TL_VENDORID
RESULT_TAB.3.MESSAGE	TL_RETURNMSG

And finally, only after we have properly set setup gMD_BAPI_VENDOR_FIND is IsInitialized set to True.

Now that we properly initialized the SAPMetaData objects, we can use the CallRFC function to run our BAPI. You can go into the details of the CallRFC function, found in the SAPaccessUtilities script library, but it is really not necessary because of the way the script libraries were built. As we saw earlier, in our simplified calling function, after the initialization takes place, the only argument needed for CallRFC is gMD_YOUR_BAPI_HERE. And finally, we see the extraordinary value of the Notes access for SAP solutions script libraries. Our back-end development is done!

Although it is complicated and will take some practice to get used to the formatting of your initialization function, this step and finding the correct script library in Lotus Notes access for SAP solutions to copy into your application will be the main part of developing any application. In general, to develop the back-end portion of almost any SAP/Notes application integration, you need to:

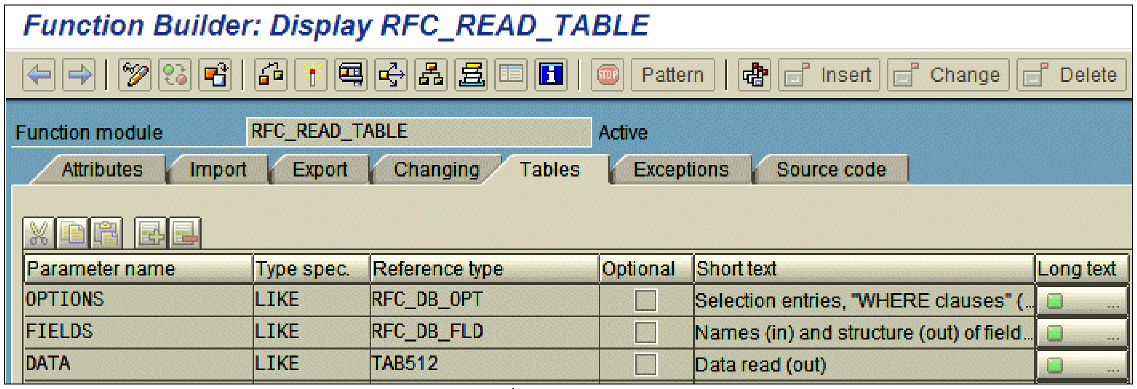
1. Ask your SAP administrator for access to the SAP server.
2. Ask your Notes UI developer what is needed from SAP.
3. Ask your SAP developer what BAPI or table to call for that SAP information.
4. Ask your SAP developer for details of how to execute the BAPI, focusing carefully on the input and output variables (listed in the Import, Export, and Tables tabs of the BAPI).
5. Experiment with the BAPI or table (through RFC_READ_TABLE).
6. Look through the sample test agents database to find a similar example to begin testing your agent.
7. Present the data to the UI.

To describe the slight differences of accessing a table rather than a BAPI with the Lotus Notes access for SAP solutions template script libraries, we go through a sample agent. And in order to solidify the complications of step 3, we go through setting up the initialization function once more, but by calling RFC_READ_TABLE for the LFA1 table. This also makes a convenient connect to use in describing how we return our SAP data to our Notes UI.

You can find the details of the test agent in the download database in agent AA-TestFetchVendorNames. This time, our test agent calls a different script library called SAPaccessESS-MSS that holds the functions we will use to access our SAP table LFA1 through RFC_READ_TABLE.

However, we still use the same construct to access SAP. We set up an initialization function followed by the CallRFC function. The only major difference here is that unlike calling a BAPI where the IMPORTS, EXPORTS, and TABLES are relatively fixed, with RFC_READ_TABLE, every time we call a new table the parameters of the call will change to correspond to the different columns of that specific table. Therefore, there is an additional step we must take into account to compensate for the different columns of the table.

To understand what is needed, let’s begin by looking at the RFC_READ_TABLE in the SAP GUI, specifically at the Tables tab Figure 4-11).



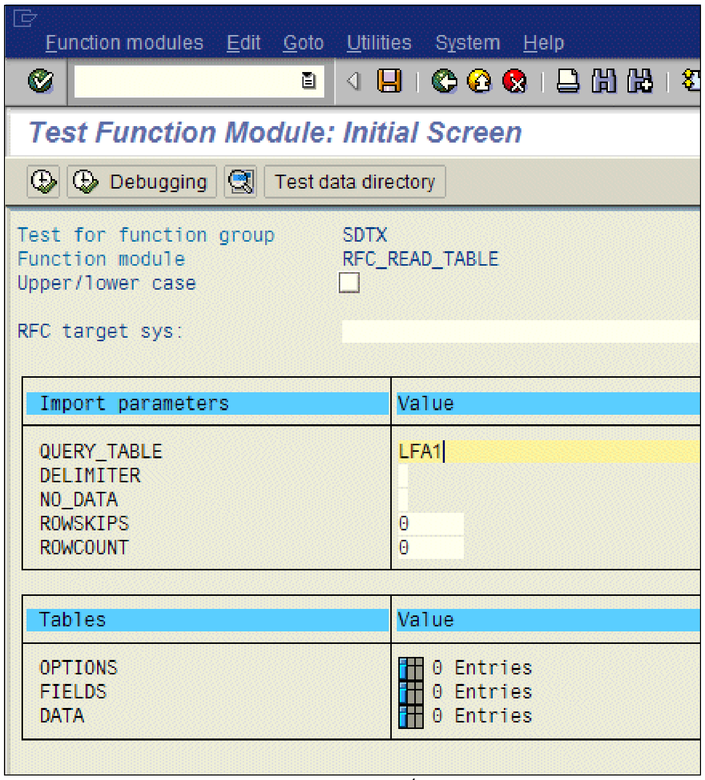
Function module RFC_READ_TABLE Active

Parameter name	Type spec.	Reference type	Optional	Short text	Long text
OPTIONS	LIKE	RFC_DB_OPT	☐	Selection entries, "WHERE clauses" (...	🟩 ...
FIELDS	LIKE	RFC_DB_FLD	☐	Names (in) and structure (out) of field...	🟩 ...
DATA	LIKE	TAB512	☐	Data read (out)	🟩 ...

Figure 4-11 Tables tab of RFC_READ_TABLE[1]

[1] © SAP AG 2006. All rights reserved.

As we see from Figure 4-11 on page 81, the FIELDS table defines the columns of the table that we want returned, while the OPTIONS table is used for the WHERE clause. Again, let's run RFC_READ_TABLE on the LFA1 table to get a better idea of what we need to expect. Our SAP developer group tells us we only need to enter `LFA1` in the QUERY_TABLE import parameters. For the details about how to configure the correct inputs for the FIELDS and OPTIONS tables, we first click **FIELDS** icon in the Value column (Figure 4-12).



Test for function group	SDTX
Function module	RFC_READ_TABLE
Upper/lower case	☐
RFC target sys:	

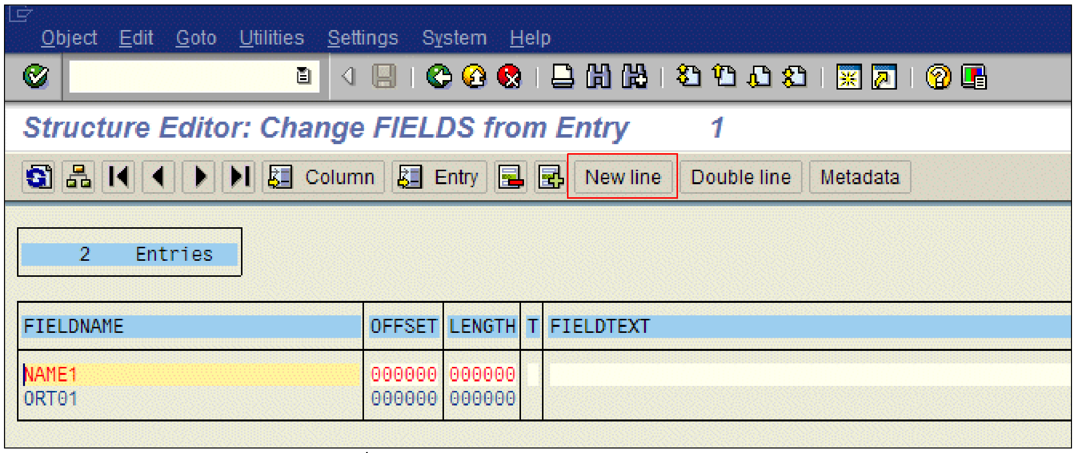
Import parameters	Value
QUERY_TABLE	LFA1
DELIMITER	
NO_DATA	
ROWSKIPS	0
ROWCOUNT	0

Tables	Value
OPTIONS	0 Entries
FIELDS	0 Entries
DATA	0 Entries

Figure 4-12 Setting the desired table[1]

Next, we set up out output parameters, the column names we want to return with this run of RFC_READ_TABLE, that is, the FIELDS table entries. For the sake of brevity, we only enter `NAME1`, the company name, and `ORT01`, the city name. We enter these by clicking the **Fields** icon in the value column and entering our desired columns (Figure 4-13 on page 83). Note that to enter each new line, you can only click the new line button; a carriage return does not add a new line.

[1] © SAP AG 2006. All rights reserved.



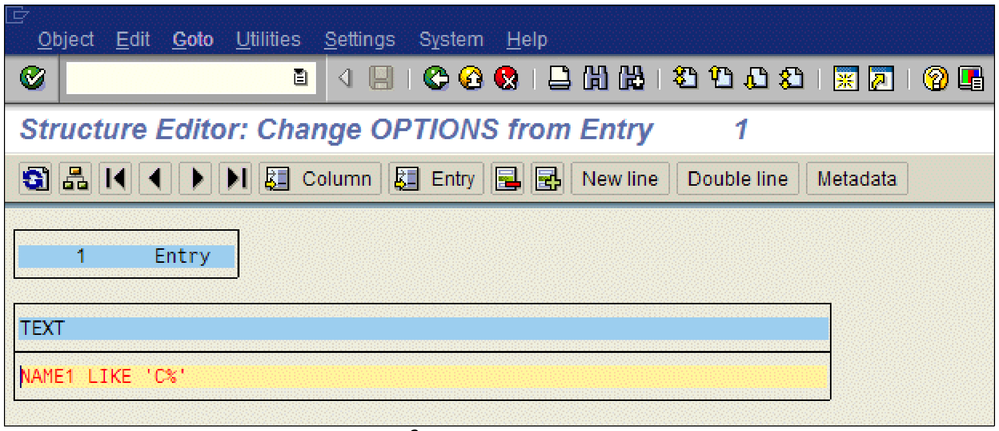
Structure Editor: Change FIELDS from Entry 1

2 Entries

FIELDNAME	OFFSET	LENGTH	T	FIELDTEXT
NAME1	000000	000000		
ORT01	000000	000000		

Figure 4-13 Fields table settings[1]

We go back and click the **Options** icon. We enter our WHERE clause (Figure 4-14).



Structure Editor: Change OPTIONS from Entry 1

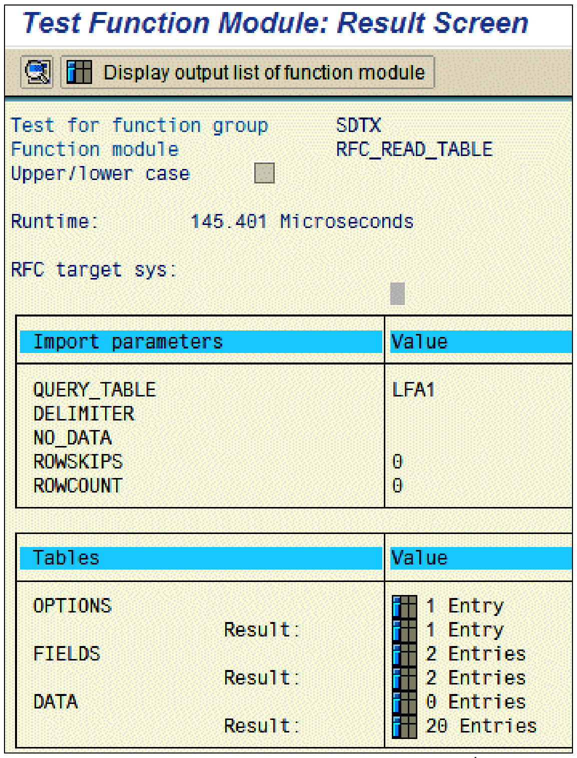
1 Entry

TEXT
NAME1 LIKE 'C%'

Figure 4-14 Setting selection options[2]

[1] © SAP AG 2006. All rights reserved.
[2] © SAP AG 2006. All rights reserved.

After setting these up and executing RFC_READ_TABLE, we have the same 20 companies as with BAPI_VENDOR_FIND (Figure 4-15).


Test Function Module: Result Screen
Display output list of function module

```
Test for function group      SDTX
Function module              RFC_READ_TABLE
Upper/lower case

Runtime:        145.401 Microseconds

RFC target sys:
```

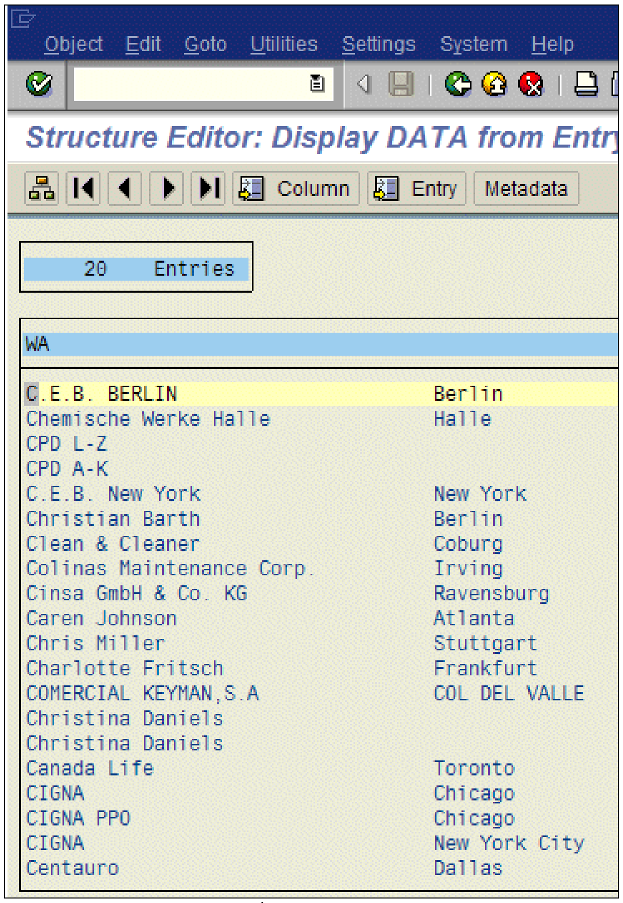
Import parameters	Value
QUERY_TABLE	LFA1
DELIMITER	
NO_DATA	
ROWSKIPS	0
ROWCOUNT	0

Tables		Value
OPTIONS		1 Entry
	Result:	1 Entry
FIELDS		2 Entries
	Result:	2 Entries
DATA		0 Entries
	Result:	20 Entries

Figure 4-15 RFC_READ_TABLE result set[1]

[1] © SAP AG 2006. All rights reserved.

The major difference with calling RFC_READ_TABLE is that we see only the columns specified in the FIELDS table (Figure 4-16).


Object Edit Goto Utilities Settings System Help
Structure Editor: Display DATA from Entry
Column
Entry
Metadata
20 Entries
WA

```
C.E.B. BERLIN                       Berlin
Chemische Werke Halle               Halle
CPD L-Z
CPD A-K
C.E.B. New York                     New York
Christian Barth                     Berlin
Clean & Cleaner                     Coburg
Colinas Maintenance Corp.           Irving
Cinsa GmbH & Co. KG                 Ravensburg
Caren Johnson                       Atlanta
Chris Miller                        Stuttgart
Charlotte Fritsch                   Frankfurt
COMERCIAL KEYMAN,S.A                COL DEL VALLE
Christina Daniels
Christina Daniels
Canada Life                         Toronto
CIGNA                               Chicago
CIGNA PPO                           Chicago
CIGNA                               New York City
Centauro                            Dallas
```

Figure 4-16 Result set[1]

Let's dissect the InitGetVendorName function in Example 4-10.

Example 4-10 InitGetVendorName function

```
If gMD_RFC_READ_TABLELFA1 Is Nothing Then
      Set gMD_RFC_READ_TABLELFA1 = New SAPMetaData(2, 4)
      gMD_RFC_READ_TABLELFA1.IsInitialized = False
End If

If Not gMD_RFC_READ_TABLELFA1.IsInitialized Then
   gMD_RFC_READ_TABLELFA1.BAPI_NAME = "RFC_READ_TABLELFA1"
   gMD_RFC_READ_TABLELFA1.Fetch1Row = False

   ' Input:  a list of column names to read back
   ' OPTIONS.TEXT should have a value such as: "SPRAS EQ 'EN''"
   ' TL_RFCFIELDNAMES should be a list of SAP column names for this table
   gMD_RFC_READ_TABLELFA1.InitInputItem 0, "T_OPTIONS", "OPTIONS", "TEXT", False
   gMD_RFC_READ_TABLELFA1.InitInputItem 1, "TL_RFCFIELDNAMES", "FIELDS",
"FIELDNAME",                False

   ' Outputs: language, 2-char code, country name, nationality name
```

[1] © SAP AG 2006. All rights reserved.

```
   gMD_RFC_READ_TABLELFA1.InitOutputItem 0, "TL_FIELDVALUE", "LFA1", "NAME1",
False
   gMD_RFC_READ_TABLELFA1.InitOutputItem 1, "TL_COUNTRY", "LFA1", "LAND1", False
   gMD_RFC_READ_TABLELFA1.InitOutputItem 2, "TL_CITY", "LFA1", "ORT01", False
   gMD_RFC_READ_TABLELFA1.InitOutputItem 3, "TL_VENDORID", "LFA1", "LIFNR", False

   gMD_RFC_READ_TABLELFA1.IsInitialized = True
End If
```

While the structure is the same, the inputs and outputs can be tricky at first. As we found in executing RFC_READ_TABLE earlier, the input consists of the OPTIONS and a FIELDS column used as an array of field names that we want to return. The table name for the QUERY_TABLE is in the name of the SAPMetaData object gMD_RFC_READ_TABLELFA1, so we do not need to include it as an input.

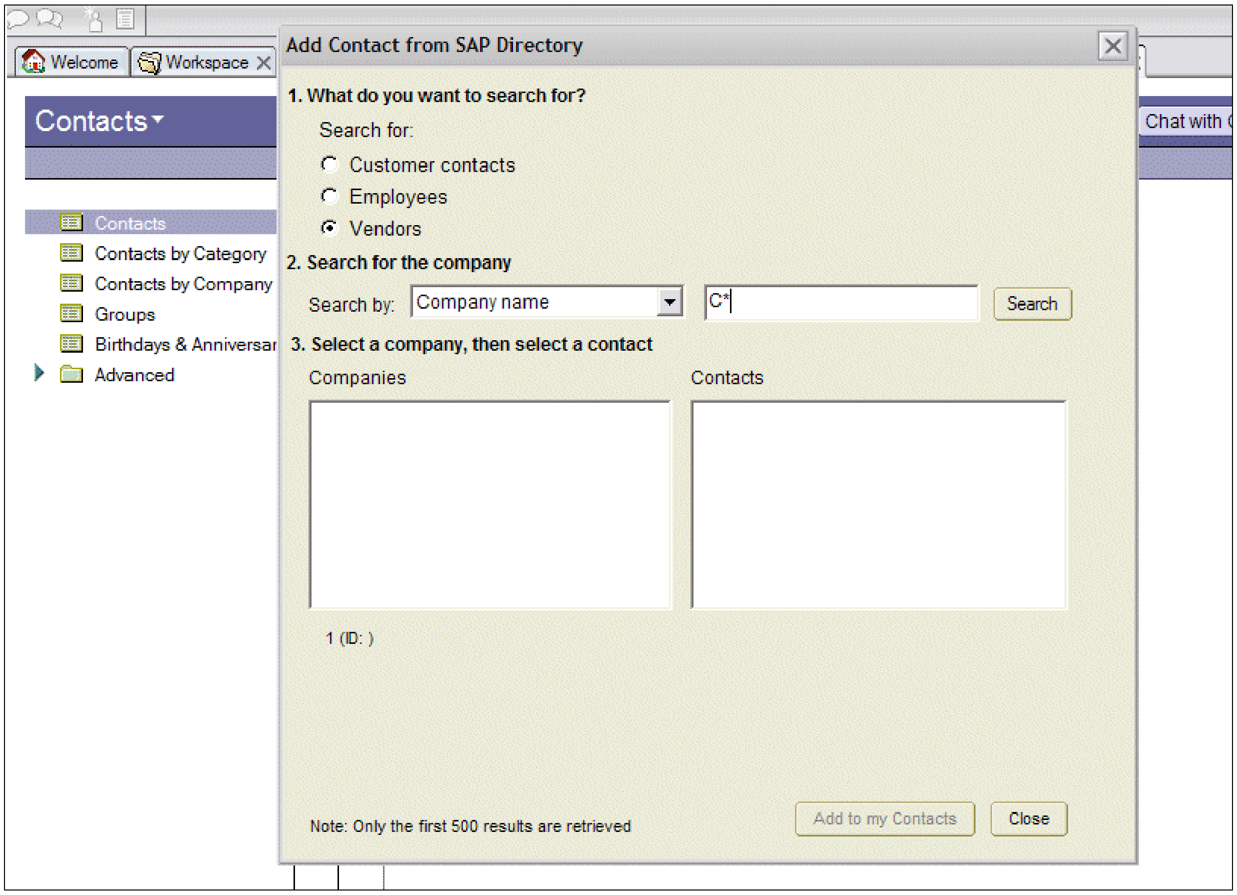
In Example 4-10 on page 85, we have a new Fetch1Row subroutine. This subroutine only determines whether we are dealing with one or multiple rows. We said earlier that the SAPMetaData object consists of arrays. That was not completely true: If there is only one row needed, the object is a scalar. Therefore, Fetch1Row denotes whether the data must be treated as an array or scalar.

The export parameters consist simply of the columns we want to return in this case. Running our agent, we see that we are able to return the company names in our simple print statement. Therefore, our back-end agent works and it is time to pass the SAP data to the UI.

4.5 Step 4: Passing SAP data to the UI

We purposely left the discussion of the UI until last. It was necessary to study the back end first so that we know what is expected before we can begin developing the UI. At its most basic, we require that the UI provides the inputs needed to feed the BAPIs or RFCs that return to the Notes client the requested SAP data generated by the BAPIs or RFCs. While the UI options are limitless, there are a few considerations that need to be taken into account with any Lotus Notes access for SAP solutions UI development. We describe the UI of our vendor example to demonstrate these considerations.

Running BAPIs or RFCs can be very confusing to the typical Notes user, but entering data into a Notes form is not. To request the list of vendors, it was only necessary for the Notes user to select the **Vendors** radio button and enter the first character of the vendor name followed by a wild card (Figure 4-17).



Add Contact from SAP Directory

1. What do you want to search for?

Search for:

( ) Customer contacts
( ) Employees
(•) Vendors

2. Search for the company

Search by: Company name C* Search

3. Select a company, then select a contact

Companies Contacts

1 (ID:)

Note: Only the first 500 results are retrieved

Add to my Contacts Close

Figure 4-17 Searching for Vendors that start with the letter C

Next, all the user needs to do is press Enter or click **Search** to return the desired list. Let's examine what is going on under the hood to get a better picture of the basics that a UI developer can expect to implement when parsing UI data for input and output from the Notes access for SAP solutions back-end script libraries.

As displayed in Example 4-11, if we look at the design behind the Search button of the form presented to the user for the vendor search, it shows two kinds of data: scalars (single values) and arrays (multiple values). This data is stored conveniently in a temporary Notes document. First, let's consider how the data is submitted to the Notes access for SAP solutions script libraries:

1. A temporary Notes document is created as a conduit for data going between SAP and Notes and function searchVends is called. At this point, as far as data entry from the user, the UI is finished. By selecting Vendor, we selected the BAPI/RFC to call, and in this case, the only other entry needed for this BAPI was a search string.

Example 4-11 Search button code

```
Sub Click(Source As Button)
   Dim s As New NotesSession
   Dim ws As New NotesUIWorkspace
   Dim thisdb As NotesDatabase
   Dim thisdoc As NotesDocument, thisuidoc As NotesUIDocument
   Dim retval As Long
   Dim sapflds List As Variant

   Set thisdb = s.CurrentDatabase
   Set thisuidoc = ws.CurrentDocument
   Set thisdoc = thisuidoc.Document

   retval = getSAPProfile(sapflds)
   retval = SAPlogin(sapflds)

   If Lcase(thisdoc.GetItemValue("SearchFor")(0)) = "customers" Then
      Call searchContacts(thisdoc)
   End If

   If Lcase(thisdoc.GetItemValue("SearchFor")(0)) = "employees" Then
      Call searchEmps(thisdoc)
   End If

   If Lcase(thisdoc.GetItemValue("SearchFor")(0)) = "vendors" Then
      Call searchVends(thisdoc)
   End If

   'Else
      'Call searchEmps(thisdoc)
   'End If

   Call SAPlogout

   Call thisuidoc.Reload
   Call thisuidoc.Refresh
End Sub
```

2. With the necessary inputs to run specific BAPIs and RFCs and specific SAP outputs needed for the Notes UI, we can develop the searchVends function, as shown in Example 4-12. At this point, the input portion of the UI development is finished and all that is needed is to set up how to handle the outputs of our SAP calls.

> **Important:** Again, we stress that a conversation between the Notes developer and the SAP developer concerning the needed inputs and outputs to run specific BAPIs and RFCs must take place for the implementation to be successful.

Example 4-12 The searchVends function definition

```
Sub searchVends(thisdoc As NotesDocument)

   Dim retval As Integer

   'Inputs needed for BAPI
   thisdoc.T_FIELDVALUE = thisdoc.GetItemValue("SearchVendorName")(0) + "*"

        'Outputs to be returned to UI for display
   thisdoc.TL_FIELDVALUE = ""
   thisdoc.TL_VENDORID = ""
   thisdoc.TL_COUNTRY = ""
   thisdoc.TL_CITY = ""

   retval = SearchVendors(thisdoc)


   thisdoc.Vendors = ""
   thisdoc.NoResultTextE = "<No results found>"

End Sub
```

3. As discussed earlier, the wisdom of the Lotus Notes for SAP solutions architects was to use arrays to return the results of the BAPI and RFC calls. Therefore, if we examine the contents of the fields TL_FIELDVALUE, TL_VENDORID, TL_COUNTRY, and TL_CITY on our temporary document, we find arrays in each. What has been cleverly crafted here is a table on our temporary Notes document that corresponds to the table (result set) returned by the BAPI. Therefore, while the vendor name was our immediate request and can be returned by the individual array values in the TL_FIELDVALUE field, after we select a specific vendor, its index, synonymous with a row number in a table, can be used to retrieve the corresponding values in the other fields, namely TL_VENDORID, TL_COUNTRY, and TL_CITY.

5

Troubleshooting

Lotus Notes access for SAP solutions templates provide an incredible set of libraries to use in developing your SAP/Notes application integrations. The reality is that when building any application, there are stumbling blocks and learning curves. This chapter lists some helpful hints we discovered when putting together the example ITSO Bank application.

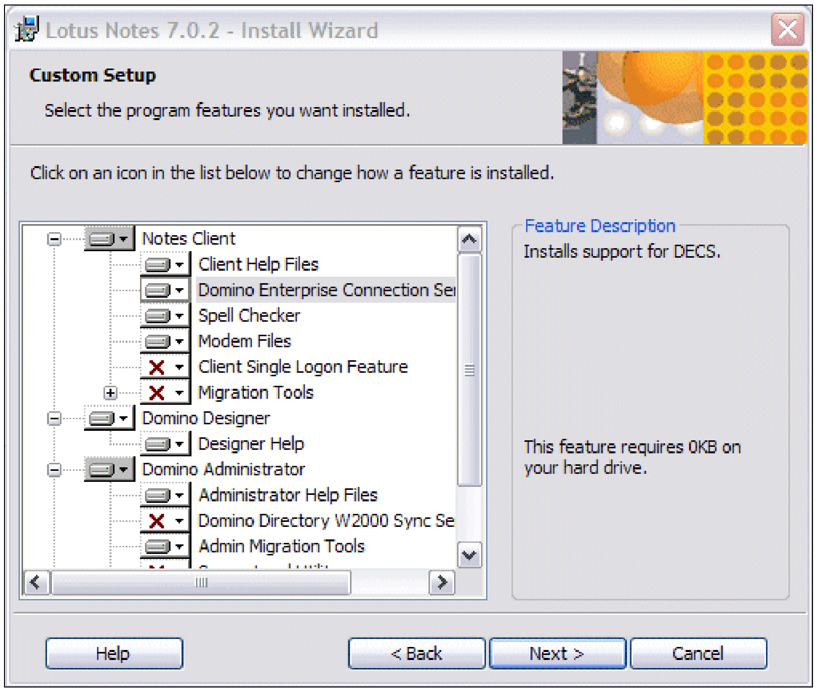
5.1 Troubleshooting installation issues

This section describes how to troubleshoot installation issues both on client and server side.

5.1.1 Troubleshooting client-side issues

Note the following client-side installation issues:

- Beginning with Notes client version 7.0.2, it is no longer optional to install the Domino Enterprise Connection Services files needed to run the Notes access for SAP solutions script libraries locally (Figure 5-1).



Custom Setup

Select the program features you want installed.

Click on an icon in the list below to change how a feature is installed.

Notes Client
Client Help Files
Domino Enterprise Connection Se
Spell Checker
Modem Files
Client Single Logon Feature
Migration Tools
Domino Designer
Designer Help
Domino Administrator
Administrator Help Files
Domino Directory W2000 Sync Se
Admin Migration Tools

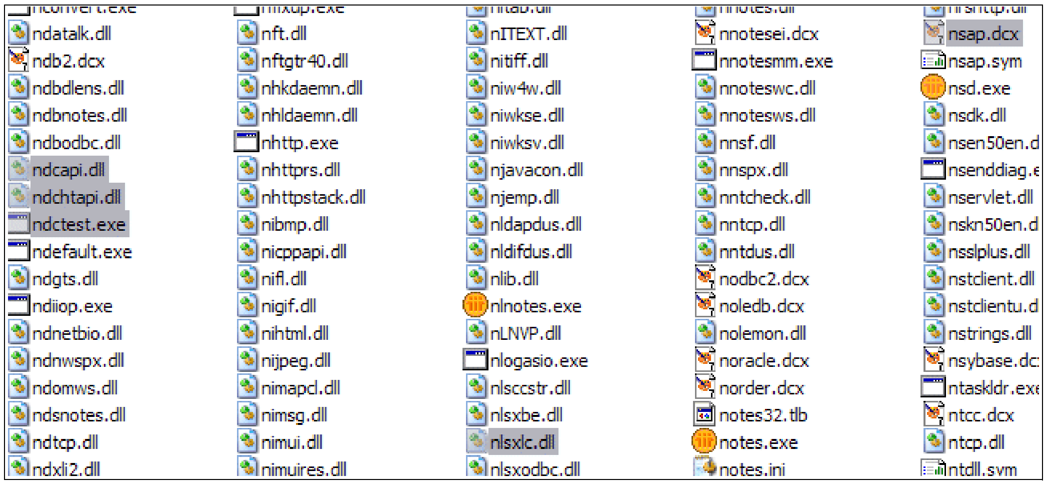
Feature Description

Installs support for DECS.

This feature requires 0KB on your hard drive.

Figure 5-1 Lotus Notes Setup options

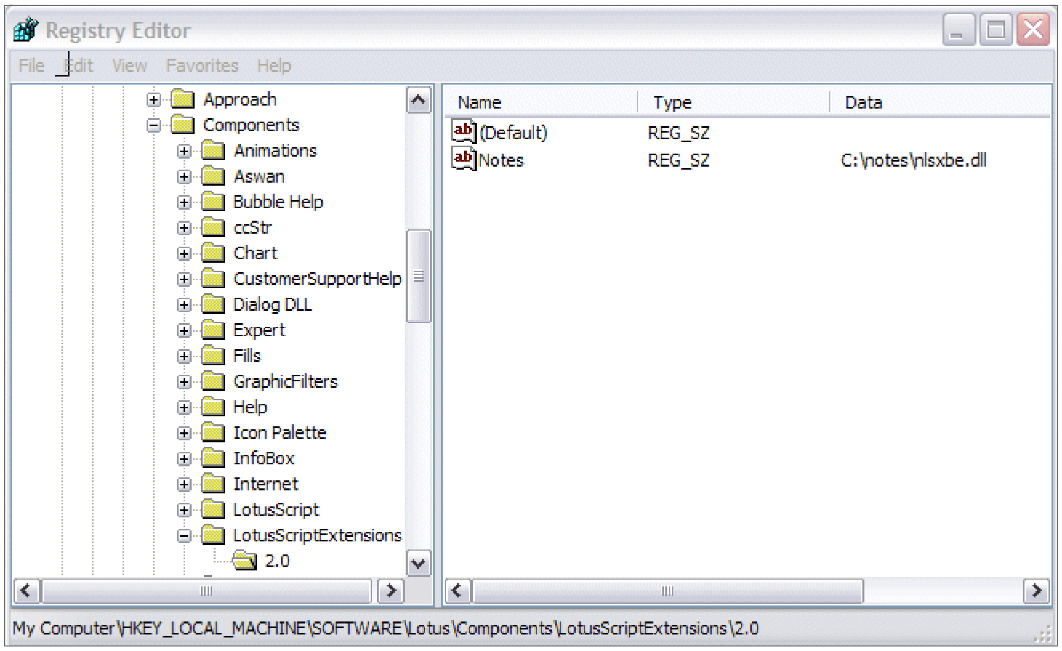
- Prior to Lotus Notes 7.0.2, you were able to unselect the files needed to run Notes access for SAP solutions scripts, resulting in error messages such as “Unable to create Product Object.” It is easy to confirm if the files highlighted in Figure 5-2 on page 93 are in the Notes program directory:
  - ndcapi.dll
  - ndchtapi.dll
  - ndctest.exe
  - nlsxlc.dll
  - nsap.dcx



ndatalk.dll	nft.dll	nITEXT.dll	nnotesei.dcx	nsap.dcx
ndb2.dcx	nftgtr40.dll	nitiff.dll	nnotesmm.exe	nsap.sym
ndbdlens.dll	nhkdaemn.dll	niw4w.dll	nnoteswc.dll	nsd.exe
ndbnotes.dll	nhldaemn.dll	niwkse.dll	nnotesws.dll	nsdk.dll
ndbodbc.dll	nhttp.exe	niwksv.dll	nnsf.dll	nsen50en.d
ndcapi.dll	nhttprs.dll	njavacon.dll	nnspx.dll	nsenddiag.e
ndchtapi.dll	nhttpstack.dll	njemp.dll	nntcheck.dll	nservlet.dll
ndctest.exe	nibmp.dll	nldapdus.dll	nntcp.dll	nskn50en.d
ndefault.exe	nicppapi.dll	nldifdus.dll	nntdus.dll	nsslplus.dll
ndgts.dll	nifl.dll	nlib.dll	nodbc2.dcx	nstclient.dll
ndiiop.exe	nigif.dll	nlnotes.exe	noledb.dcx	nstclientu.d
ndnetbio.dll	nihtml.dll	nLNVP.dll	nolemon.dll	nstrings.dll
ndnwspx.dll	nijpeg.dll	nlogasio.exe	noracle.dcx	nsybase.dc
ndomws.dll	nimapcl.dll	nlsccstr.dll	norder.dcx	ntaskldr.exe
ndsnotes.dll	nimsg.dll	nlsxbe.dll	notes32.tlb	ntcc.dcx
ndtcp.dll	nimui.dll	nlsxlc.dll	notes.exe	ntcp.dll
ndxli2.dll	nimuires.dll	nlsxodbc.dll	notes.ini	ntdll.sym

Figure 5-2 Files needed for Lotus Notes access for SAP solutions to work

- In addition to the Notes files, you must install the SAP client file librfc32.dll if you do not have the full SAP GUI client installed. This file must be obtained from SAP.
- The Redpaper team discovered that there can be some issues if there is more than one version of the Notes client installed on a workstation. Pointing to the correct client can sometimes be a problem. In most cases, the registry entry **HKEY_LOCAL_MACHINE** → **Software** → **Lotus** → **Components** → **LotusScriptExtensions** → **2.0** displays which client is being used (Figure 5-3). (To open the Registry Editor, select **Start** → **Run** and type `regedit`.)



Name	Type	Data
(Default)	REG_SZ	
Notes	REG_SZ	C:\notes\nlsxbe.dll

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Lotus\Components\LotusScriptExtensions\2.0

Figure 5-3 Registry Editor

5.1.2 Troubleshooting server-side issues

The Lotus Notes access for SAP solutions templates are originally installed through the template builder application on the Domino server. When something does not work, it is either a UI problem with a form or design element, or it is a back-end issue with one of the functions in the Notes access for SAP solutions script libraries. To troubleshoot and narrow down the issue, try the test agent database supplied with this paper (see Appendix B, “Additional material” on page 113). There is a test agent for every template. It is invaluable to run the test agent in debug mode to determine the values returned by the back end.

5.2 Debug levels

The LCConnection class also includes a property called debuglevel. There are two major values to set this to:

- 1: When set to 1, Lotus Notes access for SAP solutions creates an RFC trace file named rfcxxx_xxx.trc (where each x is a number) in the directory where the Connector is installed.
- 69: This invokes the SAP GUI ABAP debugger.

The SAPLoginWithOptions function in the SAPaccessUtilities script library enables us to take advantage of these options. As we see in the following line of code taken from this function, we can set a variable named NASS_DEBUG_LEVEL in our notes.ini file and turn on these debug parameters. (Of course, you can call this variable any way you want in your own applications, but let’s use what the Lotus Notes access for SAP solutions developers chose.)

```
level = s.GetEnvironmentValue("NASS_DEBUG_LEVEL")
```

NASS_DEBUG_LEVEL=1

We found it is helpful to generate the trace file when specific users had problems. Add this parameter to their notes.ini file, re-run the problem in Lotus Notes access for SAP solutions, and then send the trace file to the Notes administrator to troubleshoot.

NASS_DEBUG_LEVEL=69

From a developmental perspective, it was very useful to set the debug level to 69. When this is set, you drop right into the SAP GUI during debug. You can see exactly how the LotusScript code runs the BAPI.

For a detailed discussion of the debug level property, see the *Lotus Connector for SAP R/3 User Guide* at:

http://www-12.lotus.com/ldd/doc/SAP/1.7.2/sapdoc.nsf

5.3 Various SAP access difficulties

In developing the ITSO Bank vendor application, we discovered a few problems specific to SAP and data type issues. First, we describe the initial problem, followed by the solution. The goal is to show that working with SAP data can be tricky and that it is critical to have access to the SAP GUI client to troubleshoot problems. Coming from a background of SQL, we were always more comfortable using RFC_READ_TABLE to look at data in a table directly because it more closely resembles SQL ideas. BAPIs, as you will see in the following examples, really follow no rules regarding data format or calling structure. As we found through trial and error, you are at the mercy of the developer who wrote the BAPI.

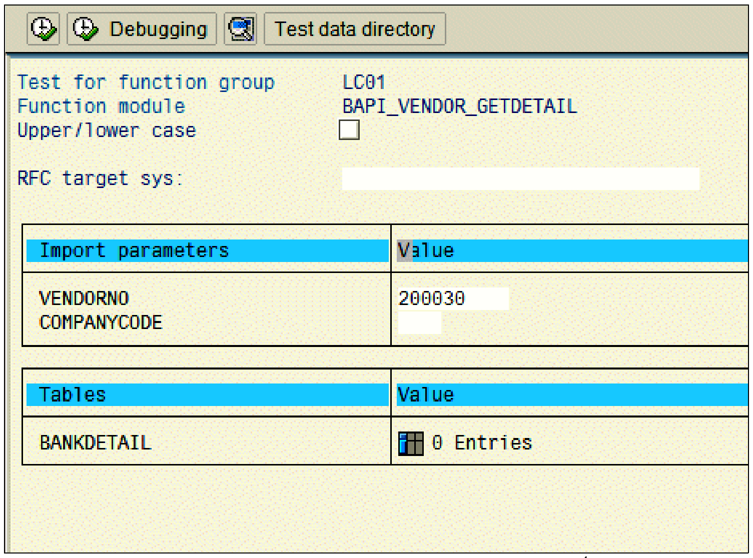
1000 for a 10 CHAR data type fails

Our first test agent to return data from the BAPI_VENDOR_GETDETAIL took some time to get working. We somehow came up with vendor number 200030 and company code 1000. Therefore, we plugged these in to our select statement to pass to BAPI_VENDOR_GETDETAIL and ran our script, as in Example 5-1. There were no errors, but also no data.

Example 5-1 Faulty code for retrieving vendor data

```
   selectstatement = |VENDORNO = "200030"|

   counter = source.Execute(selectstatement, sourcefieldlist)

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

Dim index As Long

While (source.Fetch(sourcefieldlist))

   Set field1 = sourcefieldlist.lookup("EXPORTSGENERALDETAILNAME",index)
   Set field2 = sourcefieldlist.lookup("EXPORTSGENERALDETAILSTREET",index)
   Set field3 = sourcefieldlist.lookup("EXPORTSGENERALDETAILCOUNTRYISO",index)

   Print "our data to cart to NaSs = " & field1.text(0)  & field2.text(0) &
field3.text(0)

Wend

Exit Sub
```

Next we ran BAPI_VENDOR_GETDETAIL in the SAP GUI client. We found through experimentation that COMPANYCODE was not needed, so we just entered 200030 in the VENDORNO and clicked the **Watch** icon at the top left (Figure 5-4).



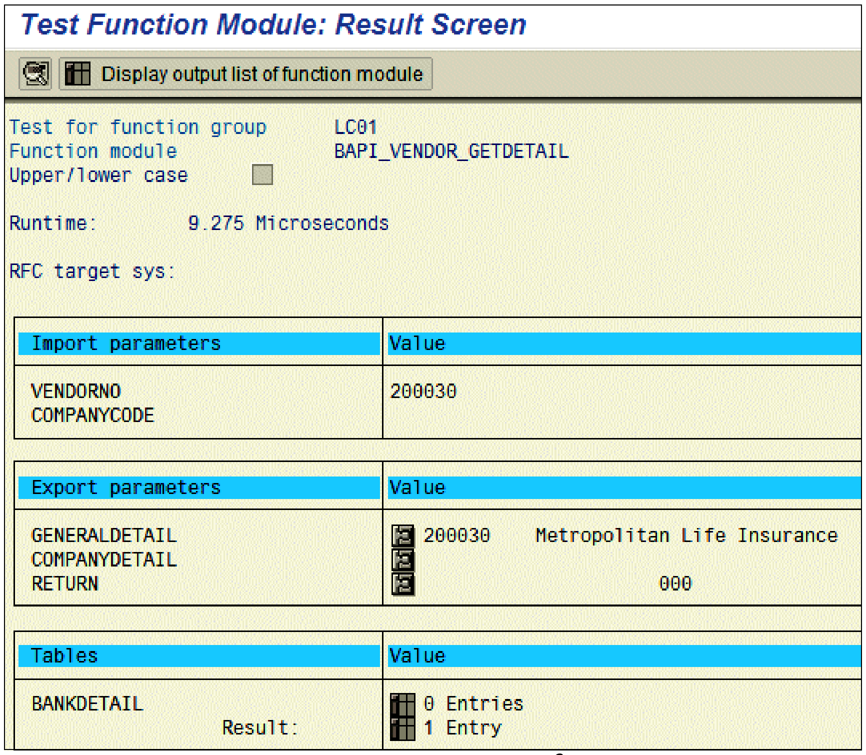
```
Test for function group      LC01
Function module              BAPI_VENDOR_GETDETAIL
Upper/lower case             ☐

RFC target sys:
```

Import parameters	Value
VENDORNO	200030
COMPANYCODE	

Tables	Value
BANKDETAIL	0 Entries

Figure 5-4 BAPI_VENDOR_GETDETAIL in SAP GUI[1]

The BAPI ran and we saw that we got one value back (Figure 5-5). Therefore, this works in SAP, but not from our LotusScript. This suggests that something is wrong with our select statement.


Test Function Module: Result Screen

Display output list of function module

```
Test for function group      LC01
Function module              BAPI_VENDOR_GETDETAIL
Upper/lower case

Runtime:         9.275 Microseconds

RFC target sys:
```

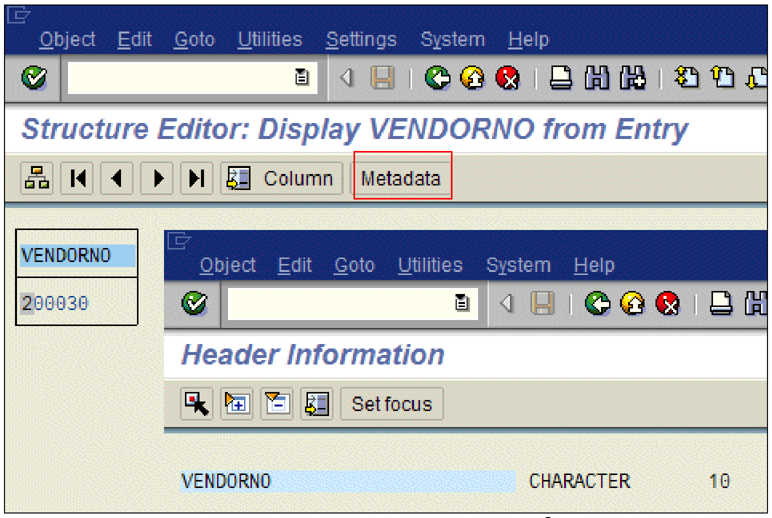
Import parameters	Value
VENDORNO	200030
COMPANYCODE	

Export parameters	Value
GENERALDETAIL	200030 Metropolitan Life Insurance
COMPANYDETAIL	
RETURN	000

Tables	Value
BANKDETAIL	0 Entries
Result:	1 Entry

Figure 5-5   BAPI_VENDOR_GETDETAIL result set[2]

After a bit of searching, we found that you can double-click the word **VENDORNO** in Import Parameters. This led us to the Structure Editor. When we clicked the **Metadata** button there, the window shown in Figure 5-6 opened.


Object Edit Goto Utilities Settings System Help
Structure Editor: Display VENDORNO from Entry
Column
Metadata
VENDORNO
200030
Object Edit Goto Utilities System Help
Header Information
Set focus

```
VENDORNO                      CHARACTER        10
```

Figure 5-6   Structure Editor entry for VENDORNO[3]

[1] © SAP AG 2006. All rights reserved.
[2] © SAP AG 2006. All rights reserved.
[3] © SAP AG 2006. All rights reserved.

We see here that VENDORNO is a CHARACTER 10. This gave us an idea that VENDORNO might require 10 characters to work. So, we wrote our script with 0000200030 instead of 200030 (Example 5-2).

Example 5-2 Revised script for retrieving vendor data

```
‘selectstatement = |VENDORNO = "200030"|
‘This did not work and after a look at the metadata
‘definition of VENDORNO in SAP we decided to try with leading
‘zeros to get 10 characters - IT WORKED!

selectstatement = |VENDORNO = "0000200030"|

   counter = source.Execute(selectstatement, sourcefieldlist)

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

Dim index As Long

While (source.Fetch(sourcefieldlist))
```

We ran our script and it worked. We got one value returned in Notes, as we did in SAP.

Note two things from this exercise. One, it is critical to think carefully about data types. Two, after looking through the script libraries, we found that there is a PadZerosLeft function within the SAPaccessUtilities library, and there are lots of additional functions to handle other issues as well. The Lotus Notes access for SAP solutions developers put in several useful functions to handle issues such as this one. Therefore, in some cases, the functionality is already built into the Notes access for SAP solutions libraries. Usually, the connector software that Lotus Notes access for SAP solutions uses does all the necessary conversions for us in the background, but sometimes you still have to check manually if something goes wrong.

Properly accessing an import within a structure

While developing our ITSO Bank application, we came across the following BAPI_USER_EXISTENCE_CHECK and realized we have not yet described how to access a structure. We include this example as an extra. In this case, the agent failed because, unlike a table or a character string, the structure has one further layer to reference. Therefore, if we look at BAPI_USER_EXISTENCE_CHECK, as shown in Example 5-3 on page 98, we see that all that is needed to run this BAPI in the SAP GUI is to enter `DDIC` in its one import parameter. Running the LotusScript agent failed when we used:

```
selectstatement = |BAPIBNAME= "DDIC"|
```

Further experimentation showed that all that was necessary to resolve the problem was to use dot notation to properly reference the required input variable:

```
selectstatement = |USERNAME.BAPIBNAME= "DDIC"|
```

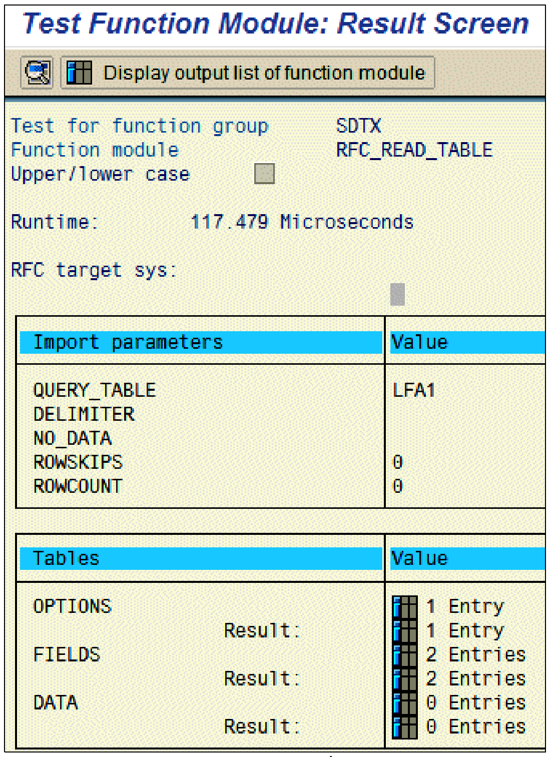
Example 5-3 BAPI_USER_EXISTENCE_CHECK

```
Option Public
Option Declare
Uselsx "*lsxlc"

Sub Initialize

   On Error Goto errorhandler

   Dim session As New Lcsession

   Dim source As New Lcconnection("sap")

   Dim sourcefieldlist As New Lcfieldlist

   Dim counter As Integer
   Dim selectstatement As String

   'now that I see what I want add enough field for out put

   Dim field1 As lcfield
   Dim field2 As lcfield
   Dim field3 As lcfield

   source.Database = "BAPI_USER_EXISTENCE_CHECK"

   source.Userid = "muster"
   source.Password = "ides"
   source.Client = "800"
   source.SystemNo = 0
   source.Language = "EN"
   source.Server= "your.server.org.com"
   source.Destination = "LD3"
   source.debuglevel = 0

   source.Connect

   source.Metadata = "*"

                ' Just copying the select staement for other tests such as
   'selectstatement = |VENDORNO = "0000001000"| + |COMPANYCODE = "1000"|
                ' lead us to try
                 'Selectstatement = |BAPIBNAME= "DDIC"|
   ' this failed because the input is actually a structure USERNAME and thus we
needed the following
                ' to properly call this BAPI

   selectstatement = |USERNAME.BAPIBNAME= "DDIC"|


   counter = source.Execute(selectstatement, sourcefieldlist)

   If (counter = 0) Then
      Msgbox "No records returned by R/3"
      Exit Sub
   End If


   Dim index As Long

   While (source.Fetch(sourcefieldlist))
      'Set field1 = sourcefieldlist.lookup("gobbledeguck",index)

      'replace gobbledeguck with meaningful SAP exports gained
      'from cheating the names from the debugger

                ' Using the same technique with the debugger we found that 
EXPORTSRETURNMESSAGE
                ' was what we needed hence the following

      Set field1 = sourcefieldlist.lookup("EXPORTSRETURNMESSAGE",index)

      Print "message= " & field1.text(0)

   Wend

   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

End Sub
```

Tables use % as wildcard versus BAPIs using *

We had some trouble searching with wildcards until we realized that to search tables with RFC_READ_TABLE, you need to use a percent sign (%), much like regular SQL. To search in other BAPIs, we had to use an asterisk (*) however.

Note that in Figure 5-7 there are no result entries in the DATA table. This is because we used an asterisk to search in the OPTIONS table, as shown in Figure 5-8.



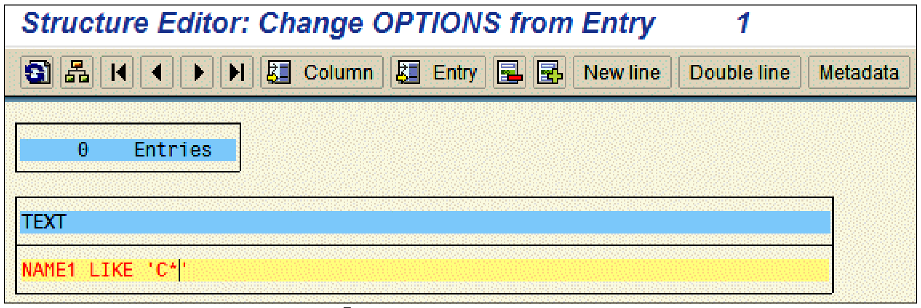
Test Function Module: Result Screen

```
Test for function group      SDTX
Function module              RFC_READ_TABLE
Upper/lower case

Runtime:        117.479 Microseconds

RFC target sys:
```

Import parameters	Value
QUERY_TABLE	LFA1
DELIMITER	
NO_DATA	
ROWSKIPS	0
ROWCOUNT	0

Tables		Value
OPTIONS		1 Entry
	Result:	1 Entry
FIELDS		2 Entries
	Result:	2 Entries
DATA		0 Entries
	Result:	0 Entries

Figure 5-7 DATA table empty[4]



Structure Editor: Change OPTIONS from Entry 1

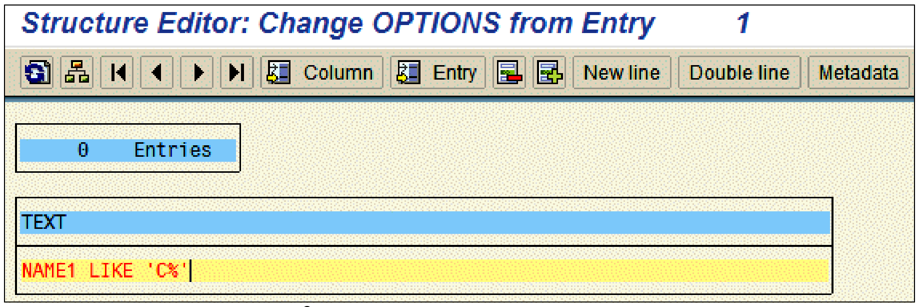
0 Entries

TEXT
NAME1 LIKE 'C*'

Figure 5-8 Search not working[5]

4 © SAP AG 2006. All rights reserved.
5 © SAP AG 2006. All rights reserved.

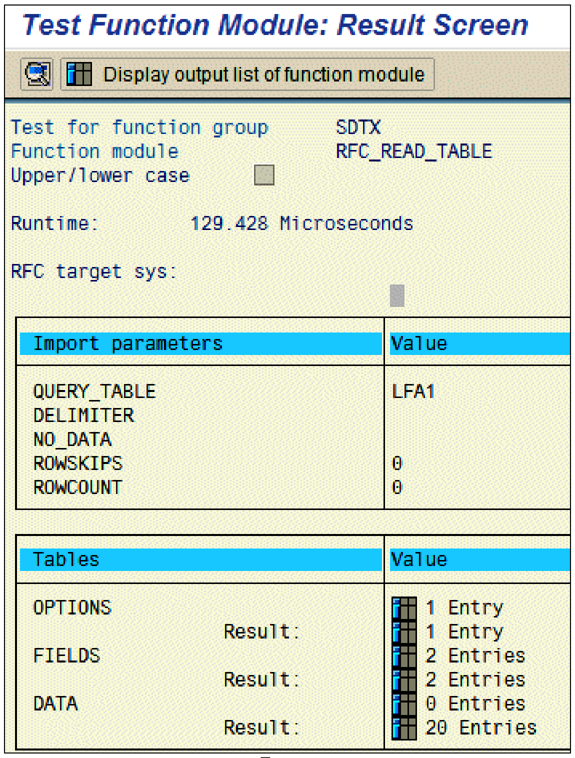
Now, we change the * to a % in our search string, as shown in Figure 5-9, and execute **RFC_READ_TABLE** with the new search string.



```
0    Entries

TEXT
NAME1 LIKE 'C%'
```

Figure 5-9 Search working[6]

We now get what we expected. We now see that there are 20 entries in the DATA table result set (Figure 5-10).


Test Function Module: Result Screen
Display output list of function module

```
Test for function group      SDTX
Function module              RFC_READ_TABLE
Upper/lower case

Runtime:         129.428 Microseconds

RFC target sys:
```

Import parameters	Value
QUERY_TABLE	LFA1
DELIMITER	
NO_DATA	
ROWSKIPS	0
ROWCOUNT	0

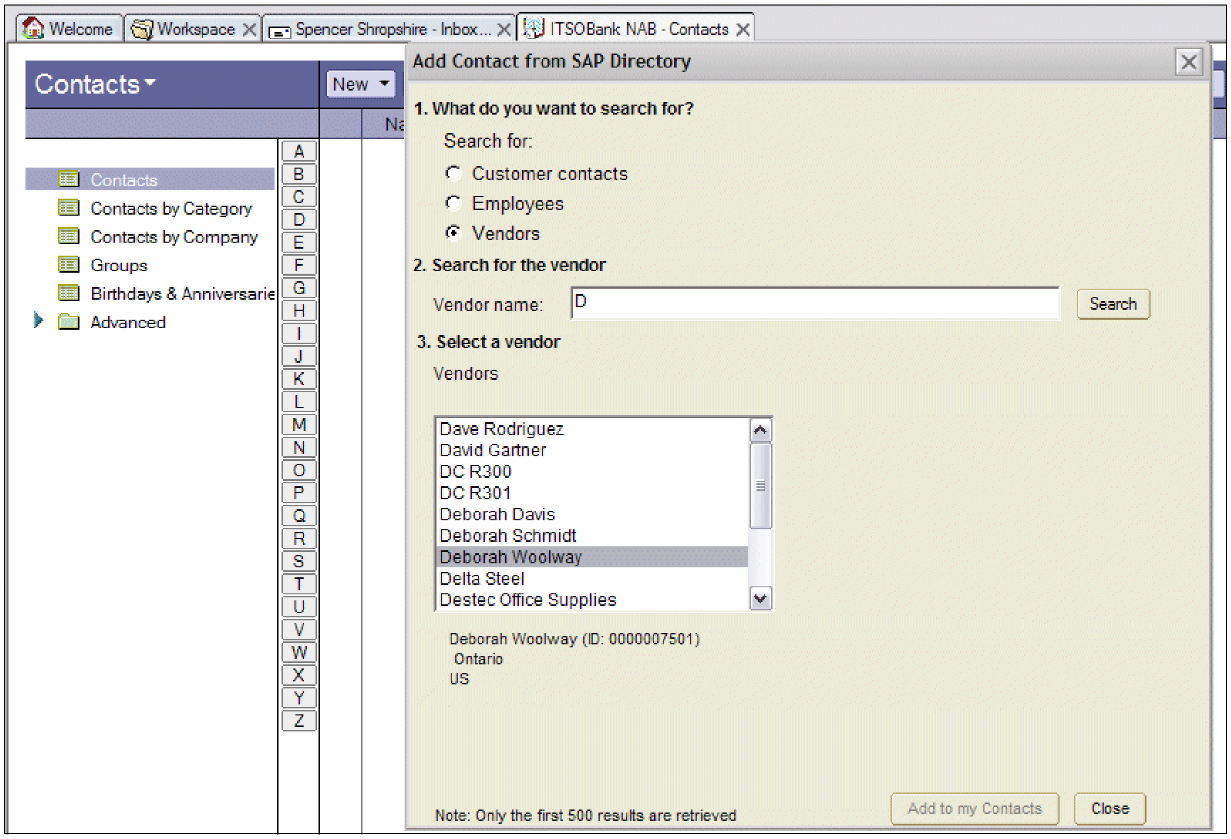
Tables	Value
OPTIONS	1 Entry
Result:	1 Entry
FIELDS	2 Entries
Result:	2 Entries
DATA	0 Entries
Result:	20 Entries

Figure 5-10 Result Set[7]

[6] © SAP AG 2006. All rights reserved.
[7] © SAP AG 2006. All rights reserved.

Actual data can cause unexpected results

During the development of the vendor application, we also encountered some other unexpected results. For example, when you choose D as a search string and select Deborah Woolway, the word Ontario does not line up properly, as displayed in Figure 5-11. In addition, we notice that the country is US, while Ontario is in Canada.



1. What do you want to search for?

Search for:

( ) Customer contacts
( ) Employees
[✓ Vendors]

2. Search for the vendor

Vendor name: D

3. Select a vendor

Vendors

Dave Rodriguez
David Gartner
DC R300
DC R301
Deborah Davis
Deborah Schmidt
Deborah Woolway
Delta Steel
Destec Office Supplies

```
Deborah Woolway (ID: 0000007501)
 Ontario
US
```

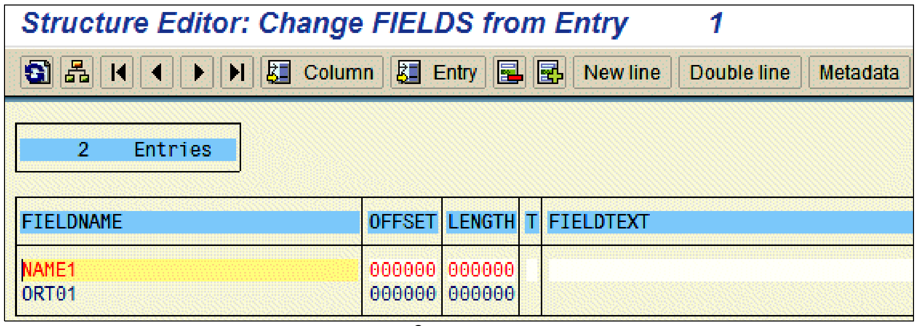
Note: Only the first 500 results are retrieved

Figure 5-11 Looking for a vendor

So what is going wrong here? Because all this data is stored in the LFA1 table, we decide to look up the actual data values in the table. We use transaction SE37 to run RFC_READ_TABLE. We entered parameters to search for companies that start with D and return the columns for company name (NAME1) and City (ORT01).

In addition to setting the QUERY_TABLE, we also set one OPTIONS and two FIELDS parameters. In setting the search string for the OPTIONS table, as we described earlier, we had to use %.

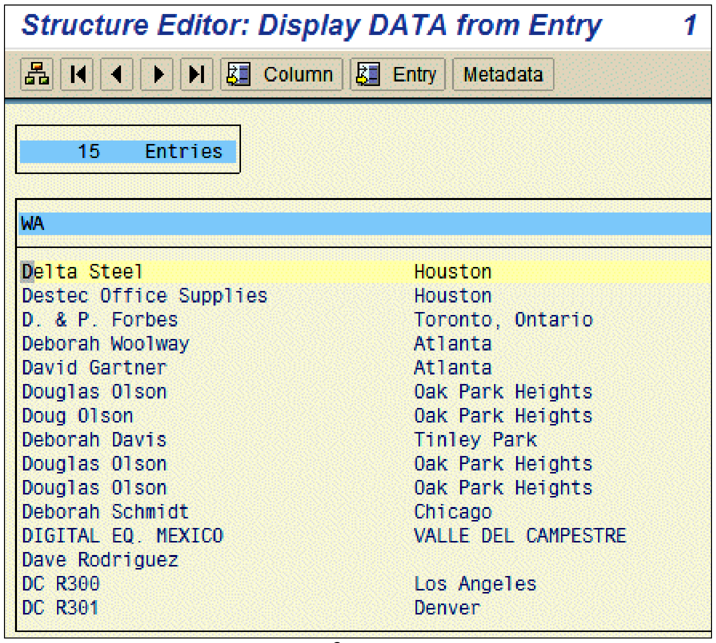
We set the column names that we want to be returned by RFC_READ_TABLE (Figure 5-12). Remembering as well that ORT is German for city, we choose **NAME1** for company and **ORT01** for city in the FIELDS table.



FIELDNAME	OFFSET	LENGTH	T	FIELDTEXT
NAME1	000000	000000		
ORT01	000000	000000		

Figure 5-12 Setting the FIELDS table[8]

We click the clock icon to run RFC_READ_TABLE, and we get 15 results. We click the **15 Entries** icon and see the following results set (Figure 5-13).

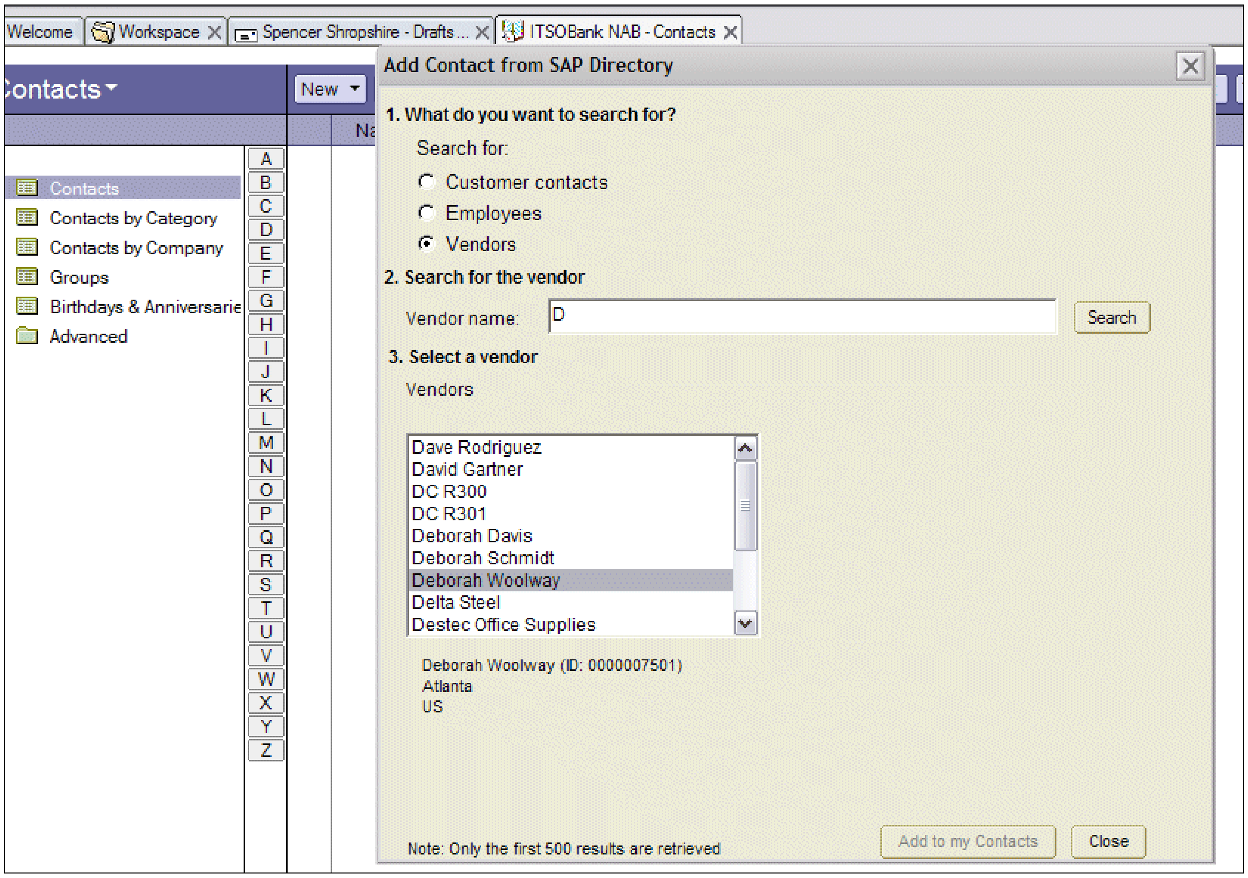


```
WA
Delta Steel                        Houston
Destec Office Supplies             Houston
D. & P. Forbes                     Toronto, Ontario
Deborah Woolway                    Atlanta
David Gartner                      Atlanta
Douglas Olson                      Oak Park Heights
Doug Olson                         Oak Park Heights
Deborah Davis                      Tinley Park
Douglas Olson                      Oak Park Heights
Douglas Olson                      Oak Park Heights
Deborah Schmidt                    Chicago
DIGITAL EQ. MEXICO                 VALLE DEL CAMPESTRE
Dave Rodriguez
DC R300                            Los Angeles
DC R301                            Denver
```

Figure 5-13 Result set details[9]

[8] © SAP AG 2006. All rights reserved.
[9] © SAP AG 2006. All rights reserved.

Careful examination reveals our problem. Look at the value of ORT01 above the Deborah Woolway entry. It is the only one with two city entries, and there is a comma in between. We found the problem! In our UI, we take the values and separate the string values by commas. Therefore, it should be a simple fix to change our separators to some other character. We chose $ and implemented the code change. As we see in Figure 5-14, we fixed the problem.



Add Contact from SAP Directory

1. What do you want to search for?

Search for:

( ) Customer contacts
( ) Employees
(•) Vendors

2. Search for the vendor

Vendor name: D

Search

3. Select a vendor

Vendors

Dave Rodriguez
David Gartner
DC R300
DC R301
Deborah Davis
Deborah Schmidt
Deborah Woolway
Delta Steel
Destec Office Supplies

Deborah Woolway (ID: 0000007501)
Atlanta
US

Note: Only the first 500 results are retrieved

Add to my Contacts

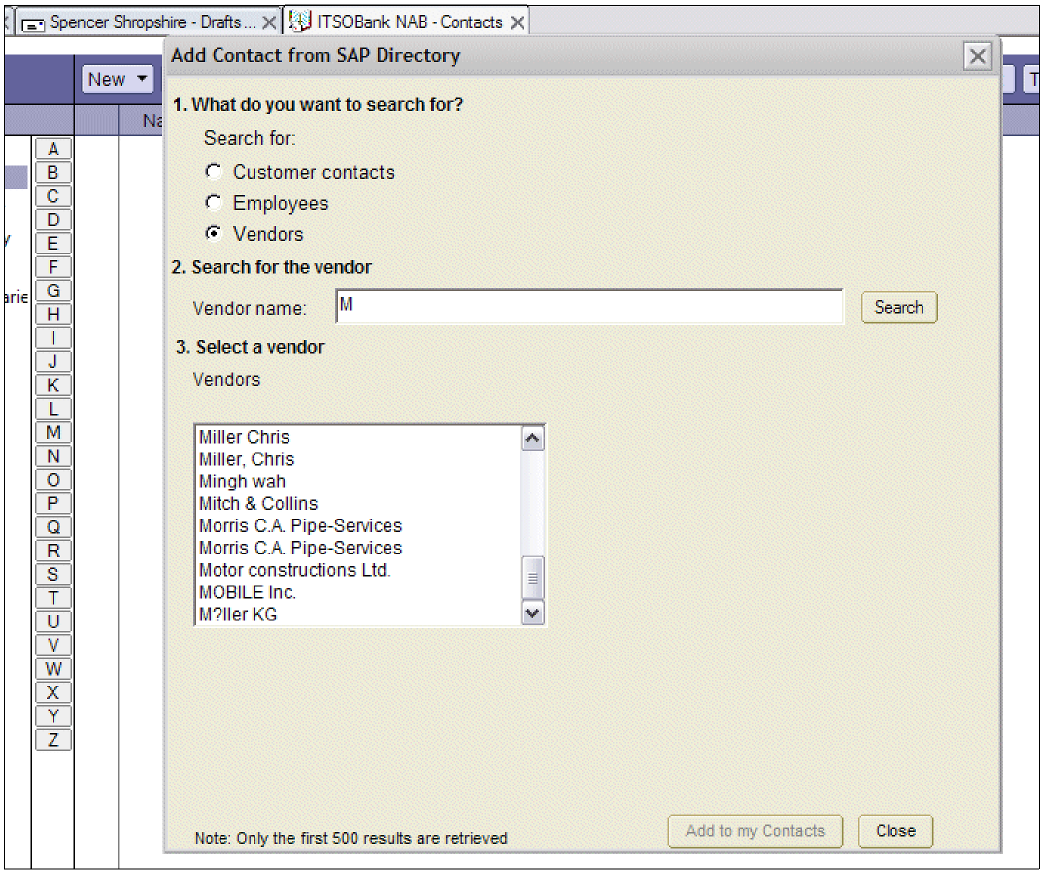
Close

Figure 5-14 Correct vendor address data

Note: The error discussed resulted by choosing a string with commas as separators in our data structure. This problem is avoided by the Lotus Notes access for SAP solutions templates because they use array for multi-value data. In addition, there are several useful functions that handle empty values and other problems that can occur when storing the SAP data into the SAPMetaData object.

Character set issues

In addition to the issues listed earlier, we also had a character set issue for our German customers. As you can see in Figure 5-15, the umlaut for Müller shows up as M?ller.


Add Contact from SAP Directory

1. What do you want to search for?

Search for:
- Customer contacts
- Employees
- [✓ Vendors]

2. Search for the vendor

Vendor name: M

3. Select a vendor

Vendors

```
Miller Chris
Miller, Chris
Mingh wah
Mitch & Collins
Morris C.A. Pipe-Services
Morris C.A. Pipe-Services
Motor constructions Ltd.
MOBILE Inc.
M?ller KG
```

Note: Only the first 500 results are retrieved

Spencer Shropshire - Drafts ...
ITSOBank NAB - Contacts
New
Search
Add to my Contacts
Close
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

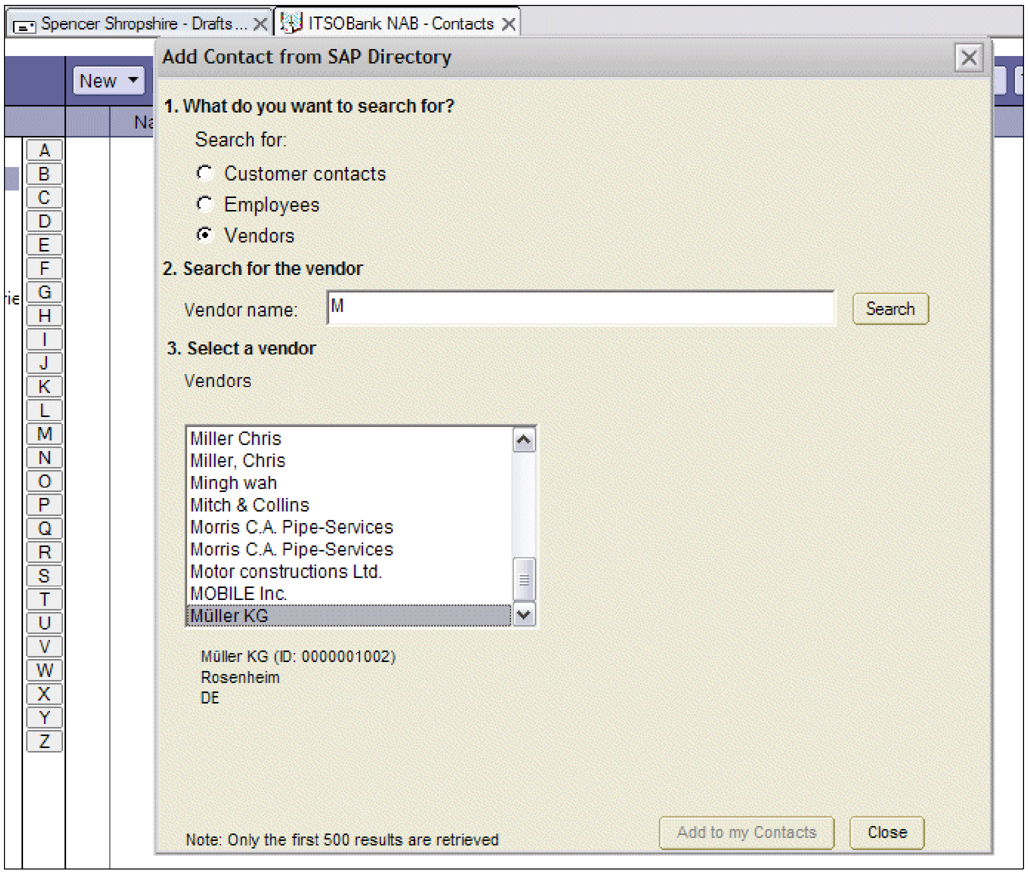
Figure 5-15   Character set problem

By default, the SAP connector is set to the UTF-8 character set. However, there is a property of the SAP connector, sapreadcodepage, that enables us to set the character set appropriately for our needs. After experimenting with some of the available character sets, we added the following line in Example 5-4 to our LCConnection properties file for logging on to the SAP server.

Example 5-4   Setting the correct code page in LotusScript

```
source.Userid = "muster"
source.Password = "ides"
source.Client = "800"
source.SystemNo = 0
source.Language = "EN"
source.Server= "yoursap.server.com"
source.Destination = "LD3"
'source.debuglevel = 1
'the default charset of the SAP connector is UTF8
source.SAPReadCodePage=LCSTREAMFMT_NATIVE
```

Therefore, by simply resetting the code page to native, we now see the correct German characters (Figure 5-16).


Spencer Shropshire - Drafts ...
ITSOBank NAB - Contacts
New
Add Contact from SAP Directory
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

1. What do you want to search for?

Search for:

( ) Customer contacts
( ) Employees
(•) Vendors

2. Search for the vendor

Vendor name: M [Search]

3. Select a vendor

Vendors

Miller Chris
Miller, Chris
Mingh wah
Mitch & Collins
Morris C.A. Pipe-Services
Morris C.A. Pipe-Services
Motor constructions Ltd.
MOBILE Inc.
Müller KG

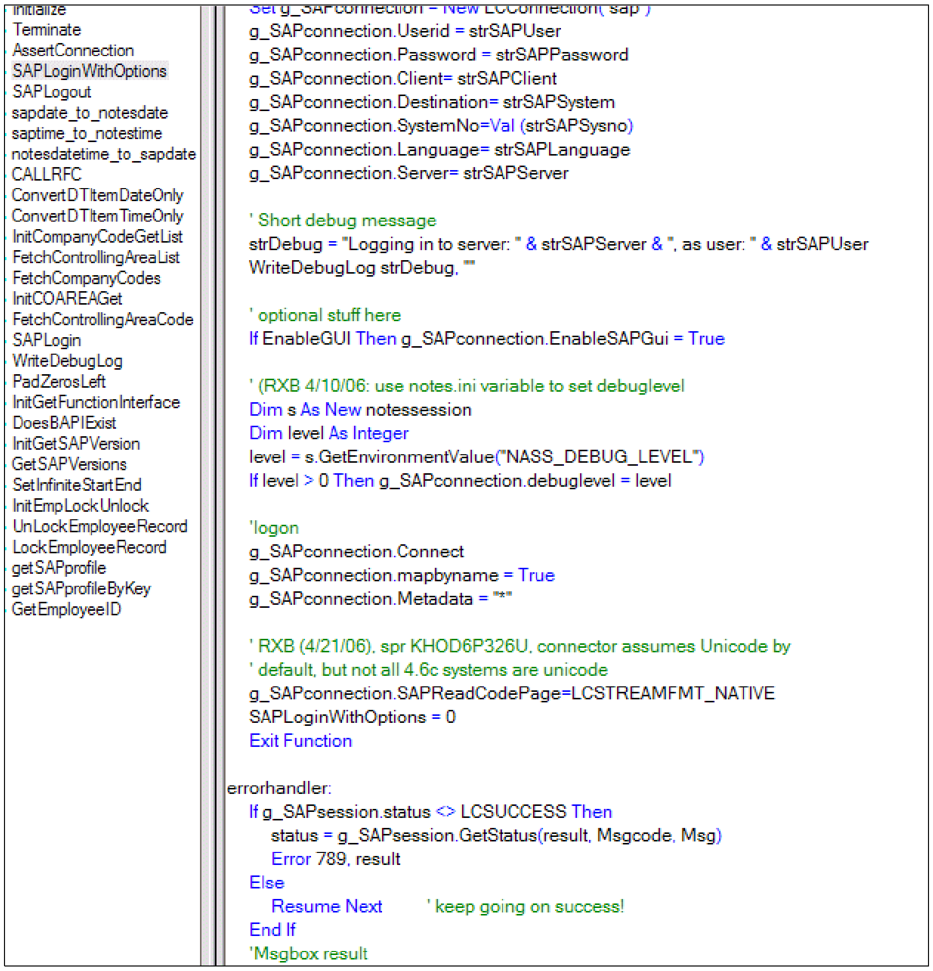
Müller KG (ID: 0000001002)
Rosenheim
DE

Note: Only the first 500 results are retrieved

[Add to my Contacts] [Close]

Figure 5-16 Character set problem fixed

In the ITSO Bank vendor example, we added an extra line to our script. In order to do this with the Lotus Notes access for SAP solutions template, you need to find where this is optimally located. The SAPLogonWithOptions function is in the SAPaccessUtilities library, as shown in Figure 5-17. Within this function, we can see where it is intended for customizations to be added. By default, the Lotus Notes access for SAP solutions template sets the SAP code page to native as well.



```
    g_SAPconnection.Userid = strSAPUser
    g_SAPconnection.Password = strSAPPassword
    g_SAPconnection.Client= strSAPClient
    g_SAPconnection.Destination= strSAPSystem
    g_SAPconnection.SystemNo=Val (strSAPSysno)
    g_SAPconnection.Language= strSAPLanguage
    g_SAPconnection.Server= strSAPServer

    ' Short debug message
    strDebug = "Logging in to server: " & strSAPServer & ", as user: " & strSAPUser
    WriteDebugLog strDebug, ""

    ' optional stuff here
    If EnableGUI Then g_SAPconnection.EnableSAPGui = True

    ' (RXB 4/10/06: use notes.ini variable to set debuglevel
    Dim s As New notessession
    Dim level As Integer
    level = s.GetEnvironmentValue("NASS_DEBUG_LEVEL")
    If level > 0 Then g_SAPconnection.debuglevel = level

    'logon
    g_SAPconnection.Connect
    g_SAPconnection.mapbyname = True
    g_SAPconnection.Metadata = "*"

    ' RXB (4/21/06), spr KHOD6P326U, connector assumes Unicode by
    ' default, but not all 4.6c systems are unicode
    g_SAPconnection.SAPReadCodePage=LCSTREAMFMT_NATIVE
    SAPLoginWithOptions = 0
    Exit Function

errorhandler:
    If g_SAPsession.status <> LCSUCCESS Then
        status = g_SAPsession.GetStatus(result, Msgcode, Msg)
        Error 789, result
    Else
        Resume Next        ' keep going on success!
    End If
    'Msgbox result
```

Figure 5-17 SAPLoginWithOptions function

A

Useful SAP information

The following sections contain key SAP information that will aid you with your SAP integration efforts:

- Glossary
- SAP connection/login information
- SAP transactions

Glossary

Table A-1 provides a glossary of SAP terms.

Table A-1 Useful SAP terminology

Term	Definition
ABAP	Advanced Business Application Programming. This is the language for developing applications in SAP. It used to be procedural and interpreted, but since version 4 it allows for object-oriented structures also.
BAPI	Business Application Programming Interface. An R/3 interface for integrating third-party software with SAP. BAPIs represent methods that can be applied to SAP business objects.
IDES	International Demonstration and Education System. A preconfigured SAP system, loaded with the data of a fictitious company for testing purposes. As with many SAP abbreviations, the definition has changed in the course of time, it has also been “Internal...” and “Internet...” instead of “International” and “Evaluation” instead of “Education.”
RFC	Remote Function Call. An SAP interface that calls SAP software functions from another computer that is connected over a network. Other vendors call this technique Remote Procedure Call (RPC)
Structure	An SAP structure is a data object that is made up of one or more components (simple ones such as strings or complex ones such as structures). This allows for easy processing of complex objects while still retaining the possibility to look at each of the components in detail if need be.

SAP connection/login information

Table A-2 lists the pieces of information you need to gather from your SAP administrator.

Table A-2 SAP connection/login information

Item	Recommended value
System number	This value is usually set to 00.
SAP server	Enter your SAP server’s IP address or DNS host name here. Make sure that you do not have any firewall issues.
User ID	(If not provided by your SAP administrator, try muster on an IDES system.)
Password	(If not provided by your SAP administrator, try ides on an IDES system.)
Client number	This value is usually set to 800.
Language	(If not provided by your SAP administrator, try EN.)
Destination	(If not provided by your SAP administrator, try LD3.)

SAP transactions

Note the following SAP transactions.

Data dictionary SE11

This is where the central data structures of any SAP system are stored. Of course, ABAP developers can keep their custom data definitions local (in the module the developer is writing), but it is generally considered to be good style to store them centrally in the ABAP data dictionary. This facilitates reuse and increases transparency for other developers, reducing overall errors.

Function builder SE37

This is the ABAP workbench, where all ABAP code is developed. It is the development environment for an SAP developer to design and test code.

Seen from a Notes perspective, it is the universal tool to closely examine RFC-enabled modules, their input/output parameters, structures, and tables. As laid out in many sections in this paper, proper development, even on the Notes side, often would be impossible without the function builder and its valuable insights.

Data browser SE16

Here you can browse all the SAP tables to which you have access. It is often a last resort when data problems arise during development and all other troubleshooting has failed. However, it sometimes also a good starting point if you need to find out what data is provided by a given table (and what is not).

B

Additional material

This Redpaper refers to additional material that can be downloaded from the Internet as described here.

Locating the Web material

The Web material associated with this Redpaper is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/REDP4215

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the Redpaper form number, REDP4215.

Using the Web material

The additional Web material that accompanies this Redpaper includes the following files:

File name	*Description*
ITSOBankNAB.nsf	Notes database used throughout the paper. Contains all example code.
NassTestAgents2.nsf	Notes database for experienced application developers.

How to use the Web material

Put the databases in your Notes data directory on your workstation and enjoy experimenting.

Note that in the ITSOBankNAB database, in the code for the Search button you will notice the onclick event as shown in Example B-1.

For the purposes of this paper, we used the searchVends2 function for the example development in Chapter 3, “Customization” on page 35 and used the searchVends function for the development in Chapter 4, “Extending” on page 63.

Example: B-1 Search button code

```
Sub Click(Source As Button)
   Dim s As New NotesSession
   Dim ws As New NotesUIWorkspace
   Dim thisdb As NotesDatabase
   Dim thisdoc As NotesDocument, thisuidoc As NotesUIDocument
   Dim retval As Long
   Dim sapflds List As Variant

   Set thisdb = s.CurrentDatabase
   Set thisuidoc = ws.CurrentDocument
   Set thisdoc = thisuidoc.Document

   retval = getSAPProfile(sapflds)
   retval = SAPlogin(sapflds)


   If Lcase(thisdoc.GetItemValue("SearchFor")(0)) = "customers" Then
      Call searchContacts(thisdoc)
   End If

   If Lcase(thisdoc.GetItemValue("SearchFor")(0)) = "employees" Then
      Call searchEmps(thisdoc)
   End If

   If Lcase(thisdoc.GetItemValue("SearchFor")(0)) = "vendors" Then
      Call searchVends2(thisdoc)
   End If
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

IBM Redbooks

For information about ordering these publications, see "How to get IBM Redbooks" on page 116. Note that some of the documents referenced here may be available in softcopy only.

- *Domino Designer 6: A Developer's Handbook*, SG24-6854
- *Implementing IBM Lotus Enterprise Integrator 6*, SG24-6067
- *Lotus Domino 7 Application Development*, REDP-4102
- *Notes and Domino Connectivity - A Collection of Examples*, REDP-0115
- *Understanding Lotus Notes Smart Upgrade*, REDP-4180

Other publications

These publications are also relevant as further information sources:

- *Introducing IBM Lotus Notes Access for SAP Solutions*

  `http://www.lotusadvisor.com/doc/18206`
- *IBM Lotus Notes and Lotus Domino integration technologies for SAP software*

  `ftp://ftp.software.ibm.com/software/lotus/lotusweb/product/SAPconnector/SAP_integration_white_paper.pdf`
- *Lotus Connector for SAP R/3 User Guide*

  `http://www-12.lotus.com/ldd/doc/SAP/1.7.2/sapdoc.nsf`
- *Lotus Connector LotusScript Extensions Guide*

  `http://www-12.lotus.com/ldd/doc/lei/6.5/lsxlc6.nsf`

Online resources

These Web sites are also relevant as further information sources:

- IBM Lotus downloads on IBM developerWorks

  `http://www.ibm.com/developerworks/lotus/downloads/`
- IBM Lotus Notes access for SAP solutions product page

  `http://www.ibm.com/software/sw-lotus/products/product4.nsf/wdocs/notesforsap`
- To learn more about IBM Lotus Connector for SAP Solutions software

  `http://ibm.com/lotus/sapconnector`

- To learn more about IBM Lotus Enterprise Integrator

  `http://ibm.com/lotus/lei`
- Accessing Relational Data with the Lotus Connectors LSX

  `http://www.lotus.com/ldd/sandbox.nsf/ecc552f1ab6e46e4852568a90055c4cd/04059282425693c385256e51006cd9e1?OpenDocument&Highlight=0,ad208`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM®

Lotus Notes access for SAP solutions

Redpaper

Out-of-the-box SAP integration for your Lotus Notes users

Advanced customization techniques

Additional integration features

IBM Lotus Notes access for SAP solutions can provide even greater business value for companies using both Lotus Notes software and SAP enterprise systems. This feature lets you access SAP information and business processes in your familiar Lotus Notes messaging and collaborative environment, starting with Lotus Notes 7.0.1. The Lotus Notes access for SAP solutions feature builds on proven IBM and SAP integration technology that has been on the market for eight years. The new capabilities extend this integration into calendars and scheduling, contact management, workflow processing, and other common business tasks. This feature leverages Lotus Notes strengths in personal information and workflow management to complement SAP application processes.

IT architects, application developers, and business leaders in an organization looking to integrate SAP business applications and workflow into their familiar, security-rich Lotus Notes collaboration client at no additional charge will want to read this Redpaper.