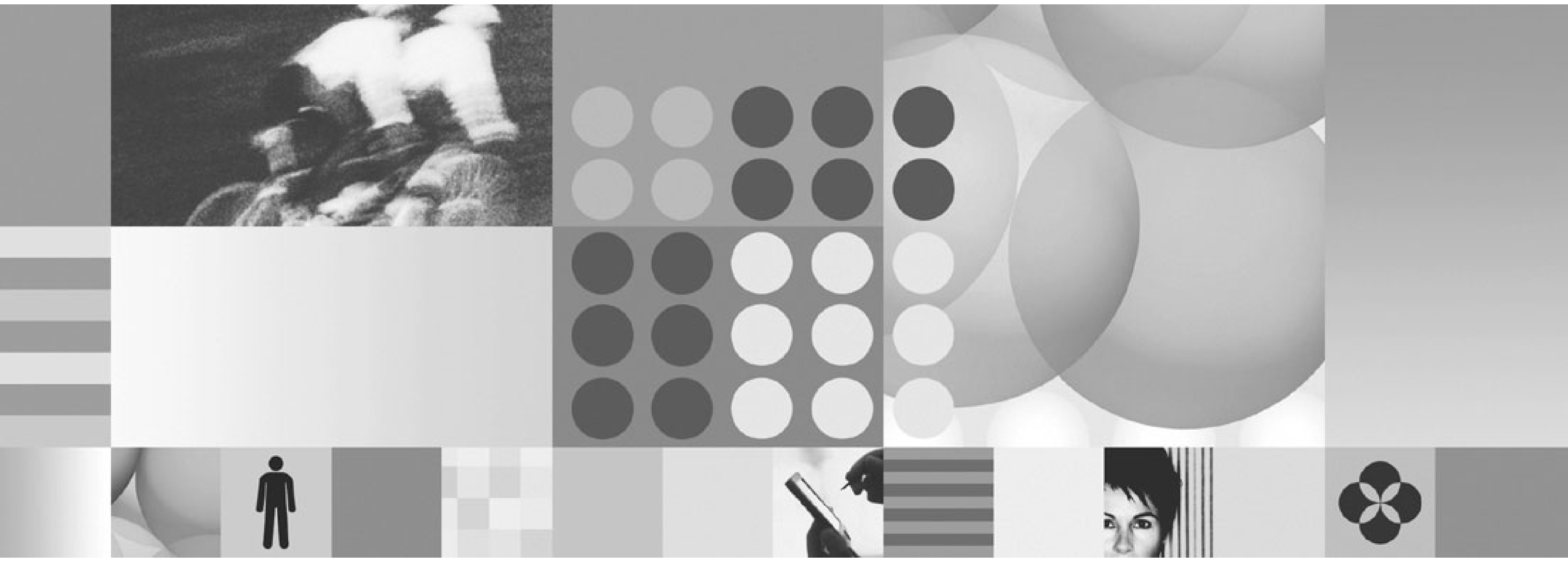
Lotus. Software

IBM

Composite Applications in Notes - Benefits and Technical Overview

Lotus® Software



Composite Applications in Notes - Benefits and Technical Overview

Note

Before using this information and the product it supports, read the information in Appendix C, “Notices ,” on page 71.

Contents

About this book

This document is designed to illustrate how businesses can accelerate business processes by building a bridge between its employees and the information that they need on a day to day basis. This book describes how composite applications can be used to optimize information in a Lotus® Notes® environment.

Always check the Composite Application wiki for the most current version of this publication:

http://www-10.lotus.com/ldd/compappwiki.nsf

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this material, use either of the following options:

- Use the online reader comment form, which is located at:

  http://www.ibm.com/software/data/rcf/
- Send your comments by E-mail to comments@us.ibm.com. Be sure to include the name of the book and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

Who should read this book

This guide is intended for Business Leaders (Line of Business Development), Technical Leaders (I/T Senior Management, CIO) and Business Partners (Lotus, WebSphere®, Eclipse). Business Leaders will get the most out of the sections "Introduction: Applying knowledge" on page 1 and "Business value of Notes 8.x application development " on page 3. Technical Leaders can then explore the value of Composite Applications illustrated by the remainder of the document and with the hands on samples hosted in the accompanying wiki.

How this book is organized

This book contains the following sections:

- Executive overview:
  - "Introduction: Applying knowledge" on page 1
  - "Business value of Notes 8.x application development " on page 3
    - "Exploring the power of Composite Applications" on page 5
    - "Where to use Composite Applications" on page 14
    - "Executive conclusion " on page 15
- Application development roadmap:
  - "Roles and tools" on page 17
  - "Architecture overview" on page 21
  - Component development:
    - "Getting started with Eclipse as a Notes developer" on page 27
    - "Building NSF components" on page 31
    - "Building Java components" on page 37
  - Application production:

    - "Assembling Composite Applications and creating widgets" on page 43
    - "Deploying applications" on page 51

Related information

This section lists documentation that you might find helpful:

- IBM Composite Applications wiki

Introduction: Applying knowledge

Your employees apply their knowledge every day in order to keep your business running. This involves the coordination of scarce resources, the prioritization of what is important, as well as leveraging the expertise of your organization.

Yet every day, people go through an information safari, hunting for what that they need to do their jobs – searching through multiple silos of applications, cutting and pasting information needed to complete the business process. This manual process wastes precious time.

As a CTO, CIO or business leader, you know that providing your people with the right information can accelerate your business processes. Linking the people who run your business to the information they need can make your people more productive (working quickly), more effective (working on the right things) and more efficient (aligned with business goals)

The purpose of this document is to illustrate how your business can apply knowledge to accelerate business processes by building a bridge between your employees and the information that they need on a day to day basis using Composite Applications in IBM® Lotus Notes.

Composite applications integrate multiple applications together on the desktop to share and dynamically change information in real time as the end user works the business process. By optimizing the information for the task at hand and freeing the end users from distraction, precious seconds can be shaved off of time sensitive business processes.

Composite applications can bridge both information technology and line of business systems so that end users have the right information, at the right time. This enables your team to respond to changes in the information as they happen, to speed up response time and to increase their ability to adapt to external pressures placed on the business.

Leveraging the power of Lotus Notes, composite applications can provide your team with the ability to coordinate calendaring, project management and other activities so that scarce resources are used efficiently.

This document illustrates the business value of accelerating information across your organization, regardless of which of the following is your source of information:

- Information technology
- Line of business
- End users
- External sources

Optimization of information can be as simple as publishing real time information to your employees' desktop systems, creating a context sensitive toolbar to eliminate unwanted distractions or to integrate a new productivity tool into the desktop.

The document also relies on the existing skills within your organization, so that your information technology and line of business developers can easily understand the concepts it presents. It relies on your existing applications and technologies, so that results can be felt in your business quickly. The document also shows the art of the possible or end game, but also contains samples that can be used to illustrate the concepts.

This document, the demonstrations and the accompanying wiki is meant to be used by your development team as a roadmap to optimizing information throughout your business.

Business value of Notes 8.x application development

This section introduces the business value of Composite Applications by describing the business and information environment as well as the challenges to be solved. Next, the benefits are applied to key roles. Composite applications provide benefits

- End users - by linking people to the process
- IT - by extending the process to the people as well as extending application development beyond traditional boundaries
- Business partners - by exposing an easily extended programming model

Lastly, this section's examples reinforce the value of Notes 8.x application development and describe several scenarios.

Fostering business and IT potential for flexibility, growth and integration

"What used to take our employees 6 minutes, now takes them 40 seconds." The ability to optimize information across multiple applications is key to delivering phenomenal gains in productivity and achieving the efficiency needed to focus on innovation.

Bridging Line of Business and Information Technology

Business and Information Technology (IT) are intricately entwined. A business needs to respond, grow and succeed with applications that leverage existing assets - including technology, skills and intellectual capital - to streamline business processes and support innovation. Time on task, the number of calls per hour, the number of customer visits per day and customer satisfaction are all important considerations for a business which strives to give its employees the tools they need to get the job done faster and with greater accuracy. The role of Information Technology is to build scalable systems that keep the business running in compliance to corporate policy, while maintaining governance and audit readiness. The focus is on systems integrity, completing new projects on time and reducing risk. As internal pressures for new systems functions and features, or external pressures such as competition or new partnerships, both Line of Business (LOB) and IT search for solutions to bridge one another and to provide the best potential for flexibility, growth and integration.

The challenge

For many IT departments, operational silos, heterogeneous environments and disjointed development platforms are the norm. These kinds of infrastructures keep development teams from effectively sharing and reusing assets, such as data models, and also make it difficult to leverage skills across multiple projects. Proprietary and closed applications limit IT's ability to extend the value of existing investments and infrastructure, and lock organizations into single vendor relationships. Moreover, when IT or application issues arise, or when new initiatives, such as service-oriented architecture (SOA), are launched, lack of support and resources from reliable and trusted vendors can hinder worker productivity.

Linking the people to the process (End User view)

IBM Lotus Software makes it easier to integrate line-of-business solutions and data into new types of applications, called Composite Applications. With Composite Applications, developers can create and aggregate components on the screen to present content from multiple systems in a unified interface for the end user. By integrating information across applications, your employees will notice they no longer need to switch between applications to get the job done. The screens automatically appear, as needed, and can be pre-populated with data. As the users change the data, they will notice that other applications that rely on that data automatically change. The desktop environment is no longer a place for static and stale information, but rather a dynamic work environment that has up to the minute information, presented in the context of what the user is doing. Silos of applications are integrated together in one desktop that can accelerate the business process and reduce the amount of time it takes to get tasks done. Existing applications can also be integrated into the Composite Application desktop. Reuse of applications can reduce the amount of retraining for your staff. New functions can be introduced beside old ones to minimize migration concerns.

Another key strength often overlooked is the ability to leverage existing native application code. Legacy Microsoft® Visual Basic, ActiveX, or Microsoft .NET applications that run on desktops today, integrate into the Notes 8 client to provide end users with a highly productive work environment. Information can be shared across applications and optimized for the user. The reuse of applications and the reduction in training can lead to cost savings.

For example, a Composite Application for a sales team might include access to relevant back end sales order entry, order tracking, service management and accounts receivable programs, as well as such front end productivity-enhancing and collaboration components as E-mail, calendars, people finders and document libraries.

Bottom line: A Composite Application provides collaboration in context, providing the information your people need, when they need it.

Extending the process to the people (Application Developer view)

Your application developers will discover that Composite Applications are built on an extensible client integration platform that provides the bridge between IT and LOB. This provides them with existing applications that run your business today, and the flexibility you need to innovate and grow your business in the future.

Existing applications are reused, avoiding the rip and replace of other proprietary systems. Applications that can be integrated into the environment include web, Portal, Host Access, Java™ EE, WebSphere, Eclipse, spreadsheets, documents, Lotus Notes and legacy Visual Basic, Microsoft .NET, ActiveX or SharePoint based applications. In addition, widgets that provide up to the minute information, such as RSS Feeds, can also be integrated into the desktop.

The platform utilizes the existing skills and tools that your developers have today, so that they can adapt the people to the process very quickly. Your Lotus Notes developers, using Domino Designer®, will be able to develop components which develop components which link the people to the process. IT developers can continue to build components and assemble Composite Applications to provide more flexible applications targeted to meet their end users requirements, extending the process to the people.

Additional benefits can be realized by developers designing and creating a catalog of reusable components which can be shared across the organization or company

Bottom Line: You save development time, reduce silos of information and enable your company to be more responsive to changing business needs and customer demands.

Involve business users in process innovation

With the Composite Application platform, IT can now expand the possible resources to empower advanced business users to assemble applications while still maintaining infrastructure integrity and security.

Lotus software provides tools that enable developers to build Composite Applications and the components that constitute them. The Composite Application Editor allows developers and authorized business users to assemble components into Composite Applications relevant to current business requirements. This separation has defined a new role of application assembler. Components can be used to create new or extend existing Composite Applications by additional authorized and trained resources in the application assembler role. This increases the pool of resources able to develop applications, as well as allowing the resources who best understand the business solution to adapt to business changes.

Business partners

So far, the business value has been illustrated using scenarios related to companies. However, business partners can also realize many of the benefits mentioned to this point for their own solution development as well as for providing benefits to their customers. All components need not be developed in house by a company's IT organization. IT organizations can augment their component catalogs by purchasing components from business partners who have specific domain experience. In addition to components, business partners can also develop Composite Applications that can be extended by customers to meet changing business requirements.

Exploring the power of Composite Applications

The Composite Application programming model is flexible, bridging the information to the people who need it, and allowing you to leverage any or all of the benefits. Over time, benefits can be realized and functionality adopted as you gain experience or have specific business requirements. Both the development organization, and the business as a whole, will respond more quickly to marketplace changes that require new ways of doing business. Lotus software can help you achieve just that with the ability to create Composite Applications.

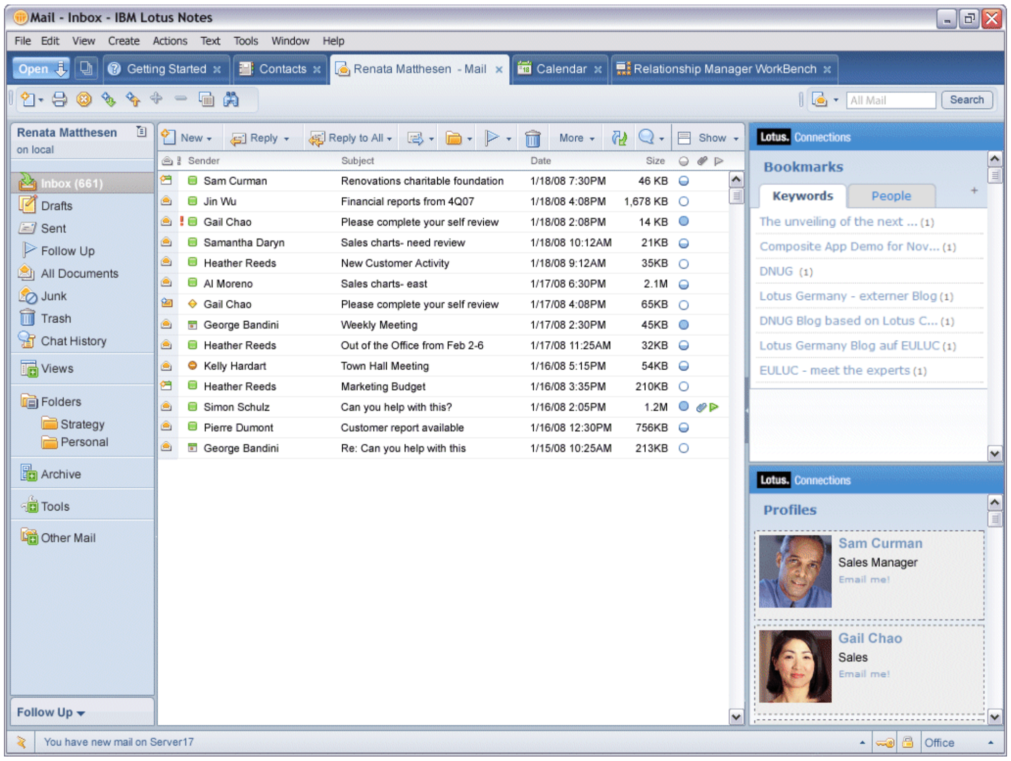
From a business perspective, the following examples reinforce the value of running different applications together in context, reusing new and existing assets, and Notes as an integration platform.

Improve End User effectiveness by running different applications together in context

The integration of information from multiple business components enables developers to provide application users with content specifically geared to their business roles and tasks in the appropriate context. This is accomplished with event and action relationships among the components to support communication across application and system boundaries.

User-facing Composite Applications also help improve effectiveness, reduce learning curves and create opportunities for innovation. When you can work within a UI that unifies content from multiple applications and systems, you can complete tasks more quickly and reduce errors. User initiated actions in one component can trigger the publication of information, causing other components to execute logic. For example, a Composite Application for a sales team might include access to relevant back end sales order entry, order tracking, service management and accounts receivable programs, as well as such front end productivity-enhancing and collaboration components as E-mail, calendars, people finders and document libraries. This application can be further improved by adding context. When a user selects an entry in an order entry component, the updated status can display in the accounts receivable component. This relationship is defined when the Composite Application is assembled and reduces the need for the user to rely on "copy and paste" or search to manually transfer the context between components. You can reduce the chances of introducing errors and impacting user efficiency, by removing the additional steps to manually transfer context. The following explores a few samples to better understand how context is used in Composite Applications.

Figure 1 has extended a user's E-mail, which in Notes 8 is a Composite Application, to include two new components. These components display contextual information from Lotus Connections for Bookmarks and Profiles. As the user navigates through his or her E-mail, the Profiles and Bookmarks component updates to show information for people listed in the **To**, **From** and **Cc** fields or **Subject**. The user does not have to copy a name or retype information into the components; the information is automatically transferred as defined by the application assembler. This is different then the Activities Sidebar plug-in, which displays global information for the logged in user instead of leveraging any context from the active Composite Application.



Sender	Subject	Date	Size
Sam Curman	Renovations charitable foundation	1/18/08 7:30PM	46 KB
Jin Wu	Financial reports from 4Q07	1/18/08 4:08PM	1,678 KB
Gail Chao	Please complete your self review	1/18/08 2:08PM	14 KB
Samantha Daryn	Sales charts- need review	1/18/08 10:12AM	21KB
Heather Reeds	New Customer Activity	1/18/08 9:12AM	35KB
Al Moreno	Sales charts- east	1/17/08 6:30PM	2.1M
Gail Chao	Please complete your self review	1/17/08 4:08PM	65KB
George Bandini	Weekly Meeting	1/17/08 2:30PM	45KB
Heather Reeds	Out of the Office from Feb 2-6	1/17/08 11:25AM	32KB
Kelly Hardart	Town Hall Meeting	1/16/08 5:15PM	54KB
Heather Reeds	Marketing Budget	1/16/08 3:35PM	210KB
Simon Schulz	Can you help with this?	1/16/08 2:05PM	1.2M
Pierre Dumont	Customer report available	1/16/08 12:30PM	756KB
George Bandini	Re: Can you help with this	1/15/08 10:25AM	213KB

Lotus. Connections

Bookmarks

Keywords | People

- The unveiling of the next ... (1)
- Composite App Demo for Nov... (1)
- DNUG (1)
- Lotus Germany - externer Blog (1)
- DNUG Blog based on Lotus C... (1)
- Lotus Germany Blog auf EULUC (1)
- EULUC - meet the experts (1)

Lotus. Connections

Profiles

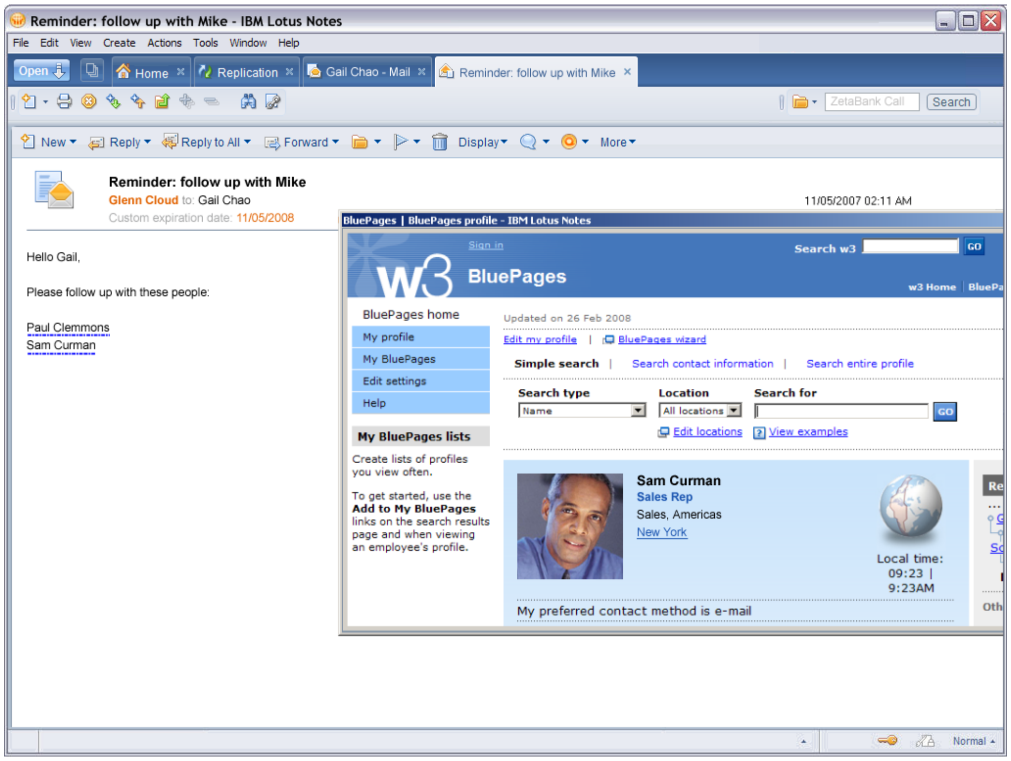
Sam Curman
Sales Manager
Email me!

Gail Chao
Sales
Email me!

Figure 1. Sharing information between Lotus Notes 8 and Lotus Connections

Lotus Notes 8.0.1 enables end users to see and act on Live Text within their Notes documents, including mail. For example, in Figure 2, a Notes user can see a person's name specially highlighted and instantly act on it to look up information about the person in the corporate directory. Without this function, the user would need to copy the person's name, find and launch the corporate directory Web page and then paste or retype the name. Productivity is improved by making common operations easier to perform. Developers or power users can create widgets that drive this type of function based on Notes views, forms, documents or even external sources such as a Web page, feed or Google Gadget and deploy them from a central catalog to specific users. Superficial information can be filtered from a Web page, allowing the user to focus on only the requested information. Live Text can be used to obtain instant information for phone numbers, addresses, part numbers, customer IDs, flight numbers and many other options.

Power users can create widgets based on a Notes view, form, or document (or a Web page, feed, or Google Gadget) and deploy them from a central catalog to specific users. Superficial information can be filtered from a Web page widget, allowing the user to focus on only the requested information. Live Text recognizers can include any pattern definable by a regular expression. Examples include phone numbers, addresses, part numbers, customer IDs, flight numbers and many more.


Reminder: follow up with Mike - IBM Lotus Notes
File Edit View Create Actions Tools Window Help
Open Home Replication Gail Chao - Mail Reminder: follow up with Mike
ZetaBank Call Search
New Reply Reply to All Forward Display More
BluePages | BluePages profile - IBM Lotus Notes
Sign in
Search w3 GO
w3 BluePages
w3 Home
BluePages home
My profile
My BluePages
Edit settings
Help
My BluePages lists
Edit my profile | BluePages wizard
Simple search | Search contact information | Search entire profile
Search type Location Search for
Name All locations GO
Edit locations View examples
Normal

Reminder: follow up with Mike
Glenn Cloud to: Gail Chao
Custom expiration date: 11/05/2008
11/05/2007 02:11 AM

Hello Gail,

Please follow up with these people:

Paul Clemmons
Sam Curman

Updated on 26 Feb 2008

Create lists of profiles you view often.

To get started, use the **Add to My BluePages** links on the search results page and when viewing an employee's profile.

Sam Curman
Sales Rep
Sales, Americas
New York

Local time: 09:23 | 9:23AM

My preferred contact method is e-mail

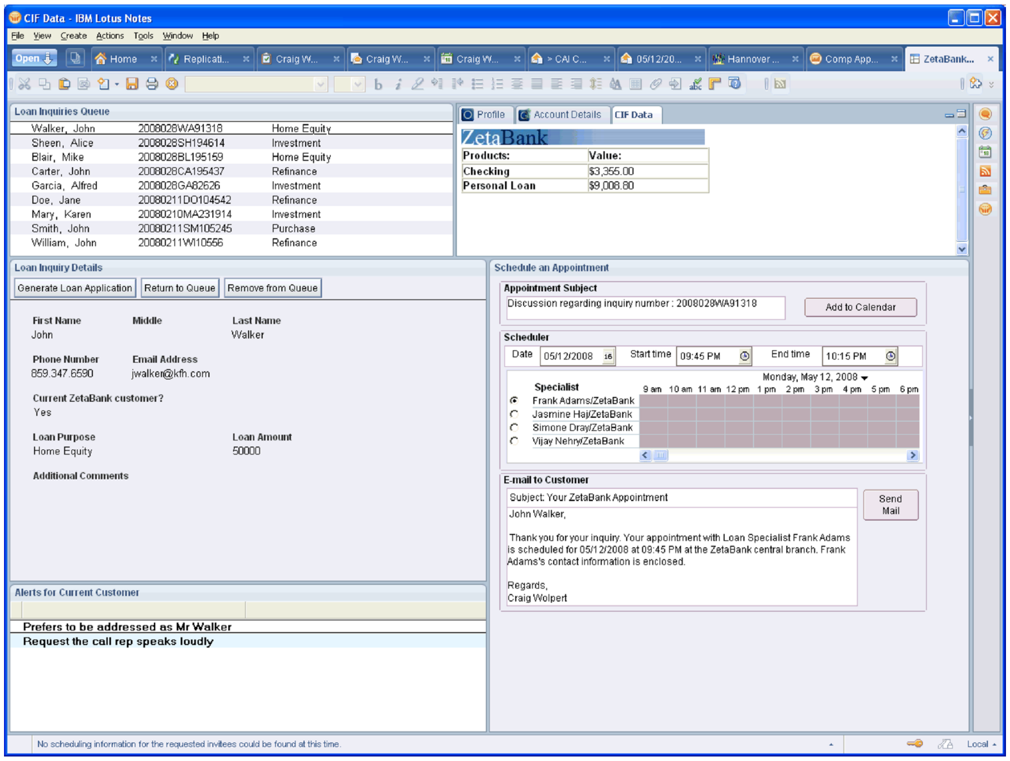
Figure 2. Avoiding Copy and Paste with the LiveText and Directory Widget

The value of reusing new and existing assets

In today's world, everyone is busy meeting their business requirements. Companies are distributed over a variety of locations and time zones, making it difficult to share assets. In many cases, developers create new assets and duplicate features, as code developed for one solution is often difficult to reuse in another. The Composite Application Framework provides a programming model which breaks down many of these barriers.

A Composite Application is comprised of constructed components which can be developed using different tools and frameworks. This enables a business to establish a set of common components that can be used across multiple applications, resulting in higher quality components, less maintenance and greater software asset reuse within an organization. To avoid duplication and achieve greater reuse, components can also be shared across the development team using a catalog. One example of a product catalog is the Widget Catalog (8) provided with Notes 8. Both developers and application assemblers (power users) can create and publish widgets to the catalog for use throughout the organization. It can also be used to augment a company's component catalogue with Third Party Widgets. This enables build versus buy, and component sharing on an industry-wide basis. The Composite Application Component Library on OpenNTF.Org is an example of a catalog that uses conventions, and can be adopted by a company or organization. This reduces the resources needed to build components, which in turn increases the shared assets used in Composite Applications.

So far, the discussion has focused on component reuse, but the real advantage can be seen when you assemble Composite Applications from components. The Lotus Notes 8 Banking Industry Composite Application Demo is a perfect example, as shown in Figures 3 and 4. This demo reuses CIF Data, Alerts, and Account Details components and the sharing of information between components along with Personal Information Management (PIM), such as Contacts (Personal Address Book), Mail, and Calendar.



Loan Inquiries Queue

Walker, John	2008028WA91318	Home Equity
Sheen, Alice	2008028SH194614	Investment
Blair, Mike	2008028BL195159	Home Equity
Carter, John	2008028CA195437	Refinance
Garcia, Alfred	2008028GA82626	Investment
Doe, Jane	20080211DO104542	Refinance
Mary, Karen	20080210MA231914	Investment
Smith, John	20080211SM105245	Purchase
William, John	20080211WI10556	Refinance

Products:	Value:
Checking	$3,355.00
Personal Loan	$9,008.80

Loan Inquiry Details

First Name: John
Middle:
Last Name: Walker

Phone Number: 859.347.6590
Email Address: jwalker@kfh.com

Current ZetaBank customer?: Yes

Loan Purpose: Home Equity
Loan Amount: 50000

Additional Comments

Alerts for Current Customer

Prefers to be addressed as Mr Walker
Request the call rep speaks loudly

Schedule an Appointment

Appointment Subject: Discussion regarding inquiry number : 2008028WA91318

Scheduler

Date: 05/12/2008 Start time: 09:45 PM End time: 10:15 PM

Monday, May 12, 2008

Specialist	9 am	10 am	11 am	12 pm	1 pm	2 pm	3 pm	4 pm	5 pm	6 pm
● Frank Adams/ZetaBank										
○ Jasmine Haj/ZetaBank										
○ Simone Dray/ZetaBank										
○ Vijay Nehry/ZetaBank										

E-mail to Customer

Subject: Your ZetaBank Appointment

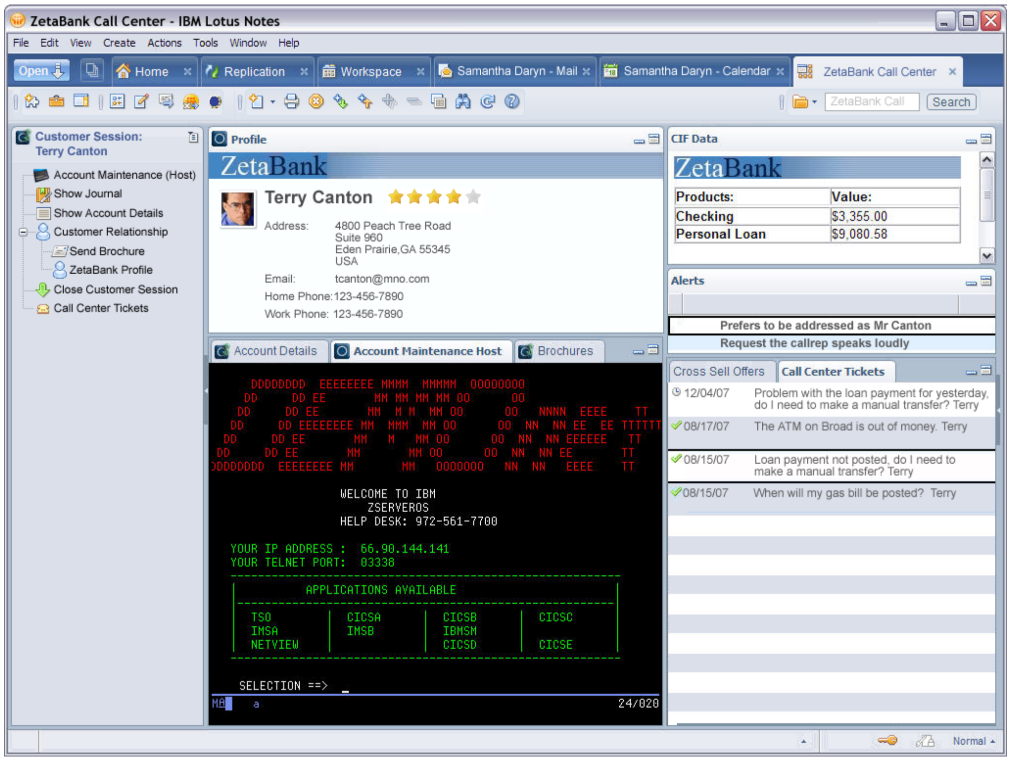
John Walker,

Thank you for your inquiry. Your appointment with Loan Specialist Frank Adams is scheduled for 05/12/2008 at 09:45 PM at the ZetaBank central branch. Frank Adams's contact information is enclosed.

Regards,
Craig Wolpert

No scheduling information for the requested invitees could be found at this time.

Figure 3. Customer Service Bank Call Center Composite Application



Terry Canton

Address: 4800 Peach Tree Road
Suite 960
Eden Prairie,GA 55345
USA

Email: tcanton@mno.com
Home Phone:123-456-7890
Work Phone: 123-456-7890

Products:	Value:
Checking	$3,355.00
Personal Loan	$9,080.58

Prefers to be addressed as Mr Canton
Request the callrep speaks loudly

12/04/07	Problem with the loan payment for yesterday, do I need to make a manual transfer? Terry
08/17/07	The ATM on Broad is out of money. Terry
08/15/07	Loan payment not posted, do I need to make a manual transfer? Terry
08/15/07	When will my gas bill be posted? Terry

```
     DDDDDDDD  EEEEEEEE MMMM  MMMMM  OOOOOOOO
    DD     DD EE       MM MM MM MM OO      OO
   DD     DD EE       MM  M M  MM OO      OO   NNNN  EEEE    TT
  DD     DD EEEEEEEE MM  MMM  MM OO      OO  NN  NN EE  EE TTTTTT
 DD     DD EE       MM   M   MM OO      OO  NN  NN EEEEEE   TT
DD     DD EE       MM       MM OO      OO  NN  NN EE       TT
DDDDDDDD  EEEEEEEE MM       MM  OOOOOOO   NN  NN   EEEE    TT

                   WELCOME TO IBM
                      ZSERVEROS
                   HELP DESK: 972-561-7700

  YOUR IP ADDRESS :  66.90.144.141
  YOUR TELNET PORT:  03338
  ---------------------------------------------------------
  |          APPLICATIONS AVAILABLE                       |
  |-------------------------------------------------------|
  |  TSO          |  CICSA        |  CICSB        |  CICSC        |
  |  IMSA         |  IMSB         |  IBMSM        |               |
  |  NETVIEW      |               |  CICSD        |  CICSE        |
  ---------------------------------------------------------

    SELECTION ==> _
MA    a                                                  24/020
```

Figure 4. Loan Composite Application

The last benefit of reuse does not lie between components in Composite Applications, but rather sharing functionality from an existing application.

The Lead Manager Composite Application sample for IBM Lotus Notes 8 was developed from four existing Lotus Notes applications (Figure 5). All component creation can be accomplished without breaking existing functionality. This allows part of your user community to continue using the existing applications as others are migrated to the Composite Application (Figure 6). This reuse of applications can reduce the amount of training for end users, since they are already familiar with the function of an application. Because applications share information, the additional integration saves them time navigating through multiple screens and applications.

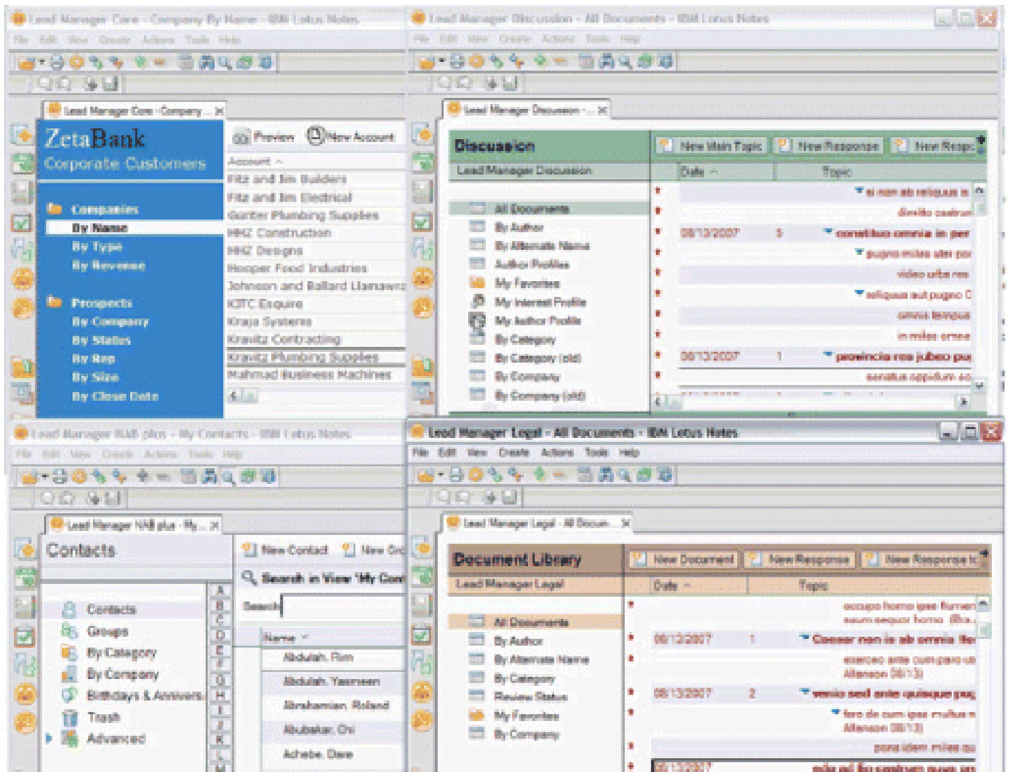
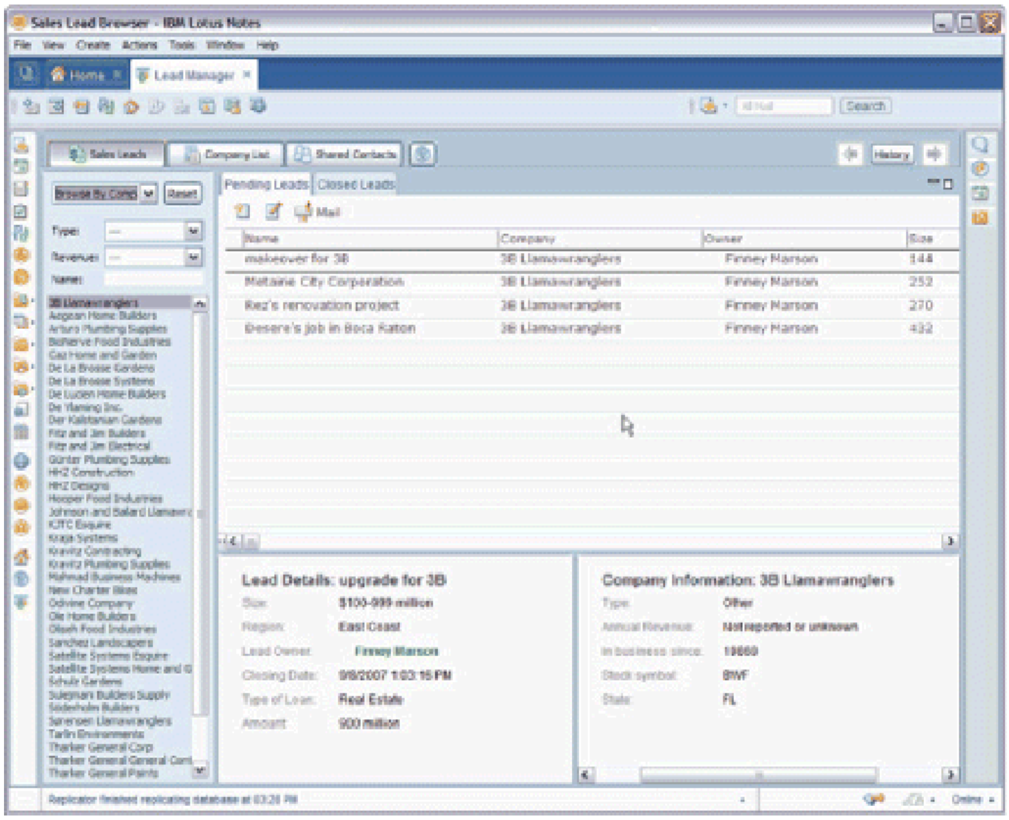


Figure 5. Existing Lotus Notes applications used to track Sales Lead



3B Llamawranglers
Aegean Home Builders
Arturo Plumbing Supplies
Bohserve Food Industries
Caz Home and Garden
De La Brosse Gardens
De La Brosse Systems
De Lucien Home Builders
De Yiaming Inc.
Der Kalstanian Gardens
Fitz and Jim Builders
Fitz and Jim Electrical
Günter Plumbing Supplies
HHZ Construction
HHZ Designs
Hooper Food Industries
Johnson and Balard Llamawr
KJTC Esquire
Kraja Systems
Kravitz Contracting
Kravitz Plumbing Supplies
Mahmud Business Machines
New Charter Bikes
Odivine Company
Ole Home Builders
Olseh Food Industries
Sanchez Landscapers
Satellite Systems Esquire
Satellite Systems Home and G
Schulz Gardens
Suleman Builders Supply
Söderholm Builders
Sørensen Llamawranglers
Tarlin Environments
Tharker General Corp
Tharker General General Cont
Tharker General Paints

Name	Company	Owner	Size
makeover for 3B	3B Llamawranglers	Finney Marson	144
Metairie City Corporation	3B Llamawranglers	Finney Marson	252
Kez's renovation project	3B Llamawranglers	Finney Marson	270
Desere's job in Boca Raton	3B Llamawranglers	Finney Marson	432

Lead Details: upgrade for 3B

Size:	$100-999 million
Region:	East Coast
Lead Owner:	Finney Marson
Closing Date:	9/8/2007 1:03:16 PM
Type of Loan:	Real Estate
Amount:	900 million

Company Information: 3B Llamawranglers

Type:	Other
Annual Revenue:	Not reported or unknown
In business since:	19860
Stock symbol:	BWF
State:	FL

Replicator finished replicating database at 03:28 PM

Figure 6. Lead Manager Composite Application sample

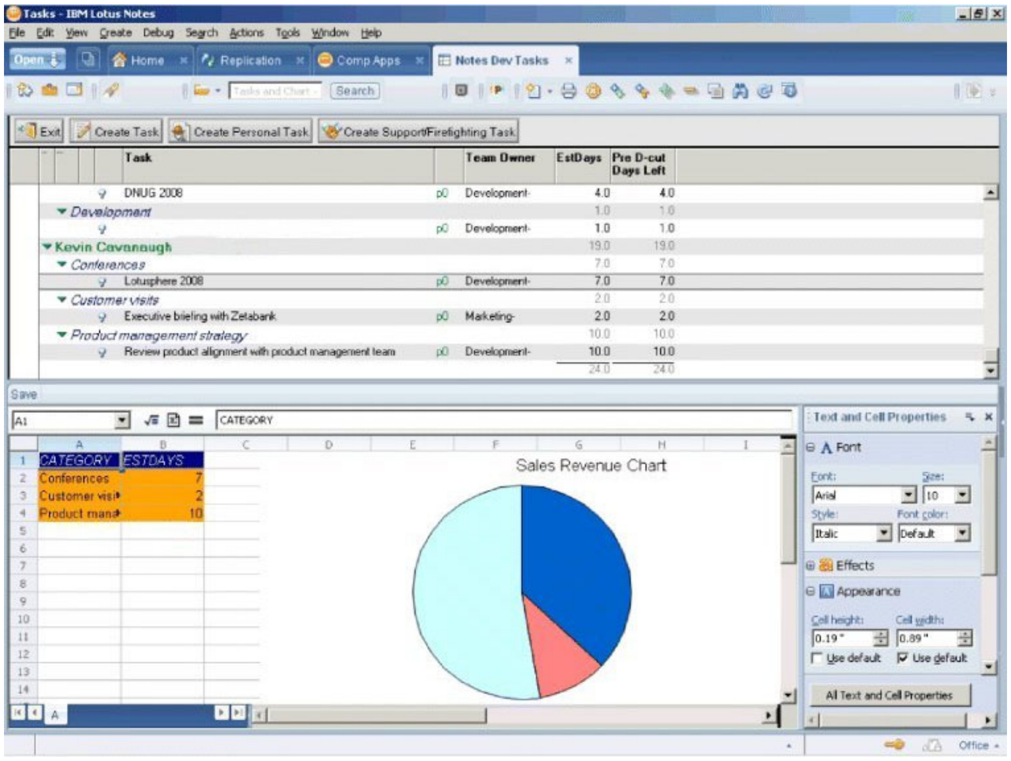
This Lead Manager Composite Application sample serves as a transition to the next section, because Composite Applications improve integration. For more information on this sample, refer to the IBM Composite Applications wiki page http://www-10.lotus.com/ldd/compappwiki.nsf/dx/ibm-lead-manager. Additional articles and information are available at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/lead-manager-sample-composite-application-articles.

The value of Lotus Notes as an integration platform

Lotus Notes 8 software provides support for heterogeneous technologies in Composite Applications. Developers can mix and match Lotus Notes applications with a wide variety of both client (Web, Web 2.0, Eclipse and legacy ActiveX, Sharepoint and .NET) and server (Portlets, Host Access, Transaction Processing...) applications. So far, the examples and values discussed have focused on context and reusability, but they also provide examples of Lotus Notes as an integration platform. Multiple applications were integrated on the glass to create a new application to help employees complete their tasks faster and with less errors. The following examples provide more detail on these benefits.

The Customer Service Bank Call Center and Loan Composite Application (Figures 3 and 4) illustrates an integration with many different types of components. Both applications include Host Access Transformation Services (HATS), Eclipse Standard Widget Toolkit (SWT), Lotus Symphony, JSR 168 portlets and Lotus Notes Views.

IBM Lotus Symphony, included in Notes 8, provides functionality for the end user to operate on spreadsheets, presentations and documents. It can also be used as a component in a Composite Application. Figure 7 illustrates how a chart can be created by integrating a spreadsheet with a Lotus Notes View. In this figure, specific view columns populate a spreadsheet, which then creates the chart. By leveraging the Lotus Symphony Public APIs to build components, many integration options are revealed. For example, a document template component which uses a property from another component to populate template values.



Task		Team Owner	EstDays	Pre D-cut Days Left
DNUG 2008	p0	Development-	4.0	4.0
Development			1.0	1.0
	p0	Development-	1.0	1.0
Kevin Cavanaugh			19.0	19.0
Conferences			7.0	7.0
Lotusphere 2008	p0	Development-	7.0	7.0
Customer visits			2.0	2.0
Executive briefing with Zetabank	p0	Marketing-	2.0	2.0
Product management strategy			10.0	10.0
Review product alignment with product management team	p0	Development-	10.0	10.0
			24.0	24.0

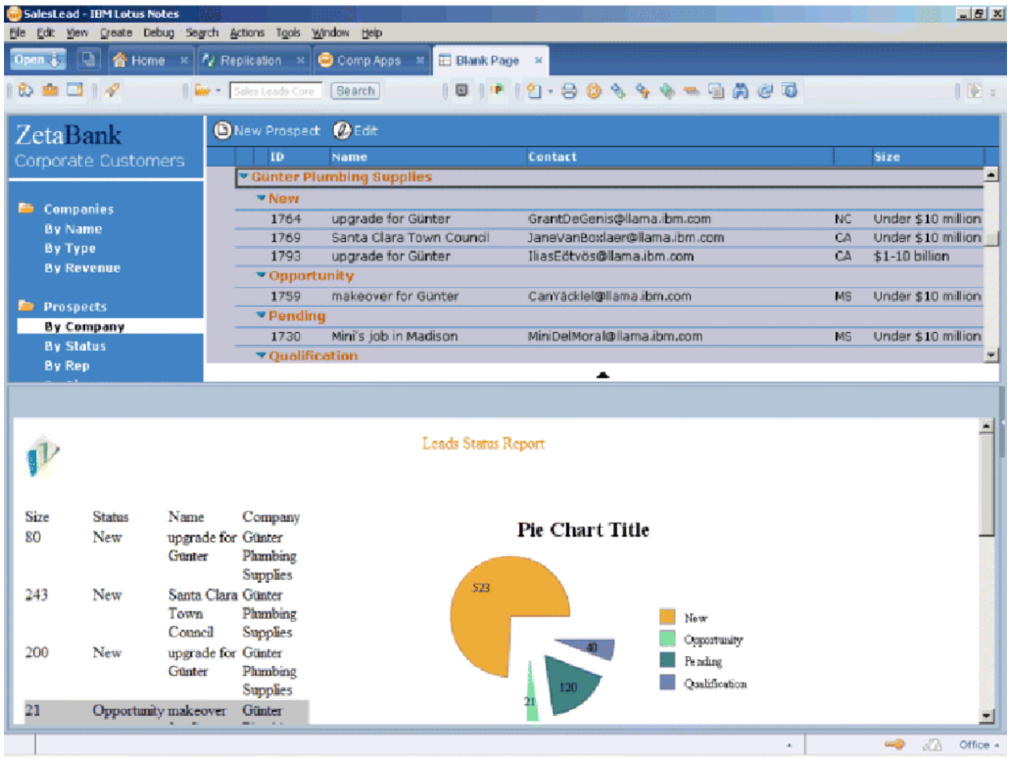
A1 CATEGORY

	A	B
1	CATEGORY	ESTDAYS
2	Conferences	7
3	Customer visi	2
4	Product mana	10

Font: Arial Size: 10
Style: Italic Font color: Default
Cell height: 0.19" Cell width: 0.89"
Use default [✓ Use default]

Figure 7. IBM Lotus Symphony in a Composite Application

Open source Eclipse projects represent another community that can be leveraged to build components. The Business Intelligence and Reporting Tools (BIRT) Eclipse-based reporting system is one such example, shown in Figure 8. A similar technique for integrating BIRT was also used with Lotus Symphony, detailed in Figure 7. Instead of populating cells in a spreadsheet, the component uses the Lotus Notes Java API to extract the information for a view. The necessary view and category information is specified as a component action.



ID	Name	Contact		Size
Günter Plumbing Supplies				
New				
1764	upgrade for Günter	GrantDeGenis@llama.ibm.com	NC	Under $10 million
1769	Santa Clara Town Council	JaneVanBoxlaer@llama.ibm.com	CA	Under $10 million
1793	upgrade for Günter	IliasEdtvös@llama.ibm.com	CA	$1-10 billion
Opportunity				
1759	makeover for Günter	CanYäcklel@llama.ibm.com	MS	Under $10 million
Pending				
1730	Mini's job in Madison	MiniDelMoral@llama.ibm.com	MS	Under $10 million
Qualification				

Size	Status	Name	Company
80	New	upgrade for Günter	Günter Plumbing Supplies
243	New	Santa Clara Town Council	Günter Plumbing Supplies
200	New	upgrade for Günter	Günter Plumbing Supplies
21	Opportunity	makeover	Günter

Figure 8. BIRT – Third Party Reporting Tool integrated into Notes 8

The integration described previously allows end users to leverage the information they need. But how are these solutions created?

Without Composite Applications, these solutions would be created by IT and application developers, a critical resource in many companies. With Composite Applications, the pool of available resources increases beyond IT to also include authorized LOB users. Application assembly is a new role which uses the Composite Application Editor to layout and wire components. Besides creating applications, resources can also be saved through incremental updates and modification to existing applications by LOB users. As business environments and processes change, an authorized LOB user can add reusable components from a catalog to enhance applications without having to burden IT. This shortens the development cycle, as a LOB user who understands the new requirements is able to apply and verify the changes.

In addition to Composite Applications, the new architecture of Lotus Notes enables a wide range of options to extend and customize the user experience. For example, using Eclipse plug-ins to extend the sidebar and toolbar. Lotus Notes uses this technique by providing constant access to your IBM Lotus Sametime® Contacts, Calendar, Feed Reader, and Activities on the sidebar.

Where to use Composite Applications

To realize the benefits of Composite Applications discussed so far, you must understand where to use and not to use Composite Applications. The following rules can serve as a guide:

1. They must be used for on the glass integration.
2. They can be used to integrate different technologies in a platform independent approach.

3. They are not a replacement of NSF database design, but rather an evolution.
4. They do not require Eclipse skills.
5. They do not require an IBM WebSphere Portal server for Lotus Notes.

The first and most important rule to understand is that Composite Applications provide on the glass integration of coarse grained components. Until a reasonable limit is reached, the platform becomes more valuable as more components are integrated. All the components can come from a common technology, as described in Figure 6, the *Lead Manager Composite Application sample*. Or many separate technologies can also be used, as described in Figure 3, *Customer Service Bank Call Center Composite Application*.

The second rule is an extension of the first. When providing on the glass integration, a platform independent technique integrates different technologies. This is accomplished by creating components which access the technology. You can add these components and/or plug-ins to the sidebar or assemble them into Composite Applications using the Composite Application Editor.

The remaining three rules attempt to resolve possible confusion. Developing NSF components is not a replacement for NSF database design, but an evolution to create reusable components. Your existing skills and experience with Lotus Domino Designer are relevant and can be leveraged. Eclipse skills are not required and components can be created using Lotus Domino Designer. Eclipse skills provide additional integration options, and as they are learned they can be applied. IBM WebSphere Portal server is not required. Components and Composite Applications can be developed and deployed in a Lotus Notes and Domino® only configuration. If an IBM WebSphere Portal server configuration already exists, then it can be leveraged.

Executive conclusion

IBM Lotus Software makes it easier to integrate line-of-business solutions and data into Composite Applications. With Composite Applications, developers can create and aggregate components on the screen to present content from multiple systems in a unified interface for the end user. With this platform, end users can also assemble Composite Applications to create personal productivity tools. Because composite applications are based on a Notes and Domino infrastructure, integrity and security are maintained throughout the process. Finally, reusable components, shared between developers on an industry level, can help a business get the most value out of code reuse. Code that is tested, proven in production, and written by someone else.

This concludes the Executive Overview of Composite Applications within a Notes 8 environment. The remainder of this document allows your developers to begin exploring in more detail the concepts and value described above.

Roles and tools

The roadmap to information optimization

Application development is best shown in light of samples and examples. Which technologies, which tools and which APIs will help you to respond to the needs of the business? This paper is designed to act as demonstration vehicle, showing you how to optimize information to end users. When used in conjunction with the IBM Composite Applications wiki, it provides a roadmap to assist you in applying the concepts illustrated.

First, there are two types of development: fast and furious in response to a one-off requirement or need for information, and the integration of multiple applications together on the desktop for mass deployment. Mashups versus Composite Applications.

While both mashups and Composite Applications provide a framework for integrating disparate applications into a common user experience, one can think of mashups as a lightweight implementation of Composite Applications. Mashups and Composite Applications use widgets and components respectively as the building blocks for the application.

Composite Applications

Composite applications involve a “service-oriented” wiring between applications and are deployed to enterprise platforms. This event-based wiring of data between applications can create a dynamic desktop where, as the workflow inside one application is completed, all other applications that rely on this application can reflect the change. The completed application is then placed in the background and the user focuses on the next task.

This enables not only component reuse within an organization, but also provides a framework for application reuse across the business. In this way, both line of business (LOB) and Information Technology (IT) applications can be integrated together on the desktop.

Mashups

A ″mashup″ combines data from multiple applications into an extremely lightweight application rendered in a browser. Mashups can extend enterprise data with services from the public internet such as Google Maps for location services, RSS (Really Simple Syndication) Feeds to publish content or Widgets (web plug-ins) that provide specific functions such as blog, wiki or social networking capabilities.

Mashups are not intended to be strategic, systematically built, industrial-strength enterprise applications; rather, they are created quickly or opportunistically to meet a focused tactical need. Mashups are generally personalized to fulfill personal productivity needs rather than the requirements of a long-standing corporate role. [1]

Mashups leverage content and logic from other Web sites and Web applications, they’ are lightweight in implementation and are built with a minimal amount of

1. G. Phifer, D. Gootzit, 2007, Gartner, Inc., Hype Cycle for Client applications

code (which can be client-side JavaScript™ or server-side scripting languages, such as PHP or Python). These are not fixed requirements, but reflect the original implementation of the mashup concept in Web 2.0 startup companies, which typically do not use enterprise oriented platforms, such as Java or .NET.

Your skills and tools

Your skills and tools can be leveraged to create Composite Applications. The framework minimizes the amount of rework necessary to integrate applications into the composite desktop.

Typically, there are three developer types who work on component and application creation:

- Notes developers
- Java developers
- Eclipse developers
-

Notes developers

By defining the UI through a set of design elements, adding controls to those elements and defining the behavior of those controls, developers can build a fully functional application without having to write any code. This has made Notes developers extremely productive, as well as making the Notes programming model widely accepted.

The Notes application model's flexibility has allowed it to support a variety of application types. This includes project management, document tracking, CRM and sales force automation. The nature of Notes' semi structured data store and its associated APIs allow developers to quickly build workflow functionality into their applications.

The Notes application model also benefits from platform services, such as security, allowing fine grained control over data access or replication allowing workers to with the application when disconnected.

For applications that require more complex business logic, or for access to the data and business logic in a Notes application, a wide variety of application programming interfaces are provided, including LotusScript, COM, JavaScript, Java, Web services, RSS, XML and ODBC. Notes applications can also be hosted on a Domino server for browser access, leveraging developers skills in HTML, JavaScript, CSS and XML.

For Notes developers, Domino Designer continues to provide an integrated development environment (IDE) for building NSF components. You can directly apply many of the skills you use today to develop Notes applications towards developing components.

For Notes developers, the IBM Composite Applications wiki contains updated exercises and a roadmap so that you can begin to place the concepts into production.

Java and Eclipse developers

There are over two million Eclipse developers worldwide. The Eclipse Rich Client Platform and Equinox form the core of the Lotus client products. For all developers, it simply enables applications to integrate together to form a dynamic work environment.

For Java and Eclipse developers, it provides a framework for application integration within Notes, Symphony, Sametime and Expeditor environments with application and component reuse across the entire organization.

It enables developers to use their favorite Integrated Development Environment (Eclipse, Rational® Application Developer, Instantiations Window Builder Pro, etc.).

As Java developers with knowledge of the Eclipse plug-in model, you will be able to:

- Integrate existing applications to reduce end user retraining
- Integrate both line of business and information technology applications to bridge end users to the information they need for their job
- Easily integrate third party applications so that functions, such as reports, can be integrated into applications
- Reparent applications that utilize other technologies, such as, Java, host access, Eclipse, native and legacy .NET, VB, Office, to further enhance end user productivity
- Wire portal applications to accelerate information to customer care employees in your call centre or point of sale, point of service

For Java developers, the IBM Composite Applications wiki has updated exercises and a roadmap, so that you can begin placing the concepts into production.

Application Assembly/Application Deployment

The value in Composite Applications is the ability to wire applications together to optimize information to the end user. To create this dynamic end user environment, both Notes developers and Java developers will use the Composite Application Editor and the underlying Property Broker provide the event-based between applications.

As previously mentioned, there is an updated set of applications and roadmap on the IBM Composite Applications wiki.

Architecture overview

At the core of the Composite Applications architecture is the Eclipse Rich Client Platform (RCP). RCP is an application integration framework that allows developers to quickly build a professional-looking application with a native look-and-feel on multiple platforms, allowing them to focus on their value-add. The inherent extensibility of Eclipse allows them to not only build products, but also an open-ended platform.

Extensibility is accomplished by delivering Plug-ins or components to the Eclipse RCP. Each component includes a manifest file that declares the extension points it provides to other components and the extension points it makes use of from other components. This defines how the component can be used.

The multiple components inside a composite application can consist of many applications and controls - from Java Foundation Class Swing, Microsoft Visual Basic and ActiveX and applets to native or Web views like Asynchronous JavaScript and XML (Ajax), Adobe PDF and Java Server Pages.

When the component is installed, the extensions are registered with the registry. Events are sent to those interested that a new extension has been installed.

Lotus Expeditor is a commercial implementation of the Eclipse Rich Client platform and adds middleware, security, provisioning and the ability to run Portlets, integrate host access and use IBM Lotus Forms as components.

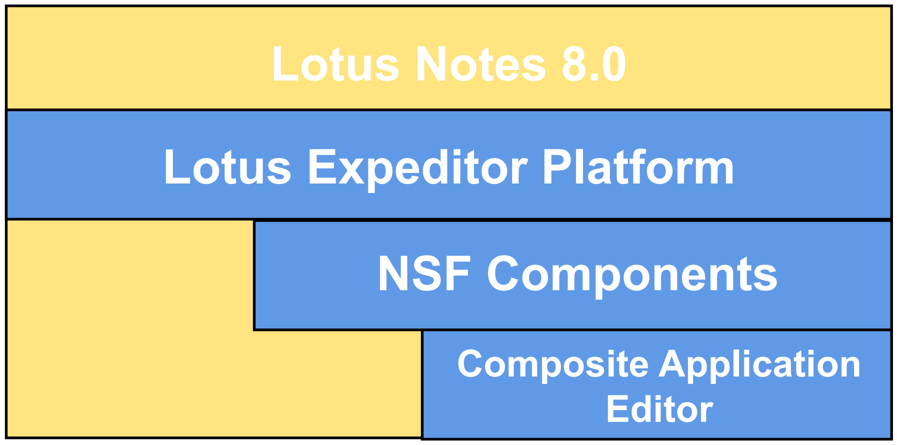


Figure 9. Lotus Expeditor platform and Lotus Notes 8.0

In addition to the components supported through Lotus Expeditor, Lotus Notes 8 enables the following design elements to be ″surfaced′ as NSF components in composite applications:

- Pages
- Forms
- Views
- Folders
- Framesets

- Navigators

Component interaction is loosely coupled through properties and actions across database boundaries, enabling developers to mix-and-match component technologies across applications.

The component model also enables Widgets to be integrated on the desktop. Widgets in the sidebar panel in Lotus Notes allow users to quickly and easily perform business actions using widgets created from Lotus Notes views, Web pages, feeds and Google Gadgets. These widgets can be standalone or launched from Live Text in Lotus Notes documents.

Instead of implementing "hard-wired communication" components, Widgets can rely on the selection service. It decouples components where items can be selected from others reacting on selection changes.

A property is a typed exchangeable data item that a component produces while an action is the logic to consume a property. Components define their properties and actions. Wiring connects properties to actions with a causal relationship.

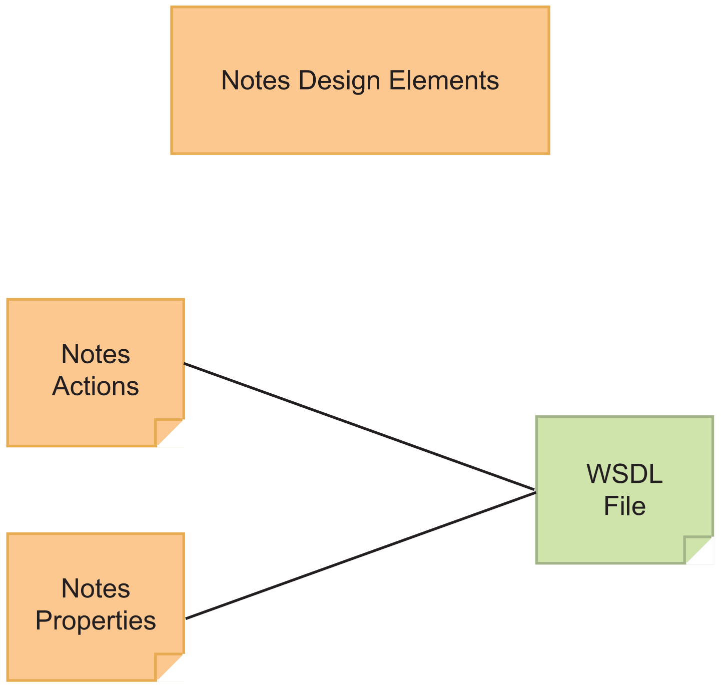


Figure 10. Notes Design Elements

The composite programming model present on the client is based on declarative communication between two applications or components. The Property Broker is a framework that provides registration, wiring and communications to decoupled components. This delivers a messaging system where actions can be called with input property changes and any number of output property changes.

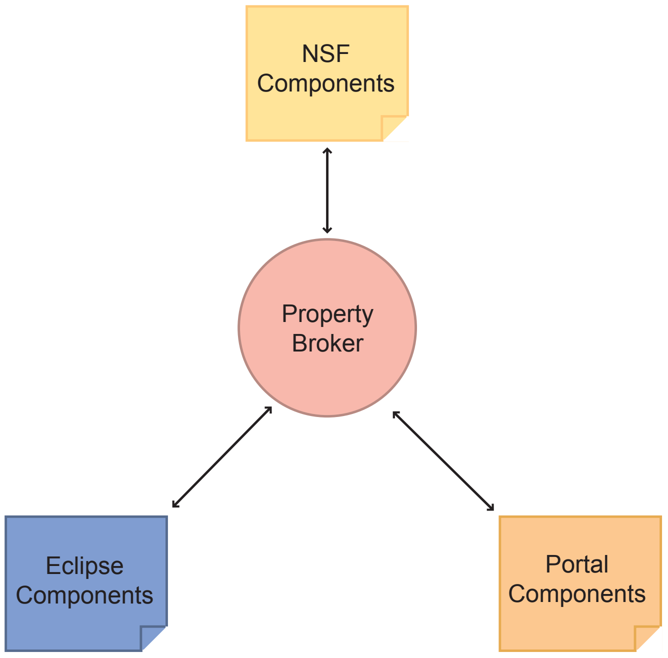


Figure 11. Property Broker framework

Other applications that have registered for changes are notified and can dynamically react. For example, the data in a spreadsheet application can be wired to a host access session on the desktop. As a cell within the spreadsheet changes, the Host Access screen automatically reflects the change in data. The framework can also be used to launch or close views on the desktop to respond to input from either the user or the system. Property changes can also be chained together to drive a dynamic desktop environment where the screens change according to the input from the user. This is very similar to the inter Portlet communication wiring mode.

The Composite Application data model

There are three levels of data:

1. Application
2. Pages
3. Components

A page in the client platform is equivalent to an Eclipse perspective. The page ID (which can be accessed from the `Page` class) is actually the Eclipse perspective ID. You can use the Eclipse APIs to show a page using that same ID.

Most components, not all, are equivalent to Eclipse views. The secondary ID of the view is used as the unique ID for the component. You can get a handle to the preferences of a component by using the secondary ID of a view part.

Java developers can access the model using the Topology Handler or Eclipse extension registry APIs in Lotus Expeditor.

More information is available on the Composite Application Wiki under these two areas:

- Understanding the data model

- Accessing component data using Eclipse API's

Composite Applications on the rich client platform

Lotus Expeditor is IBM's client side services oriented architecture platform. It is a universal client platform spanning Windows®, Macintosh, WinMobile, Linux® and more. The Composite Application infrastructure allows for the interoperation of new components based on virtually any kind of technology. This means you can use existing data and applications within this framework with little or no coding changes.

Although the Portal windowing environment is different, a mapping from Portal Pages to Eclipse perspectives and portlets enables the use of Portlets (JSR 186) within composite applications. Figure 9 illustrates this relationship map.

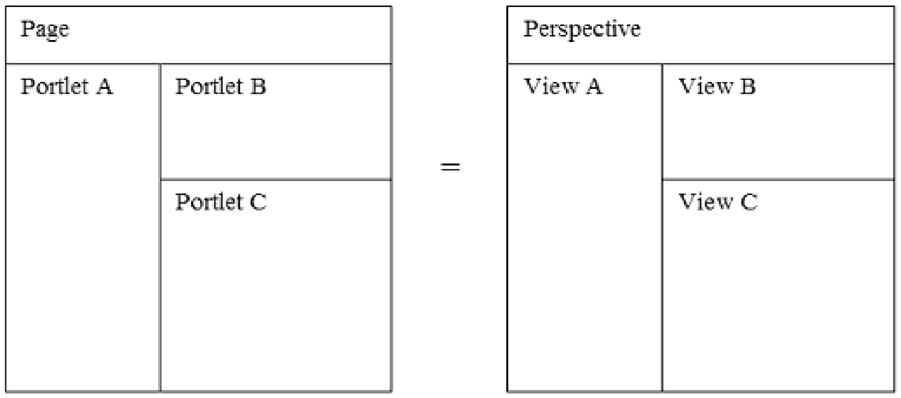


Figure 12. Relationship map

Table 1 shows the different products that use Composite Applications and the containers they support. **Lotus Notes inherits all of the component support from Expeditor*.

Table 1. Composite Application products, components and samples

Lotus Notes	
NSF based	Existing notes components
Symphony	Document, Spreadsheet, etc.
Widget support	Notes views, Web pages, feeds, and Google gadgets
Lotus Expeditor	
Embedded Web browser	HTML, Ajax, CSS, Dynamic HTML, Applets, Active X controls
Web	JSP 2.0, Servlet 2.4
Portlet	JSR 168
SWT RCP	Eclipse view support
AWT Swing	Standard Java
OLE NET	SWT based OLE container wrapper
Window Re-parenting	Legacy applications - VB, C++, etc.

In addition to basic components that can be laid out on the screen and wired declaratively with a tool like the Composite Application Editor, components can also be defined using the Lotus Widget wizard. Using this wizard, you can define four different kinds of widgets to be displayed in a pop up window, a new tab, or attached to the right sidebar. The Widgets are closely tied to the Live Text feature – where widgets can be launched from the Live Text menu.

Widgets and Live Text

The Eclipse platform has a very powerful selection and context menu architecture. This architecture lets developers create decoupled associations with well known selection types. The Lotus client platform extends this capability to include a declarative content type system that seamlessly integrates with Eclipse context selection menus. This capability gives users, assemblers, and developers the ability to associate common text with a widget and actions. For instance, Live Text allows you to associate an E-mail address with a specific web page. That web page can not only be launched as a new window tab, pop up window, or sidebar view, but also driven by the selected (underlined) text. The Live Text feature automatically recognizes text that fits a specific pattern and makes accessible, through the context menu, all available actions and widgets for that content type.

Getting started with Eclipse as a Notes developer

Eclipse can be thought of as a universal tool platform – an open extensible Integrated Development Environment (IDE) for anything and nothing in particular.[2] Eclipse is an open source project that started as a development platform for providing a flexible programming development environment, and then evolved into a flexible Rich Client Platform (RCP).[3] The Eclipse Newcomers FAQ will prove valuable if you are new to Eclipse.[4]

The same feature and plug-in architecture the IDE was built on is now the core framework for the Eclipse Rich Client Platform. The runtime platform is based on the Open Services Gateway Initiative (OSGi) and the implementation is called Equinox. The Eclipse project Equinox [5] is the implementation of the OSGi R4 specification.[6] OSGi technology provides a service-oriented, component-based environment for developers, and offers standardized ways to manage the software lifecycle.[7] At the very core of the rich client platform is the feature and plug-in architecture. This architecture allows each plug-in to have its own class loader, along with defining its dependencies on other plug-ins.

Eclipse Plug-ins, Features, and Fragments

Eclipse is not a single, monolithic program, but rather a small kernel called a plug-in loader surrounded by hundreds (and potentially thousands) of plug-ins.[8] Plug-ins may rely on and provide services to other plug-ins. This allows for better modularity and the reuse of functionality to build applications not intended by the original plug-in developer. Plug-ins describe their dependencies using a descriptor (or manifest), which consists of a `plugin.xml` and `MANIFEST.MF` file. This descriptor tells the runtime what other plug-ins and Java packages are needed in order for the plug-in to resolve properly. In Eclipse, the dependent plug-in only loads if requested by the depending plug-in. On startup, the plug-in loader scans the `plugin.xml` or manifest file for each plug-in and builds a structure containing this information.[9] This structure is used later to load the appropriate classes from the correct plug-in.

A plug-in provides extensibility through extension points. These extension points define how the feature of a particular plug-in can be extended by other plug-ins, which can also provide extensions to the extension points, as shown in Figure 10. An extension point is the definition of a port – an entry point for other plug-ins to offer services. The extension implementation is the actual service, packaged in a

2. Eclipse.org, available online at Eclipse.org
3. Eclipse RCP, available online at http://wiki.eclipse.org/index.php/Rich_Client_Platform
4. Eclipse Newcomers FAQ, available online at http://www.eclipse.org/home/newcomers.php
5. Equinox, available online at http://www.eclipse.org/equinox/
6. P. Carlson, 21 Aug. 2007, IBM, Getting started with the IBM Lotus Expeditor Toolkit 6.1, available online at http://www.ibm.com/developerworks/lotus/library/expeditor-toolkit/
7. OSGi Home page, available online at http://www.osgi.org/Main/HomePage
8. E. Clayberg, D. Rubel, 2004, Pearson Education Inc., *Eclipse, Building Commercial-Quality Plug-ins,* page 129
9. E. Clayberg, D. Rubel, 2004, Pearson Education Inc., *Eclipse, Building Commercial-Quality Plug-ins,* page 130

way that makes it callable through an extension point.[10]

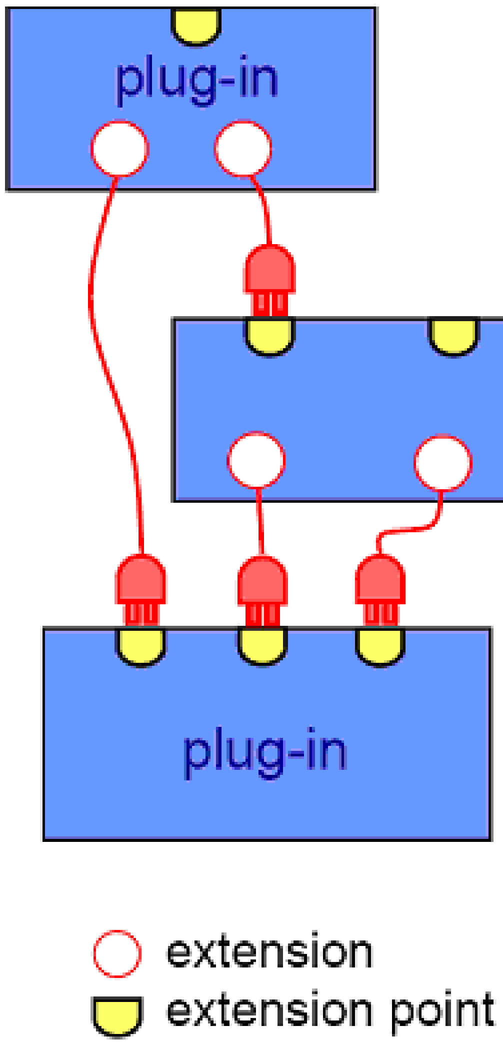


Figure 13. Extensions and extension points

One or more Eclipse plug-ins can be grouped together into an Eclipse feature so that a user can easily load, manage, and brand those plug-ins as a single unit.[11] Features allow the packaging of one or more plug-ins and usually describe a set of functionality that the containing plug-ins bring to the platform as a collective.

Sometimes it is useful to make parts of a plug-in optional, allowing it to be installed, uninstalled or updated independently from the rest of the plug-in. For example, a plug-in may have a library that is specific to a particular operating system or windowing system, or a language pack that adds translations for the plug-in's messages.[12] In this case you would want to implement a plug-in fragment. All three of these types of projects (plug-in, feature, and fragment) can easily be created using the Eclipse project wizard.

10. B. Marchal, 3 Feb. 2005, Working XML: Define and load extension points, available online at http://www.ibm.com/developerworks/xml/library/x-wxxm29.html

11. E. Clayberg, D. Rubel, 2004, Pearson Education Inc., *Eclipse, Building Commercial-Quality Plug-ins,* page 653

12. Eclipse wiki, *FAQ What is a plug-in fragment?*, available online at http://wiki.eclipse.org/FAQ_What_is_a_plug-in_fragment%3F

Features and plug-ins are installed from an Eclipse update site. An update site is a set of files and a directory structure that is recognized by the update manager. The article *How To Keep Up To Date*[13], is an excellent primer for creating features and update sites. An update site can be referenced in a local directory structure, on a web site and, in the case of Lotus Notes, inside of a notes database that uses the Update Site template. This gives Lotus Notes administrators a lot of options for how they want to administer their feature deployment. For example, Lotus Notes shops that do not deploy a Domino server can use basic NSF technology (over NRPC) to deploy features. If Domino servers are in the environment, then the update sites will simply be a directory structure on the server or a container in a database that inherits the Update Site template. The Composite Application Wiki [14] has a section dedicated to the management of features and plug-ins.

The Eclipse Workbench

The Eclipse workbench is where the entire user interface is defined. Like Eclipse, Lotus Notes has the same definition and uses the same technology under the covers. The Workbench provides the UI building blocks that make Eclipse applications easy to write, easy to use, scalable and extendable.[15] The workbench defines an explicit lifecycle for the window, toolbars, action menus and components. Lotus Notes provides its own Workbench advisor and also offers a convenience layer above it called the Personality framework. The personality of an application defines the framework the platform uses to determine what perspectives or windows, menus, action bar items and status line controls are displayed when the application starts.[16] This means you can build a custom look-and-feel, or "branding", for your entire client platform.

Being a scalable and extensible platform means you can contribute extensions to many parts of Lotus Notes 8 in ways you could not do very easily in previous versions of Lotus Notes.

Threading 101 for Eclipse and Notes

There are two basic sets of threads in the rich client platform – threads that run in the UI and threads that run in the background. For a best practice, you should use the Eclipse Job [17] application programming interfaces for all of your threading needs. The Job API’s give you easy to use classes for both background and foreground (user interface) threads. The API’s also allow easy integration with the progress monitoring dialogs that come with the base Eclipse framework. This means your job status can display in the status bar or in a modeless dialog that shows the percentage complete. The API comes complete with scheduling and run rules to insure you have complete control over the execution of the thread.

For jobs that you want to run against Lotus Notes data (threads requiring a Notes Session) you should use the `NotesJob` class. This job class ensures you have proper session access to the Notes runtime. See Table 2 on page 30 for the list of job

13. D. Glozic, D. Birsan, 27 September 2003, IBM, *How To Keep Up To Date*, available online at http://www.eclipse.org/articles/Article-Update/keeping-up-to-date.html

14. Composite Application Wiki, IBM, *Introduction to provisioning and updating*, available online at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/introduction-to-provisioning-and-updating

15. J. McAffer, J. Lemieux, 2005, Addison-Wesley, Eclipse – *Rich Client Platform,* page 215

16. Personalities, 2007, *Lotus Expeditor InfoCenter*, available online at http://publib.boulder.ibm.com/infocenter/ledoc/v6r1/index.jsp?topic=/com.ibm.rcp.tools.doc.appdev/ui_usingpersonalities.html

17. M. Valenta, 20 September 2004, IBM, *On the Job: The Eclipse Jobs API*, available online at http://www.eclipse.org/articles/Article-Concurrency/jobs-api.html

classes and their descriptions.

Table 2. Job classes and descriptions

Package	Class	Description
`org.eclipse.core.runtime.jobs`	`Job`	A `Job` is a units of runnable work that can be scheduled to be run with the job manager
`org.eclipse.ui.progress`	`UIJob`	The `UIJob` is a `Job` that runs within the UI Thread using an asyncExec
`com.ibm.notes.java.api.util`	`NotesJob`	Eclipse `Job` wrapper for the `NotesThread`

Making your Eclipse plug-ins dynamic aware

One of the key factors of the Eclipse platform is its ability to be completely dynamic. Which means your plug-ins can come and go in the runtime and not adversely affect other plug-ins still running. In order to "fit" within this platform and function properly with other installed plug-ins, your plug-in must follow some basic guidelines. Your plug-in must be dynamic aware and dynamic enabled.

Dynamic-awareness has to do with updating your plug-ins data structures in response to changes in the set of installed plug-ins.[18] This means your plug-in must know when other plug-ins are added or removed from the system. The Eclipse runtime broadcasts registry change events to any registered listeners. To register your plug-in as a registry change listener, consult the Eclipse Javadoc.[19] Once registered, your plug-in can react to the addition or removal of other plug-ins.

Dynamic-enablement means writing a dynamic-enabled plug-in so that it correctly handles its own dynamic addition and removal. This means properly cleaning up objects and cache to prevent leaks. These should be handled in the `start()` and `stop()` methods of your plug-in (or bundle). Objects to look for are files, graphical objects (images, fonts, colors, and so on) and socket connections.

Creating your first Eclipse view

One of the first things you are likely to attempt is creating an Eclipse view part. View parts are the corner stone of Composite Applications. They are the primary "wrapper" for all component types. If it is a component, then it is ultimately an Eclipse view part. Since the Composite Application infrastructure uses views and dynamically adds views to the perspectives, you need control of the secondary identifier of the view part. This means all views that are to be "components" must allow for multiple instances. The `allowMultiple="true"` setting must be set in the view extension definition. You can learn more about creating Eclipse based components in "Building Java components" on page 37.

18. J. McAffer, J. Lemieux, 2005, Addison-Wesley, *Eclipse – Rich Client Platform*, page 357

19. Eclipse.org, *Extension Registry API's*, available online at http://help.eclipse.org/help32/topic/org.eclipse.platform.doc.isv/reference/api/org/eclipse/core/runtime/IExtensionRegistry.html

Building NSF components

This section describes how NSF components can be developed with Lotus Domino Designer, as well as how existing NSF databases can be used in Composite Applications.

What are NSF components?

A component in a Composite Application is basically a rectangle in the UI. A NSF component displays a Notes object like a Notes view in a rectangle. A NSF component can be configured differently for every occurrence in an application as part of the Composite Application using static component preferences. Additionally, a NSF component defines its output properties and actions.

In summary, a NSF component consists of:

- A Notes URL
- Static component preferences (optional)
- Output properties and actions (optional)

The lead manager sample application [20] shows how four different NSF applications can be set as NSF components in a Composite Application.

NSF components can be added to the palette in the Composite Application Editor (for more information, refer to "Using the Composite Application Editor" on page 45) by selecting **Add Components > Add NSF component**. From the CAE, a NSF component picker allows you to navigate to a specific Notes database and then pick the Notes object you want. Alternatively, NSF components can be added to the "MyWidgets" category on the palette from the Notes client directly by opening the NSF object and then selecting **Configure widget from current context**.

Notes URLs can be used to refer to Notes objects. When using a Notes view the navigator can be hidden through the URL parameter `&HideNavigator` [21]. The property broker can only be used from Notes forms and Notes views.

The Notes URL can contain paths to databases or it can contain replica IDs. To identify special types of databases, the following special 'replica ID' can be used [22]:

- `notes:///0000000000000E00` - Opens the current user's mail database.
- `notes:///0000000000000E01` - Opens the current user's contacts database.
- `notes:///0000000000000000` - Searches for a NSF component in the same database that has the Composite Application XML file.

20. *Sales lead manager sample application overview*, available online at: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/ibm-lead-manager

21. Refer to the *Manage Real Estate for NSF Components in Composite applications* section of the *Composite Applications wiki*, available online at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/managing-real-estate-for-nsf-components-in-composite-applications

22. Refer to *Deployment with NSF based Composite Applications with components in the same NSF* section of the *Composite Application Wiki* available online at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/deployment-of-nsf-based-composite-applications-with-components-in-the-same-nsf

Deployment of NSF components

Composite applications are typically developed in non-production environments and test databases are often used to test the applications. When moving the Composite Applications with hardcoded links from a test database to a production environment the links can break. To avoid this, relative links can be used through `@Formulas` that are executed in the database containing the Composite Application XML file. The `@Formulas` are defined in the advanced component properties in the CAE (`com.ibm.notes.ComputedNotesURL`). For example, in these `@Formulas` you can access fields in Notes profile documents that contain the links to the databases. The values in these profile documents can then be changed using LotusScript as part of an automated deployment. For more details, refer to the section *How to calculate links to NSF Components at Runtime* in the Composite Applications wiki, available online here http://www-10.lotus.com/ldd/compappwiki.nsf/dx/how-to-calculate-links-to-nsf-components-at-runtime.

Using the property broker - built in properties and actions

Composite Applications allow the reuse of existing NSF applications. In many scenarios, it is desirable to use NSF components without changing their designs. One reason is to avoid development work for extending existing NSF applications. When building their first Composite Applications in proof-of-concepts, developers often do not have designer access to the deployed databases that they want to reuse. Another reason is to simply avoid the redeployment of the changed databases for various reasons such as avoiding retesting the databases.

Because of this, Notes has built in properties and actions for NSF components. These properties and actions can be used without changing the databases (no WSDL needs to be defined).

Built in property SelectedNotesDocumentURLChanged:
A typical scenario is that an action is supposed to be invoked when another view entry is selected. The built in property can be used to publish the Notes URL of the currently selected view entry. Other components can then use this URL as callback to access any field of the selected document using LotusScript or the Java API. To define which views this property is to be published from, the views need to be defined in the advanced component property `com.ibm.notes.publishBuiltInPropsFromView`.

Built in action FilterCurrentUIViewViaCategory:
In order to filter the current UI view, a built in action can be used (constrained list). These views have to have a sorted column which is used as the key for the lookup. Only view entries are then shown, whose values of the first sorted column match with the input property of the action.

Built in action SearchCurrentUIView:
This action performs a full text search on the current UI view.

Built in properties and actions need to be enabled in CAE:

- On page level: `com.ibm.notes.enable.preferences=true`
- On component level: `com.ibm.notes.enableBuiltInPB=true`

Refer to the *Using built-in properties and actions in Notes 8.0.1* section of the *Composite application Wiki*, available online at http://www-10.lotus.com/ldd/compappwiki.nsf//dx/using-built-in-properties-and-actions-in-notes-8.0.1 for more details.

Using the property broker - custom properties and actions

For more sophisticated scenarios, there are no built in properties and actions available. For example, if you want to publish a column of a view on selection change rather than the Notes URL. Another example might be providing an action in a form that sets certain field values received from other components. A further example is the publication of a field value in a document to other components as a property.

In these cases you need to follow these steps:

1. Define the names of your output properties and actions and their data types using the Property Broker Editor included as part of Domino Designer (result: WSDL).
2. For actions:
   a. Map your Notes actions in forms and views to the (property broker) actions defined in the WSDL.
   b. Implement your action with LotusScript and use the property broker APIs to retrieve the input property.
3. For output properties:
   a. Properties that are declaratively published when view selections change: Map the column to an output property in the WSDL.
   b. Properties that are programmatically published: Use the property broker APIs in any events in forms and views (such as OnSelect) to publish any property you want.

See the tutorial *Building Composite Applications for IBM Notes 8* available online at http://www-128.ibm.com/developerworks/lotus/library/tutorials/notes8-comp-apps/lz-dw-lz-notes8-comp-apps.html?S_TACT=105AGX13&S_CMP=LP for more information. For more details on how to use the Property Broker Editor, refer to the *Property Broker Editor for NSF Components* section of the *Composite Applications Wiki*, available online at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/property-broker-editor-for-nsf-components. See in the Domino Designer help for more details on the LotusScript property broker APIs (LotusScript classes: `NotesPropertyBroker` and `NotesProperty`).

All data types are represented as single strings when accessing them from LotusScript. You can then initiate your own LotusScript classes with whatever information these strings contain. Essentially, you must deserialize the strings in the object structure you want to use. For example, one string could contain a list of URLs that another component passed to a Notes action. You could then represent this list of URLs in an array of strings in LotusScript or use your own classes.

For data types that are more complex than just strings (such as lists) it is recommended you use JSON to serialize and deserialize these data types from and to strings. JSON is much simpler to handle than complex XML and JSON libraries are available in many other languages that other components might use.

You can either use your own JSON library or find one on the internet. For example, SNAPPS has recently released a JSON library for LotusScript that is under the Apache license.

Using static component properties

Every component can be configured differently for every occurrence in a Composite Application. This is important when you want to provide a reusable component that can be used in many different scenarios that you might not have considered. For example, you might want to use another logo or caption to be displayed on your component dependent on the context it runs in.

In order to do this, static component properties can be set in the CAE from the **Advanced Component Properties**. Here you can define these static properties as opposed to the dynamic properties that the property broker uses.

NSF components can read the values of these properties using a LotusScript API or through a `@Formula`:

- `NotesUIWorkspace.GetComponentViewPreference`
- `@GetComponentViewPreference`

See the section *How to run NSF Components in Context* of the Composite Applications Wiki, available online at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/running-nsf-components-in-context for more details.

The same APIs can be used to receive properties that are not set as static component properties but that are passed in through a URL to open an application. Composite applications can be referenced using `CAI://` URLs that contain the IDs of Composite Applications and/or the `NRPC://` URLs pointing to the actual Notes database with the Composite Application XML. You can append your own parameters to these URLs:

```
CAI://... &CustomParam=CustomParamValue
```

NSF components can then receive the value of the custom parameter in the same way they read the static component preferences. These custom parameters in the URL do not have to be defined in the static component preferences. However, if they are defined in the static component preferences, the values are overwritten with whatever is passed in using the CAI URL.

For more information, see the section *Passing Context to Components when opening Composite Applications* of the *Composite Applications Wiki* , available online at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/passing-context-to-components-when-opening-composite-applications.

In some scenarios you do not have to know a static component preference, only whether your NSF component runs inside a Composite Application or not (standalone, as in Notes 7). When running a component inside of a Composite Application there is often less space for the component than running it standalone. In these cases, you might want to hide and show certain user interface controls. For example, you might want to hide certain view actions when running the view component as one of multiple components on a Composite Application page.

To find out the context your NSF components are running in, you can use any of the following APIs:

- `NotesUIWorkspace.IsInCompositeApp`
- `@IsInCompositeApp`

Open Notes documents on pages

Notes documents can also be put on pages in Composite Applications together with other components. However, documents are special compared to other Notes objects such as views. Composite Applications can have a fixed number of static pages. However, when documents are opened from views, each document is opened in a new tab. In order to do this, template pages can be defined in the CAE that are then cloned for each document at runtime. On this template page, a placeholder component (`com.ibm.notes.isDocumentPlaceholder`) points to a Notes form of the documents that are supposed to be shown.

In order for Notes to know when to open a document 'standalone' as opposed to on a Composite Application page, a naming convention is used. If there is a page with the same alias (`com.ibm.rcp.alias`) as the alias of the form, the cloned page is used to show the document when triggered from a Notes view. See the section *Open Notes Documents on Pages* of the Composite Applications Wiki, available online at http://www-10.lotus.com/ldd/compappwiki.nsf/dx/opening-notes-documents-on-pages for more details.

What are NSF widgets?

NSF widgets are essentially special types of NSF components. They can be used in Composite Applications exactly like NSF components. NSF widgets can be added to the "MyWidgets" category on the CAE palette from the Notes client directly by opening the NSF object and then selecting **Configure widget from current context**.

In addition to NSF components, widgets provide more capabilities:
- NSF widgets can be put in the sidebar
- NSF widgets can be triggered from live text

In order to put a NSF widget in the sidebar, line of business (LOB) users can use the **Add to sidebar** toolbar icon to add the currently opened Notes object (such as a Notes view) to the sidebar with one click. This icon also creates the widget and adds it to the "MyWidgets" sidebar and to the "MyWidgets" category in the CAE palette. NSF widgets can also be added to the sidebar by administrators using the catalog. In both cases, it can be defined whether the NSF widgets should be shown in the sidebar on client startup (`contributeToSideshelfOnStartup="false"`). Note that NSF widgets can not use the property broker to interact with other components when running in the sidebar (Notes 8.0.1).

NSF widgets can either point to a Notes object, such as a Notes view, or they can be used to perform a full text search for a specific live text. You can define a full text search NSF widget by opening a view that you want to query and then selecting **Configure a widget from current context**. You can then define the NSF full text search widget to be used for specific live text or a text selection within a Notes document. The search results can be shown as any widget either in the sidebar, floating window or tab (for more information, refer to "Widgets" on page 44).

Building Java components

When talking about IBM Lotus Notes, most references to Eclipse are specifically referring to the Eclipse Rich Client Platform (RCP), which is the basis for Lotus Expeditor, Lotus Notes, Lotus Sametime, and Lotus Symphony. Before the RCP platform emerged from Eclipse.org, Eclipse was a highly pluggable Integrated Development Environment (IDE). It is in the IDE context that this section discusses Eclipse. Eclipse is a natural tooling platform for Eclipse RCP based runtime environments such as IBM Lotus Notes. The Eclipse IDE, specifically the Java Development Tools (JDT) and the Plug-in Development Environment (PDE), provide the base platform on which the IBM Lotus Expeditor Toolkit is built. The IBM Lotus Expeditor Toolkit allows Java developers to build plug-ins for the IBM Lotus Notes Client. Additionally, the toolkit allows developers to build Web Applications, Web Services, and Portlets for execution on the client platforms. This section describes some of the tasks that can be used to create Java components for use in IBM Lotus Notes.

The Lotus Expeditor Toolkit

IBM Lotus Expeditor Toolkit provides a complete, integrated set of tools that allows you to develop, debug, test, package and deploy client applications to IBM Lotus Expeditor 6.1, IBM Lotus Sametime 7.5.1, and IBM Lotus Notes 8. You can use the toolkit to develop the following types of client applications:

- Eclipse Rich Client Platform (RCP) applications (desktop client only)
- Eclipse embedded Rich Client Platform applications
- Web applications
- Embedded transaction applications
- Portlet applications (desktop client only)
- Database applications using DB2e and Derby
- Messaging applications (JMS)
- Web services applications

The toolkit provides wizards that enable you to create Client Services projects to develop client applications. The toolkit uses target definitions to provide a convenient method for you to specify the runtime environment, the build-time environment, and the set of components that can run on the platform. For example, when you create a Client Services project, you select a target definition from a list of available targets, and then a set of features from a list in the target definition. The toolkit uses this information to automatically set up the Java build path and runtime for your project. You can then edit, compile and debug your project. The toolkit provides a default list of target definitions; however, you can also create your own definitions.

Using the toolkit

For detailed configuration and usage instructions for the Expeditor Toolkit and Notes 8, please see http://www.ibm.com/developerworks/lotus/library/expeditor-notes-sametime/.

Creating Components

The Expeditor Toolkit allows developers to create different types of Java components for deployment to IBM Lotus Notes. The toolkit provides multiple project types under the Client Services category. For the Lotus Notes developer, the most commonly used projects are “Client Services Portlet Projects,” “Client Services Web Projects,” and the standard “Client Services Projects.”

The standard Client Services Project is the project to use when creating Eclipse SWT based user interfaces, business logic, and other types of standard Java code. The Client Services Portlet Project is used to create new portlets or reuse existing Portlets on the Notes client. Finally, the Client Services Web Project aids in the creation of new web applications (JSP/Servlet based) or the reusing of existing web applications on the Notes client. The following sections illustrate how to create new components of different types. See the *Developing Applications for Lotus Expeditor 6.1.x* documentation, available online at http://publib.boulder.ibm.com/infocenter/ledoc/v6r11/index.jsp for additional details on these project types and the other types of projects available.

Using Notes.jar

A plug-in within the Notes platform called `com.ibm.notes.java.api` allows developers access to Notes and Domino data. By using this plug-in, applications can directly read and write to Notes databases and documents. This allows Eclipse-based components to leverage and extend any systems in place. In particular, this allows sidebar and toolbar components to be more interactive with the data that is most important to the user. More information and detailed examples are available here: http://www.ibm.com/developerworks/lotus/library/notes8-data

Notes Extension points

When a plug-in wants to allow other plug-ins to extend or customize portions of its functionality, it declares an extension point. After an extension point has been defined by a bundle in the platform, other bundles can extend this extension point to contribute their piece of functionality. For example, the Eclipse workbench defines an action set extension point that allows other plug-ins to contribute items to the menu bar or the toolbar. The declaration of the extension point and any extension to it is done declaratively in a `plugin.xml` file. Having a solid understanding of the Eclipse extension point framework is important if you want to be able to extend the Notes or Expeditor workbench with your application. For additional information about the fundamentals of extension points, visit the Eclipse site.

The Lotus Notes Client can be extended in several ways, including:

Contributing a shelf view to the sidebar
The Lotus Notes sidebar provides significant screen real estate and, as such, is very a important contributable part of the workbench. Contributing a component to the sidebar is no more difficult than creating a standard Eclipse view.

Contributing to the toolbar
Another important contribution point in the workbench is the ToolBar/CoolBar. This provides quick access to any frequent operations, eliminating the need to navigate through the top level menus.

Contributing to a top level menu

You can contribute to top level workbench menus, such as **File > Open >** ***Your entry***.

Working with content selections

One of the main values of having a rich client platform, is having a framework in place that allows extensions to know the current user context.

Contributing to a context menu

Many items in Lotus Notes provide context menus for the user. By right-clicking on an item, the user is able to see all the actions that can be performed based on that selection. The entries on these menus, in fact, represent a hierarchy of selection.

Tracking the Selection

A component that is visible in a global location, such as the toolbar or sidebar, might find it useful to track the current selection, as it represents the user's current context.

For more information on this topic, see the developerWorks® article entitled "*Leveraging user context in the IBM Lotus Notes 8 sidebar and toolbar*" at the following location: http://www.ibm.com/developerworks/lotus/library/notes8-context

Drag and Drop

You can drag and drop certain selected items between components. Notably, Notes 8.01 has the ability to drag documents and attachments out of Notes views and documents and drop them on Composite Application and sidebar components. In Eclipse-based components, drag and drop is performed through Transfer types. More information is available here: http://www.eclipse.org/articles/Article-SWT-DND/DND-in-SWT.html

Creating your first Eclipse component

Creating your own custom component for use in a Composite Application is as easy as creating an Eclipse plug-in project with an SWT view. After the simple view component has been created you can add it to the Composite Application Editor (CAE) palette and then to your Composite Applications. The processes consists of:

- Creating a client services project
- Adding a view to your project
- Adding a UI to your component
- Adding the component to your Composite Application

For a full tutorial, refer to Developing an Eclipse component for Lotus Notes 8 (part 1) and Developing an Eclipse component for Lotus Notes 8 (part 2).

Custom Navigator for a Composite Application

To obtain a custom look and feel, developers may desire to create a custom navigator to move between the pages of their Composite Applications, instead of using the default navigator. Creating a custom navigator gives you the following advantages:

- Options to change the look and feel of the navigator
- Options to change the screen location of the navigator
- Extended options to control visibility

- Can be arranged similarly to other components, to conserve and maximize work space
- Can interact and communicate with other components

Composite Application Infrastructure, Topology, and Navigation page hierarchy and preferences

In order to create a custom navigator for Composite Applications, you should have a high level understanding of the underlying data structure for Composite Applications. Composite applications consist of one or more application pages, grouped together in a hierarchy. At the top of the hierarchy is the root page. The root page does not appear within the Composite Application Editor (CAE), and you cannot switch to it or add components to it. However, it is provided through the Topology Handler as a consistent starting point to group top level pages. These top level pages, in turn, can be containers for other child pages, and so on for as many levels as required.

Clients can access the page hierarchy model through the `TopologyHandler` service. The `TopologyHandler` provides an API to both the hierarchical page model and the components within each page. Each page and component within an application can contain any number of name/value pairs representing meta-data for the object.

Lastly, when creating a custom navigator, you will need to configure some application and page properties in order to emulate some of the original functionality of the default navigator. The properties tell the underlying Composite Application infrastructure that you are overriding the default navigation system. First, from the advanced page properties settings window in the Composite Application Editor, set `com.ibm.rcp.navigationModel` as "`custom`" and `com.ibm.rcp.useNavigator` as "`true`" . This makes each perspective show in the same tab. By default, these application page settings display each page of your Composite Application in the same tab. For any page you want displayed in a separate tab, set `com.ibm.rcp.useNavigator` to "`false`" for that page's properties.

Composite Application Data Model and the TopologyHandler Service

With Lotus Expeditor, pages are implemented as Eclipse perspectives. As a result, you can navigate between pages by using the Eclipse platform routines to navigate to a perspective with the right ID. The platform, however, assigns the IDs for pages, and the Topology Handler maintains them. They must be retrieved from the Topology Handler in order to be used. In turn, to retrieve the pages in an application you must have the ID for that application, which can also be retrieved from the Topology Handler.

Default Navigators

All three of the following navigators (figures 11, 12 and 13) share the exact same content and label providers. The only difference is the implementation of the look and feel, which is done in the viewer where the widget lives.

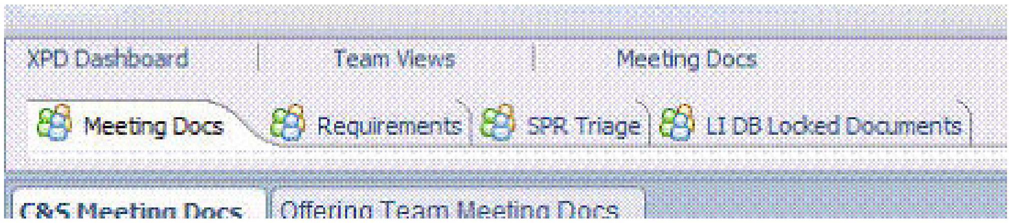


Figure 14. Tab navigator with breadcrumb support

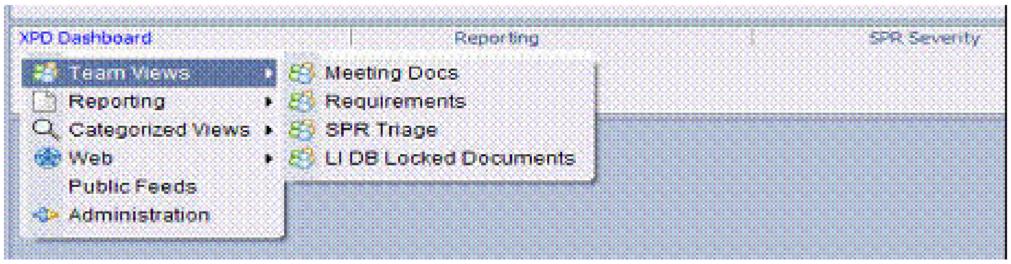


Figure 15. Breadcrumb navigator

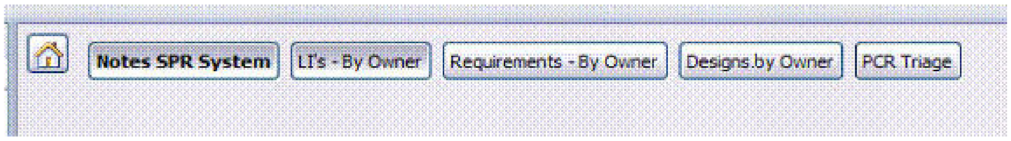


Figure 16. Button navigator

Send and receive properties from SWT views

When you have SWT view components in your application you may want to have these components communicate with each other. SWT views can publish and consume property changes to and from the property broker. Some views publish properties as values change, while some views consume properties published elsewhere. The property broker has to be aware of any property that a component wants to publish or consume. Thus, a property must be registered with the property broker. After the property has been defined and registered with the property broker, it can be used in your classes.

In order for the property broker to know your component is to be notified of a specific property change, you must wire your component to another component that publishes that property. Wiring a publisher and a consumer is done through the Composite Application Editor (CAE). You can find more information on wiring components in "Assembling Composite Applications and creating widgets" on page 43.

Launch and debug of the Notes Client

The Lotus Expeditor Toolkit provides a custom configured launcher that can be used to launch Lotus Notes without the need for additional configuration settings. You can access the launcher, named Client Services, by choosing **Run... > Run** from the IDE menu. Select the Client Services launcher in the left pane, and then click the **New** button. This creates a new instance of the Client Services launcher.

If you select the Target tab of the launch configuration, you can see that the Lotus Notes 8 target has been selected already. Because there can be only one Lotus Notes instance running at any time, shut down Lotus Notes before launching from the IDE. Clicking the **Run** button launches the Lotus Notes 8 platform. If running with Lotus Notes 8.0.1, select it from the dropdown list, You can make Lotus Notes 8.0.1 the default selection by changing the **Default Target Selection** from the **Window > Preferences > Client Services > Development preference** page.

Additional instructions on using the run and debug capabilities of the Expeditor Toolkit can be found at http://www.ibm.com/developerworks/lotus/library/expeditor-notes-sametime/.

After you log into Lotus Notes, you can see your normal Lotus Notes workspace. Java applications running in the Notes environment can be debugged in the standard Eclipse manner. For more details on debugging application from Eclipse, see Debugging with the Eclipse Platform at http://www.ibm.com/developerworks/java/library/os-ecbug/.

Assembling Composite Applications and creating widgets

This section explains how to assemble Composite Applications using the Composite Application Editor (CAE) in IBM Lotus Notes 8 and IBM Lotus Expeditor 6. Additionally, we discuss the concept of widgets as introduced with Notes 8.0.1. CAE and widgets are related because neither require any programming skills.

Composite Applications

As outlined in "Business value of Notes 8.x application development " on page 3, the Composite Application technology offers benefits on many levels to companies and organizations. In fact, this technology is part of the Notes 8 core infrastructure, as the standard Notes Mail Application has been built as a Composite Application. It contains the Notes Mail Navigator, the Notes Mail View and the Notes Mail Mini View. A Composite Application usually consists of a root page and many child or sibling pages which induce the basic structure of the application. Components are place on these application pages, as well as connections ("wires") between the components for their communication and reactive interaction. Components are coupled loosely. This means that components do not depend on each other and can be used in any Composite Application. A component never calls services of other components directly. Instead, components produce properties (typed exchangeable data items) which can be consumed by the actions of other components. Thus, Composite Applications provide a frontend of a service oriented architecture (SOA). Using the Composite Application Editor (CAE), it is very easy to design the layout of Composite Applications and to specify component interaction. Please refer to "Using the Composite Application Editor" on page 45 to learn more about the CAE.

From the user's perspective, every component instance is represented by a view part which displays as a rectangle in the UI. Internally, a component also has a set of static component properties that can be set with the CAE (for more information, see "Working with the Composite Application Editor" on page 46). The component's properties and actions, which are used for component interaction, are defined in the component's WSDL file. The section "Wiring Components" on page 47 discusses the Composite Application Editor's wiring UI, which illustrates a property of one component being wired to an action of another component.

Predefined Components in Notes

Besides the various component types which are listed and explained in "Architecture overview" on page 21, Notes comes with various predefined components which are part of the default installation.

PIM components
Personal information management (PIM) components enable the Notes Calendar, Contacts, Mail and ToDo features for Composite Applications. They can be found in the component palette in CAE (see "Using the Composite Application Editor" on page 45) and added to any Composite Application in a straightforward way. Thus, for instance, the standard Notes Mail View can be included in a Composite Application and connected with other line of business components. For more information and a sample see Using PIM Components.

Embedded browser (since 8.0.1)
The embedded browser displays web pages. For example, if a component provides a URL as a property, it can be wired to the embedded browser. It would then load the site specified by the URL provided as property from another component. For more information, refer to Using the Embedded Browser Component to learn more.

Symphony document view (since 8.0.1)
This component displays Symphony documents: word processing, spreadsheets and presentations. For more information, refer to Using the Symphony View Component.

Enabling Notes Users to Use the Composite Application Editor

In Domino server, the Desktop policy settings document contains a setting, **Allow user to use the Composite Application Editor**, that you can set to **Enabled** to allow Notes Client users to edit Composite Applications with the Composite Application Editor. To access the field, in the Desktop policy settings document, click the **Database** tab and scroll to the section **Composite Application settings** .

Widgets

As explained in "Architecture overview" on page 21, the IBM Lotus Notes My Widgets feature enables end users to see and act on automatically recognized Live Text in their Notes documents, including mail, using widgets created specifically for their use. In the context of Composite Applications, widgets offer the simplest way to create components for use in Composite Applications. Any widget created by a user is available as a component on the component palette in CAE (see "Using the Composite Application Editor" on page 45) and can be used in Composite Applications. For example, the section "Using static component properties" on page 34 describes how NSF widgets can be created for use as special types of NSF components.

For detailed information on creating and using widgets, refer to the Domino 8.0.1 Administrator Help.

Domino Server Policy for Widget Enablement

By default, IBM Lotus Notes users will not see the My Widgets sidebar panel. However, they can display it by clicking **File > Preferences > Widgets** and enabling the **Show Widget Toolbar and the My Widgets Sidebar panel** option. Enabling this option allows for full My Widgets access. However, the administrator can use IBM Lotus Domino™ policy or Notes `PLUGIN_CUSTOMIZATION.INI` preferences to control user access to various aspects of My Widgets functionality. For example, a policy can be used to hide the My Widgets user interface such that the end user cannot work with My Widgets. Please refer to "Catalog Access Rights" on page 55 for more information about these policies.

Using Widget Wizards

You can configure a new widget by displaying the widget context and then starting the widget configuration wizard, which guides you through the process of creating and configuring the widget. Widgets can be configured from any of the following contexts as listed in the user interface:

- Notes view, form, or document
- RSS/Atom feed
- Web page

- Google Gadget

Wizards are available for all widget creation tasks. Click the **Getting Started with Widgets** toolbar button to begin. Alternatively, you can navigate to the IBM Lotus Notes view, form, or document, or Web page, gadget, or feed target and then click the **Configure a Widget from Current Context** toolbar button to begin creating the widget within the context of the active Notes or Web item. The general process for configuring a new widget is as follows:

1. Start the appropriate widget configuration wizard in one of the following ways:
   - Open the Notes view/form/document, Web page, RSS/Atom feed, or Google Gadget from within Notes and click the **Configure a Widget from Current Context** toolbar button. This starts the appropriate wizard based on the active context.
   - Allow the wizard to fully guide you through the process of building the widget by clicking **Getting Started with Widgets** . Then select one of four types to base the new widget on, as instructed by the wizard.
2. Use the wizard prompts and fields to configure the new widget.
3. Test the new widget.
4. Publish the widget to the catalog on the designated server (please refer to "Deploying applications" on page 51)

If possible, the **Add to Sidebar** toolbar button creates a widget from the current context and adds it to the sidebar in a single step. You can also install widgets using XML extensions obtained from E-mail attachments or from a file server. After you have added widgets to the My Widgets sidebar panel, you can publish them to the catalog to be provisioned to various users. Please refer to "Deploying applications" on page 51 for information about widget deployment.

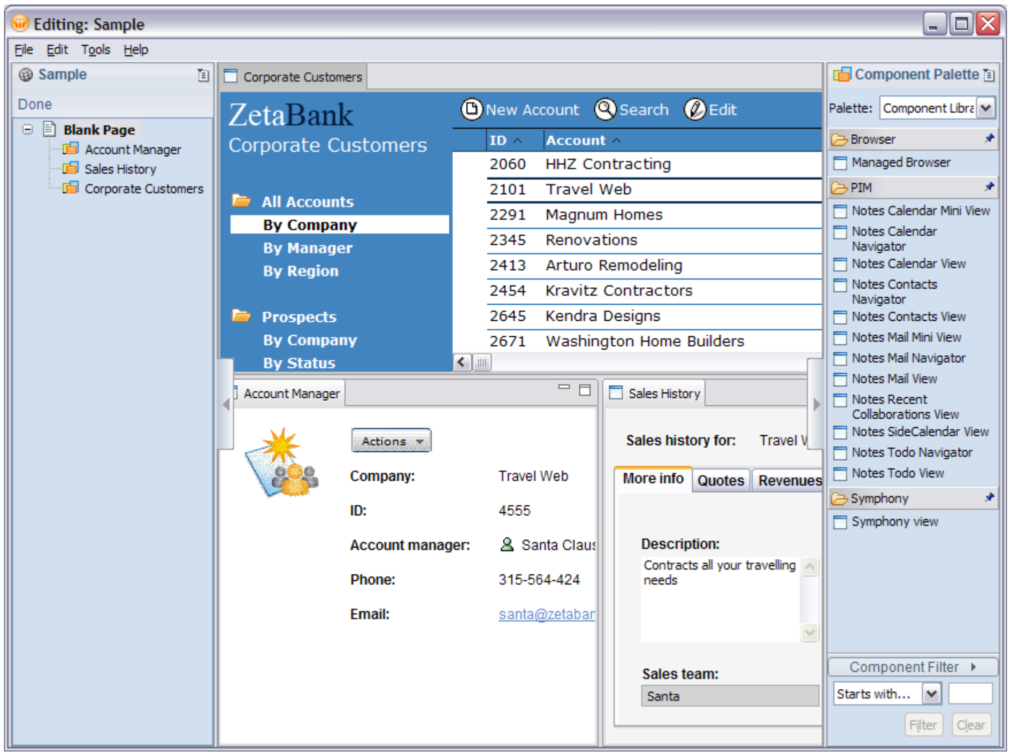
It is very easy and straightforward to create a widget as the dedicated wizards guide the user through all necessary steps. After creation, a new widget is added to the My Widgets sidebar panel which is also available as a component on the CAE palette.

Using the Composite Application Editor

The Composite Application Editor (CAE) is the main tool to assemble or edit Composite Applications. To see a list of Composite Applications that reside on a Domino server, select **File > Open > Lotus Notes Application** (or press **Ctrl-O**) to browse for an application on a server or on the local host.

Composite applications hosted on an IBM WebSphere Portal are listed in the Portal applications catalog. To see the Portal applications catalog or the Portal template library, you must have a home Portal account specified as part of your user preferences. You can view the Portal applications catalog by choosing **Open > Portal Applications**. Only the applications and templates to which you have access display. Because the CAE, in principle, works equivalently on any application type, the remainder of this section focuses on non-portal applications.

If you have access to edit an application, you can use the CAE to edit any active application. The changes that you make reflect immediately in the application. To edit a Composite Application, open the application and select **Actions > Edit Application**. When editing an application, your workplace displays an application menu pane on the left that provides you with direct access to edit application elements (pages and components), and a component palette on the right, which lists components you can add to your application. The center of the workspace displays the application components for the current page. Figure 17 illustrates the CAE user interface.



ID	Account
2060	HHZ Contracting
2101	Travel Web
2291	Magnum Homes
2345	Renovations
2413	Arturo Remodeling
2454	Kravitz Contractors
2645	Kendra Designs
2671	Washington Home Builders

Company:	Travel Web
ID:	4555
Account manager:	Santa Claus
Phone:	315-564-424
Email:	santa@zetabar

Sales history for: Travel W

Description:
Contracts all your travelling needs

Sales team:
Santa

Figure 17. The Composite Application Editor (CAE)

Working with the Composite Application Editor

On the palette, you can select between different palette instances. The read-only Component Library palette contains components that come with the Notes installation arranged into appropriate categories. In the My Palette instance, new components can be added. If the component palette does not already contain the components you need for your Composite Application, you can import a palette of components, if one is available, or add components to the palette for use in the application.

Adding Components to the Palette

In order to add components to My Palette, right click and select **Add Components**. This gives you the option to add a component from an update site, locally installed components or NSF components. Eclipse components are usually packaged as an update site as explained in "Getting started with Eclipse as a Notes developer" on page 27. Locally installed components have been added to Notes using its application management facility (see "Deploying applications" on page 51). They could also be part of the Notes install or contained in the workspace of Notes, running through the Eclipse IDE, and can thus be included in the palette. NSF components can be added from Domino servers or from the local machine. For Composite Applications hosted on a Portal server, portlets can be added from the server to the palette.

Creating the Application Layout

You can add pages to organize the components in an application. To add a page, right-click the application name (for adding a top-level page) or any existing page and then:

- click **Add page > After...** to add a page at the same level as the selected page.
- click **Add page > As child...** to add the page as a child page of the selected page.

For the new page, a page name must be provided and an optional description can be entered. Existing pages can be reordered in different ways: they can be moved up or down on the same hierarchy level or they can be moved between different levels. Adding components to an application is simple, you only need to select a component on the palette and drag it to the desired page in the center of the CAE workspace. A placement box shows you the regions where you can drop the component on a page: either in a tab structure or side-by-side in any direction. To reposition a component, just drag it to a new position on a page.

Setting Page Properties

Pages have properties and there is a dedicated dialog for setting them. Thus, it is possible to edit the title and description of a page and also to provide translations into different languages for both the page title and the page description. An advanced properties dialog gives access to edit settings for more sophisticated settings such as the association of an icon with a page. This dialog lists all advanced property names and their associated values which can be edited here. Every application has a root page, which cannot be deleted and which also has some specific properties not available to other pages. These properties act as application wide preferences (such as, for specifying the navigation model). Please refer to http://www-10.lotus.com/ldd/compappwiki.nsf/dx/advanced-page-properties for a list of all available page properties.

Setting Component Properties

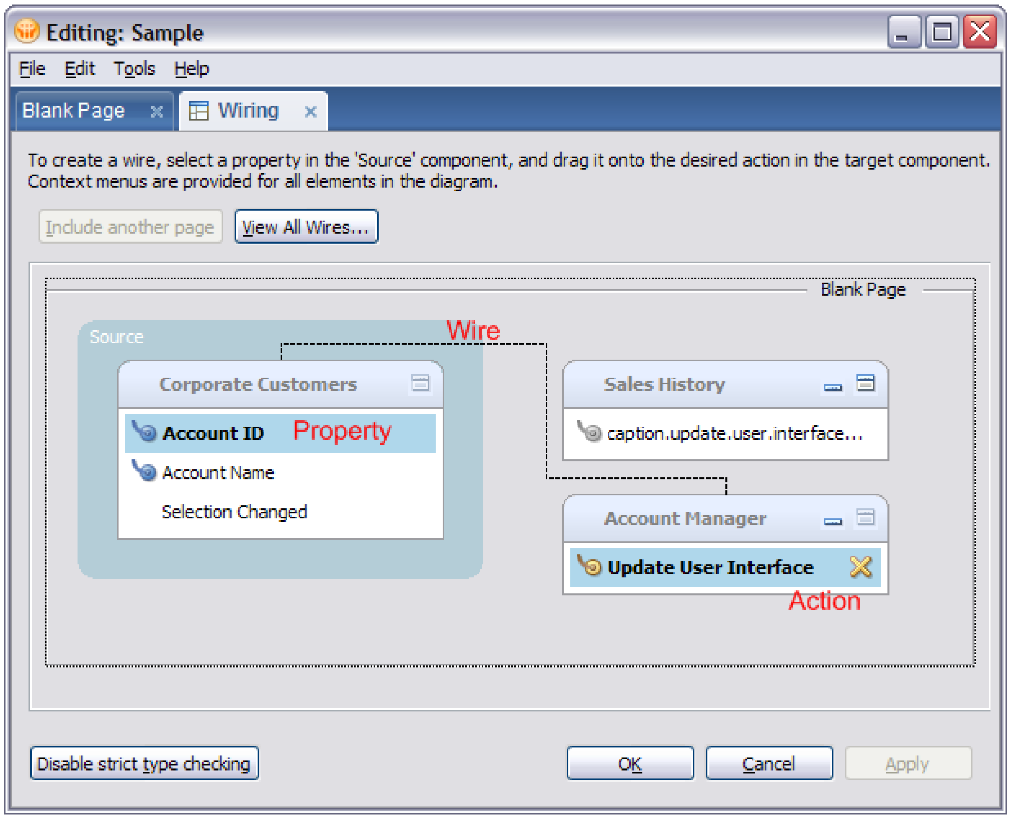
Similar to advanced page properties, there is also a dialog for setting advanced component properties. Here, for NSF components, the Notes URL which points to a certain design note in a specific Notes database can be set. Besides others, for Eclipse components, the provisioning details can be set and it is possible to flag components as hidden. Refer to http://www-10.lotus.com/ldd/compappwiki.nsf/dx/advanced-component-properties for a complete list of component properties and see "Building Java components" on page 37 for how to access these values from components programmatically.

Save the Application Changes Made with the CAE

In the CAE, any change is saved implicitly to the application definition in order to reflect changes immediately. When the CAE is closed, the user is asked if the changes should be kept or not. If the changes are requested to be discarded, the original version of the application from before launching the CAE is restored. Note that when working with the CAE, changes to an application hosted on Domino server are only persisted if the user has at least Designer access.

Wiring Components

Communication or interaction between components is achieved by "wiring" properties of components to actions of other components. When the value of a property changes in the source component, the target component reacts to that change by executing the action the property is wired to. These interactions are managed by a property broker. The wiring interface allows you to configure connections, or wires, between cooperative components. Right-click a component in the navigation pane and click “Wiring”. The component is selected as the source component in the appearing wiring interface. It is a visual representation that shows the existing wires between components in an application. It also displays the properties from one component that can connect to another component. Figure 18 shows the wiring interface of the Composite Application Editor.



To create a wire, select a property in the 'Source' component, and drag it onto the desired action in the target component. Context menus are provided for all elements in the diagram.

Figure 18. The Wiring Interface of the CAE

To create a new wire, select a property in the source component and drag it to an action in the target component. A dotted line connects the two components and a plug icon displays next to the source property and target action to indicate the presence of a wire. The user interface for the wiring component displays one component as the source at all times but the source component can be changed by right-clicking on the title of the desired component and selecting **Select as wire source**. Only wires from the currently selected output property of the source component are shown. However, other properties and actions display a **wired** icon if they are currently wired to something. Alternatively, it is possible to view all wires by clicking the dedicated **View All Wires...** button.

Disable Strict Type Checking

In CAE’s default settings, it is possible to create a wire between the two components only if the property of one component is compatible with an action in another component.

Two properties are compatible if there is a match of the data type and name space between them. Sometimes, this type checking is too strict. For instance, a component might provide a property of type `lastname` while another component could have an action of type `surname`. Using strict type checking, these two types are not compatible, although both might represent a person's surname and their values might be given as simple strings. By clicking the **Disable strict type checking** button, wiring between this property and action is possible.

Cross page wiring

Wiring is not restricted to components residing on the same page. The **Include another page** button allows users to include other pages and their components into the wiring interface. Wires can then be created in the same way as for components on the same page. In terms of the application's behavior, whenever a property is changed in the source component, the other page displays and shows the reaction in the wired component.

There are many more possibilities to create sophisticated applications and the next section provides some links for further exploration.

Deploying applications

The previous chapters focus mainly on how to build components and Composite Applications. This section describes how Composite Applications can be deployed to production environments. Additionally, widget catalogs are discussed.

IBM Lotus Notes/Domino

This section explains the available deployment mechanisms specific to Composite Applications using Lotus Notes and Domino. For example, how to create an NSF update site for deploying Eclipse components.

Creating an NSF Update Site

As explained in "Getting started with Eclipse as a Notes developer" on page 27, Eclipse components are packaged as Eclipse features and plug-ins and usually put as Eclipse update sites on an HTTP server. Additionally. Notes/Domino 8 allows you to create an NSF-based update site. This allows for installing and updating custom or third-party Eclipse features into Notes 8. You can create an NSF-based update site (which is essentially a specific NSF database) using the `updatesite.ntf` template. This template dynamically provides the `site.xml` and it can also contain the desired Eclipse features and plug-ins.

There are some advantages of NSF-based update sites compared with "classic" update sites:

- You can assign access rights to the update site/database.
- An NSF-based update site can be easily distributed using Domino replication.
- Easy-to-read Notes documents provide intelligent analysis and meta-data presentation of the entire update site, (for example, all plug-ins used by a feature are doc-linked.
- `updatesite.ntf` contains tools to globally and automatically modify embedded URLs inside plug-in JAR files.

The NSF-based update site feature in Domino allows for importing individual features, one ore more local update sites or another NSF-based update site. These import actions are not mutually exclusive. For example, an administrator might initially import an entire Eclipse update site and later choose to import specific additional or updated features. In the same manner, the administrator may later choose to import an additional NSF-based update site.

In order to use this feature in Composite Applications, you have to create your own update site database and import your Eclipse features. Then you can open the Composite Application Editor (refer to "Using the Composite Application Editor" on page 45 for more information) and point it against your NSF using the **Add Components > Add Components from Update Site** context menu item on the CAE's component palette. The dialog requires you to enter the URL of your update site. At this point you have to enter an HTTP or NRPC URL manually by using this syntax:

```
Domino server: http://updates.ibm.com/updatesite.nsf/site.xml
Domino server: nrpc://updates.ibm.com/__85257258006500E2/site.xml
Local replica: nrpc:///__85257258006500E2/site.xml
```

Editing a Composite Application's Feature Requirements Update Site URL

To use the CAE to edit the update site URL for a feature requirement, perform the following procedure:

1. In Lotus Notes, open the Composite Application and click **Actions > Edit Application...** to launch the application in the CAE.
2. Right-click the desired component and select **Edit Component Properties**.
3. Click the **Advanced...** button and select the desired feature (for example, **url.feature_NN**) and click **Edit**. Enter the update site URL.
4. Click **OK** twice to close all dialogs and close the CAE by clicking **Done**.

Creating an all-in-one NSF based Composite Application

It is possible to create Composite Applications which solely use components in the same databases. To do so, perform the following procedure:

1. Create an NSF-based update site as explained in "Creating an NSF Update Site" on page 51.
2. In Domino Designer, configure this application to launch as Composite Application.
3. Import all update sites of the used Eclipse components and add all NSF components
4. Open this application in the CAE and add the desired components from an internal update site.

In this case, it is not recommended to refer to the NSF components through a Notes URL that contains a replica ID or file path. Whenever the NSF is moved to another server, or whenever a new NSF based on an NTF is created, it is necessary to change the `CA.XML` with the new Notes URL. There is a special type of Notes URL: if `0000000000000000` is used in the Notes URL, Notes opens the NSF components from the same database it read the `CA.XML` from. Finally, deploy the application to a Domino server from where it can be provisioned as a single self-contained database.

White Lists

A new feature in Notes 8.0.1 provides a "white list", which consists of allowed update sites. When the white list option is used, the update sites specified in individual components can be ignored and the provisioning system inspects the sites listed in the white list only. The white list can be configured by the Domino server administrator[23].

Setting Access Rights for Composite Application Pages

Access rights for a page in a Composite Application can be set so that only the specified users can interact with that page in an application. For example, in a human resources application you may want a page that is only available to managers. Application users only see pages they can access listed in the application navigator. If there is an embedded way to access a page, for example a button in one component that opens a new page, an error message informs a user they do not have access to the page. To set access to a page in the CAE, right click the name of a page in the application navigator and select **Set page access**. Complete the dialog fields to allow or restrict access to the page. In Domino

23. http://www-10.lotus.com/ldd/compappwiki.nsf/dx/white-lists-and-updates

Designer, you can select a page design element in the design pane, then control the access for that page. The **Set page access...** context menu opens a dialog to let you change the access for that page.

Domino Designer also supports a new parameter on the Composite Application URL that specifies the retrieval of all pages regardless of access. This lets you edit the entire application, including pages that you would not see when using the application.

Calculate Links in NSF Components at Runtime

Links to NSF components (the Notes URLs - for more information, refer to "Building NSF components" on page 31 for more information) in a Composite Application typically change between development environment and production environment. Links may become invalid when an application deploys to a production environment. Here, the `ComputedNotesURL` preference can help. Additionally, it could be useful to have a dynamic Notes URL to customize an application at runtime based on the role or rights of the current user which is addressed by the `ProcessOnlyOnUpdate` preference. These two preferences can be set as advanced component properties as described in "Assembling Composite Applications and creating widgets" on page 43.

ComputedNotesURL ("com.ibm.notes.ComputedNotesURL")
This preference can be set to a value that is a macro of `@functions`, which resolves to a Notes URL. In the advanced component properties dialog, enter a formula to compute the Notes URL as the corresponding property value. For instance, if this preference is set with a value `@GetProfileField("URL Profile"\;"NotesURL")`, a field `NotesURL` is retrieved from a profile document `URL Profile`. Note that any semicolons used in the formula must have a preceding backslash ("\").

ProcessOnlyOnUpdate ("com.ibm.notes.ProcessOnlyOnUpdate")
This preference is set as a value of true or false. A value of `true` (default) would indicate that the `CA.XML` cache recomputes only when the `CA.XML` design note is updated. A value of `false` would indicate that the `CA.XML` cache recomputes each time it is requested. By default, the `CA XML` cache is only updated when the `CA.XML` design note is modified. It is worth noting that moving the Composite Application `CA.XML` to a new location (such as to deploy the Composite Application) results in a new `CA.XML` cache for the user. The preference "page level access" overrides the preference `ProcessOnlyOnUpdate`. For example, if "page level access" is turned on, the `CA.XML` is always returned even if the setting for `ProcessOnlyOnUpdate` is set to `true`.

Providing Extensions with Composite Applications

This approach creates an NSF-based update site which contains an "empty" Composite Application that solely serves for triggering the provisioning process of the feature to be deployed. Essentially, you must set the provisioning properties properly, especially the `url.feature` property to point to the `site.xml` file contained in the NSF-based update site and deploy it to a Domino server. End-users have to open this application only once to trigger provisioning for the desired extension.

Install Extensions as Notes Plug-ins

As Notes 8 is based on Eclipse, it is possible to install Eclipse plug-ins (such as sidebar panels) directly into any Notes client. You must provide the features and plug-ins as an Eclipse update site, either on a remote HTTP server or as local files in a folder or zipped archive (see "Getting started with Eclipse as a Notes developer" on page 27). Then, Notes must be enabled for "self-management", which allows for installing features manually. To do so, perform the following procedure:

1. 1. Edit *<notes_dir>*`/framework/rcp/plugin_customization.ini` and add this line to the file: `com.ibm.notes.branding/enable.update.ui=true`
2. Restart Notes.
3. Using the new menu item **File > Application > Install**, an update manager wizard allows for installing the new feature from the update site.

Updating Components with New Versions

IBM Lotus Expeditor introduced the manageability of Eclipse features and plug-ins by allowing the administrator or application assembler to identify what features need to be installed when a user opens a Composite Application. Each component in an application can specify what Eclipse-based features (if any) are needed for this component to be properly rendered in the client. Like the Portal managed side, the Lotus Notes side has the same capability using the Composite Application Editor.

When you bundle your plug-ins, they are contained within a feature. Each feature has a version associated with it. This is important to remember because the Eclipse provisioning system only installs new or compatible features and plug-ins. Generally, if you have a feature with a single plug-in or even several plug-ins you should keep the version of the feature and its plug-ins the same. This make problem resolution easier. If your feature happens to include other plug-ins, then the Eclipse provisioning system looks at it in layers. It first checks whether the version of the feature is already installed – if so, it will no longer examine the plug-ins. This means that if you deployed your feature once, then added a plug-in to it, you must increment the version number before deploying again, or the new plug-ins will not install on existing systems.

Composite applications handle downloading plug-ins and features in a way that is similar to Eclipse features. Composite applications only reference features in their descriptor (the `CA.XML` file). Just as with Eclipse feature deployments, if you change a plug-in in a Composite Application, you must change its version number and one of its containing features in order for it to be correctly deployed. Composite Applications also add another level of complexity, for performance reasons. The Composite Application Infrastructure (CAI) only looks at applications definitions (`CA.XML`) that have changed – the last time and date stamp of the file are preserved in the cache. As a result, if it does not appear your application has changed, the CAI does not open the file to install features. The change dependency tree looks like this:

```
---ca xml      - time and date change
---------feature    - version change
-------------plug-in  - version change
```

In order for a new plug-in to install, all three of these logic points must pass.

An Eclipse component added to a Composite Application through the Composite Application Editor contains information identifying the features that contain the Java files and other supporting files that allow the component to run. When a new version of a component is available, and you wish to have your application use the new version, you must update the information to reflect the new version of the component. Updating in the simplest case involves modifying the version number. In a more complex case you might need to add or remove features.[24]

Widgets Catalogs

The My Widgets feature installs with Notes 8.0.1. Using available policy and preference settings, an administrator can control which users may display My Widgets in their sidebar, as well as various levels of access to My Widgets functionality.

Creating a Widgets Catalog

Administrators can use the supplied catalog template to create a widgets catalog on a server. The catalog is an IBM Lotus Notes application. You must first obtain the "Widget Catalog (8)" template (`toolbox.ntf`), which installs with the IBM Lotus Domino 8.0.1 server. Create the catalog as a new Notes application using the template and assign appropriate ACLs to control access rights to the catalog application. After you have created the catalog, you can optionally create its initial set of categories. There are two types of predefined categories in the catalog - administrator categories and general categories. When you enable the "Toolbox Sweeper" agent, which is a scheduled agent set to run against new and modified documents, it ensures that Widget documents are properly created and populated; if a problem is found the offending document is removed from the user views, placed in the Administration/Document Queue, and an E-mail is sent to the document author informing him of the problem. A local copy of the catalog will automatically be created on the user's client system. Catalog replication is done based on low priority replication

Users who can display the My Widgets panel in their sidebar can click **Catalog > Replicate Catalog** and **Update Widgets** on the My Widgets option menu to initiate immediate update of their local catalog replica based on the settings in their **File > Preferences > Widgets** panel.

Catalog Access Rights

The administrator can assign catalog access rights based on user type, using a combination of standard Notes application Access Control Lists (ACLs) and policy or preference settings. The administrator can control a variety of My Widgets and catalog access settings using server based policies. If using My Widgets outside of a server-managed environment, there are equivalent settings in the Notes `plugin_customization.ini` file. Note that if a setting resides in both the Domino policy and the `plugin_customization.ini` file, the Domino policy value takes precedence. If a setting is changed using the Widgets preferences panel, that setting takes precedence for the duration of the active Notes session. For example, an administrator can control widget deployment based on categories. Categories are created in the catalog, but are administered by user policy or preference settings. Specific widgets can be deployed to specific users based on the category in which a given widget resides and the categories for which a given user is assigned.

There are preferences to set the host address of the catalog server, catalog name, the allowed widgets types, the right to publish widgets, and so on. For a complete list of all preferences, refer to the Domino Administrator Help.

24. http://www-10.lotus.com/ldd/compappwiki.nsf/dx/defining-feature-and-match-rules

Publishing to the Catalog and Controlling Access

You can publish widgets from the My Widgets sidebar panel to the catalog on the server. When you publish a widget, it exports the component and all its actions, including any custom content types and recognizers those actions are wired to. The published result is a widget IBM Lotus Notes document in the catalog application. The document is named according to the widget name and contains description information as well as an XML extension attachment representing the actual widget. When you publish to the catalog, you can specify a category for that widget so that it can be provisioned to users who are subscribed to that category. A widget can be part of more than one category. You can control user access to categories by using the IBM Lotus Domino policy settings available on the **Widgets** tab in the desktop policy settings document, or in the Notes `plugin_customization.ini` file. Only those widgets that did not originate from the catalog can be published to the catalog (for example, in your My Widgets sidebar panel, you can right-click on a widget that you have created and publish it to the catalog).

Managing Rich Clients using the Portal server

You can enable your Lotus Expeditor Client workstations to receive configuration information from a WebSphere Portal server. This allows you to centrally administer your Lotus Expeditor Client environment. Centrally administered role-based access control is a critical requirement of all large enterprises. WebSphere Portal provides a set of powerful functions specifically targeted at these requirements. The Managed Browser Administration Portlet exposes each instance of the browser as a portlet. Additionally, configurable elements of the browser instance (such as the address bar or URL restrictions) are also exposed through the portlet. By exposing these configurable elements, they can now be managed in a role-based fashion. Integrating with WebSphere Portal allows the overall user administration to remain as unified as possible, which reduces both financial and personnel cost.

Administrators can use the same page and place management, user management, and access controls to integrate, manage, and reuse application components running across a variety of device types, ranging from pervasive devices such as cell-phones and PDAs to browser-based desktops. The Portal-administered client extends this reach further to support “rich client” desktop devices based on Lotus Expeditor Client.

The term Portal-administered client simply means a side of the Lotus Expeditor Client that can install, load, and run applications defined by Portal. Portal applications are a specific kind of application, essentially a collection of pages, portlets, policies and roles. A WebSphere Portal application is much different than other applications, such as a rich client application in terms of Eclipse. The equivalent of a Portal application on the rich client is a set of navigable perspectives (pages) that contain views (portlets) and are configured with Eclipse preferences (policy) and can communicate through a property broker (wiring).

A Portal portlet maps to an Eclipse view, not a client portlet. This view can be any proper Eclipse view resident on the client. One of the views included with the client is a viewer to render HTML output from a real portlet running in the portlet container on the client. The client and the administration portlets treat this portlet viewer view in special ways, so it is easy to consider that a portlet maps to a portlet.

WebSphere Portal applications running on the client require extended client side properties that need to be specified on the pages and portlets on the server. To make the process of specifying these attributes easier, use the following portlets. These portlets expose Properties and Actions that map to the Eclipse counterpart on the client. The WAR file must then have an equivalent SWT plug-in that uses the same WSDL for the client-side property broker registration. This usage is basically SWT applications aggregated on, and deployed from, Portal.

Rich Client Layout Administration Portlet

This is an extension to the Portal page customizer that is used to manage common rich client-specific Metadata. It can inject this data into any portlet on any page as portlet preferences. A typical application with one or more pages may include instances of several types of portlets. The placeholder and administration proxies serve to establish position of portlets on pages that project as views and perspectives on the client.

JSR168 portlets also project as views but can, in addition, have client-side equivalents that run in the portlet container. Since JSR168 portlets do not have to be explicitly instrumented to manage rich client specific data, the same rich client unaware portlet can serve as an administration proxy for versions deployed at the rich client and be used as a ″normal″ portlet on the server for delivery through a browser. The Rich Client Layout Administration Portlet does not manage view-specific attributes, such as wiring properties/actions or view configuration. If this is needed for a given view, an appropriately programmed administration proxy will be required.

The Generic Placeholder portlet

This portlet aggregates any generic Eclipse view that does not expose properties and actions, or require view specific custom configuration. This portlet is little more than a container for the common layout preferences managed by the Rich Client Layout Administration Portlet (for example, the view ID, ratio, or feature requirements). This portlet is included in the web module with the Rich Client Layout Administration Portlet and can be copied and renamed as necessary by the administrator, then placed on pages to represent views.

″Administration proxy″ portlet

Unlike the generic placeholder portlet, an administration proxy is tightly coupled to the rich client view it represents. It may expose ″dummy″ properties and actions so that the portlet wiring tool can be used to create wires between administration proxies that are then sent in the Composite Application XML from the Portal server. This assumes the plug-in developer has implemented an action that can scope the wire to a specific view/secondary ID within the plug-in. This portlet may also provide a custom JSP that is used to manage view-specific configuration properties. Lotus Expeditor Client includes two administration proxies: the Managed Browser Administration portlet and the Rich Client WSRP Viewer Enablement Portlet. In general, an administration proxy must be created by the developer as part of the process of developing client-side plug-ins that support wiring and/or view configuration.

The Network Client Installer installs these portlets on the Portal server. For more information, refer to Installing with the Network Client Installer.

Connecting to Portal

Use the Home Portal Account Preference page to allow the Notes client to access applications and preferences on a Portal server. Access this page by navigating to **File > Preferences > Home Portal Account**. Enter the base URL for the server (for example, `http://wps6.example.com/portal`) in the Server field. Also enter the user ID and password that is used to access the portal server. The rest of the default settings are correct in most cases.

Using the Portal Catalog

You access the portal applications catalog by clicking **Open**, and then selecting Portal Applications from the list. If you do not see Portal Applications on the list, make sure that you have specified a home portal server by selecting **File > Preferences**, and then clicking **Home Portal Account**. To open a portal application listed in the catalog, double-click the name of the application.

User management of applications

End users can be manage their own applications through Eclipse Update sites hosted on HTTP servers. In order to use this feature, an administrator must enable user-initiated updates. For more details, see the Domino Administrator Help. Once enabled, users can install and remove applications using the Application Manager.

Using the Static Contributing extension point

This section provides detail on using the Stating Contributing extension point.

Packaging a Composite Application in an Eclipse plug-in.

In addition to deploying Composite Applications from Portal Server and NSF files, the XML which describes a Composite Application can be contributed using an extension point and deployed in a plug-in. The Composite Application XML is creating using the Composite Application Editor (CAE) in stand alone mode. The XML file is then added to one of the plug-ins that make up the application. The XML file is identified to the Notes client using the `com.ibm.rcp.portal.app.CompositeApplicationStaticContribution` extension point. For example:

```
<extension
        id="example"
        name="Static Projection of a CA"
        point="com.ibm.rcp.portal.app.CompositeApplicationStaticContribution">

 <staticfile path="CASample.xml"/>

</extension>
```

Expeditor Server

The Device Manager component of the Lotus Expeditor Server provides a common software management function for the Lotus Expeditor Client and Lotus Expeditor Client applications running on a variety of desktop and mobile devices. Device management functions provided include:

- **Device enrollment** - For registering the client.
- **Client and application configuration** - For setting client and application parameters.
- **Software distribution** - For distributing, installing and uninstalling software on the client. Includes the ability to see the status of requests.
- **Inventory** - For collecting hardware and software information about the device (available only on certain client platforms).

The Device Manager supports industry standards, such as the Open Mobile Alliance Device Management and the Open Service Gateway Initiative (OSGi). The Device Manager consists of a Web application and a relational database to store device management data. Managed clients require an agent to interact with the server. To manage an OSGi client, such as IBM Lotus Expeditor Client for Desktop and Devices, you need the plug-in and the device agent for OSGi devices. The OSGi Agent resides on the device and is responsible for running the commands sent by the Device Management Server. The OSGi Agent connects and interacts with the Device Management Server using the SyncML/DM protocol and is part of the Lotus Expeditor Client framework. The OSGi Agent does not have to be connected to the server at all times. When the agent connects, the server identifies the client and runs the jobs pending for the device.

Appendix A. Reference

Lotus Notes and Expeditor Feature Comparison Table

The following table lists the features that install by default in Lotus Notes and Lotus Expeditor. Some Lotus Expeditor features are not part of the Lotus Notes default install (identified as 'optional' below) and most collaboration features in Lotus Notes do not come with an Expeditor client license.

Developers who build software components that run on Lotus Notes 8.0.1 and require one of the 'optional' features listed below must install them onto Lotus Notes 8.0.1. For information on this installation process, refer to the developerWorks article Installing additional features from IBM Lotus Expeditor 6.1.2 onto IBM Lotus Notes 8.0.1.

Table 3. Lotus Notes and Lotus Expeditor Default Features

Feature	Expeditor	Notes
com.ibm.db2e.feature	X	X (optional)
com.ibm.esupport.client.product.SSC4TNF.feature	X	X
com.ibm.esupport.client.product.SSC4TNF61.feature	X	
com.ibm.eswe.preference.feature	X	X
com.ibm.langware.engine.feature	X	X
com.ibm.langware.v27.dic.feature	X	X
com.ibm.langware.v27.feature	X	X
com.ibm.langware.v5.dic.feature	X	X
com.ibm.langware.v5.feature	X	X
com.ibm.logging.icl.feature	X	X
com.ibm.micro.admin.feature	X	X (optional)
com.ibm.micro.feature	X	(optional)
com.ibm.mobileservices.isync.db2j.feature	X	(optional)
com.ibm.mobileservices.isync.feature	X	X (optional)
com.ibm.mqe.feature	X	X (optional)
com.ibm.mqe.jms.feature	X	X (optional)
com.ibm.mqttclient.feature	X	(optional)
com.ibm.mqttclient.jms.feature	X	(optional)
com.ibm.osg.service.cm.feature	X	X
com.ibm.osg.webapp.feature		X
com.ibm.osg.service.metatype.feature	X	
com.ibm.portal.cai.feature	X	
com.ibm.osg.service.useradmin.feature	X	X (optional)
com.ibm.portal.feature	X	X
com.ibm.pvc.ejb.feature	X	
com.ibm.pvc.jms.feature	X	X (optional)
com.ibm.pvc.jndi.provider.java.feature	X	X

Table 3. Lotus Notes and Lotus Expeditor Default Features (continued)

Feature	Expeditor	Notes
com.ibm.pvc.jta.feature	X	(optional)
com.ibm.pvc.runtime.doc.user.feature	X	
com.ibm.pvc.servlet.feature	X	X
com.ibm.pvc.servlet.jsp.feature	X	X
com.ibm.pvc.servlet.jsp.jstl.feature	X	X
com.ibm.pvc.wct.osgiagent.ext.core.feature	X	X
com.ibm.pvc.wct.osgiagent.ext.native.feature_2.1.1.0-200804210300	X	X
com.ibm.pvc.wct.osgiagent.feature	X	X
com.ibm.pvc.wct.workbench.feature	X	X
com.ibm.pvc.webcontainer.feature	X	X
com.ibm.pvc.webhttpservice.feature	X	X
com.ibm.pvcws.feature	X	X (optional)
com.ibm.pvcws.osgi.feature	X	X (optional)
com.ibm.pvcws.wsrf.feature		(optional)
com.ibm.pvcws.wss.feature	X	(optional)
com.ibm.rcp.accounts.feature	X	X
com.ibm.rcp.accounts.ui.feature	X	X
com.ibm.rcp.browser.service.feature	X	X
com.ibm.rcp.core.logger.monitor.feature	X	
com.ibm.rcp.core.logger.ui.feature	X	X
com.ibm.rcp.database.derby.feature	X	X (optional)
com.ibm.rcp.database.feature	X	(optional)
com.ibm.rcp.dombrowser.feature	X	X
com.ibm.rcp.eclipse.emf.feature	X	X
com.ibm.rcp.eclipse.gef.feature	X	X
com.ibm.rcp.eclipse.jdt.feature	X	X
com.ibm.rcp.eclipse.platform.feature	X	X
com.ibm.rcp.eclipse.platform.ui.feature	X	X
com.ibm.rcp.eclipse.rcp.ui.feature	X	X
com.ibm.rcp.eclipse.xsd.feature	X	X
com.ibm.rcp.esupport.client.doc.feature	X	X
com.ibm.rcp.esupport.client.feature	X	X
com.ibm.rcp.jvm.feature	X	X
com.ibm.rcp.lapinvoker.feature	X	
com.ibm.rcp.locationmanager.feature	X	X
com.ibm.rcp.locationmanager.ui.feature	X	X
com.ibm.rcp.managedsettings.feature	X	X
com.ibm.rcp.managedsettings.portal.feature	X	X
com.ibm.rcp.net.feature	X	X

Table 3. Lotus Notes and Lotus Expeditor Default Features (continued)

Feature	Expeditor	Notes
com.ibm.rcp.os.feature	X	X
com.ibm.rcp.os.ui.feature	X	X
com.ibm.rcp.osgiagent.cit.installhandler.feature	X	
com.ibm.rcp.personality.default.branding.feature	X	
com.ibm.rcp.personality.default.feature	X	X
com.ibm.rcp.personality.framework.feature	X	X
com.ibm.rcp.platform.feature	X	X
com.ibm.rcp.platform.management.feature	X	X
com.ibm.rcp.portal.app.ui.feature	X	X
com.ibm.rcp.portal.catalog.ui.feature	X	X
com.ibm.rcp.portal.feature	X	X
com.ibm.rcp.portletcontainer.feature	X	X
com.ibm.rcp.portletviewer.feature	X	X
com.ibm.rcp.propertybroker.feature	X	X
com.ibm.rcp.propertybroker.swt.feature	X	X
com.ibm.rcp.provisioning.feature	X	X
com.ibm.rcp.provisioning.ui.feature	X	X
com.ibm.rcp.rte.feature	X	X
com.ibm.rcp.security.feature	X	
com.ibm.rcp.servlet.jsf.ext.feature	X	X (optional)
com.ibm.rcp.servlet.jwl.feature		X (optional)
com.ibm.rcp.sharedbundle.ui.feature	X	X
com.ibm.rcp.sync.feature	X	X
com.ibm.rcp.syncui.feature	X	X
com.ibm.rcp.syncui.scheduler.feature	X	X
com.ibm.rcp.textanalyzer.feature	X	X
com.ibm.rcp.topologyhandler.ui.feature	X	X
com.ibm.rcp.ui.browser.feature	X	X
com.ibm.rcp.ui.browser.launcher.feature	X	X
com.ibm.rcp.ui.browser.portal.feature	X	X
com.ibm.rcp.ui.widgets.feature	X	X
com.ibm.rcp.webcontainer.jspcompilerbridge.feature	X	X
com.ibm.rcp.ws.axis.feature	X	X
com.ibm.rcp.wsrp.feature	X	X
com.ibm.rcp.xulrunner.runtime.feature	X	
com.ibm.syncml4j.feature	X	X
org.apache.axis.feature	X	X
org.apache.commons.feature	X	X
org.apache.derby.feature	X	X
org.apache.myfaces.feature	X	X

Table 3. Lotus Notes and Lotus Expeditor Default Features (continued)

Feature	Expeditor	Notes
org.apache.portals.bridges.feature		X (optional)
org.osgi.service.webapplication.feature	X	X
Collaboration Features		
com.ibm.collaboration.realtime.activitymonitor.feature	X	X
com.ibm.collaboration.realtime.annotate.feature	X	X
com.ibm.collaboration.realtime.app.services.feature	X	X
com.ibm.collaboration.realtime.application.feature	X	X
com.ibm.collaboration.realtime.bluepages.feature	X	X
com.ibm.collaboration.realtime.browser.feature	X	X
com.ibm.collaboration.realtime.calendar.common.feature	X	X
com.ibm.collaboration.realtime.calendar.feature	X	X
com.ibm.collaboration.realtime.chat.feature	X	X
com.ibm.collaboration.realtime.chat.logging.feature	X	X
com.ibm.collaboration.realtime.chat.logging.impl.feature	X	X
com.ibm.collaboration.realtime.chat.logging.ui.feature	X	X
com.ibm.collaboration.realtime.community.st.feature	X	X
com.ibm.collaboration.realtime.contactlist.feature	X	X
com.ibm.collaboration.realtime.core.feature	X	X
com.ibm.collaboration.realtime.credentialstore.feature	X	X
com.ibm.collaboration.realtime.embedded.feature	X	X
com.ibm.collaboration.realtime.filetransfer.feature	X	X
com.ibm.collaboration.realtime.help.feature	X	X
com.ibm.collaboration.realtime.im.community.feature	X	X
com.ibm.collaboration.realtime.im.community.impl.feature	X	X
com.ibm.collaboration.realtime.location.feature	X	X
com.ibm.collaboration.realtime.login.feature	X	X
com.ibm.collaboration.realtime.meetings.feature	X	X
com.ibm.collaboration.realtime.messages.feature	X	X
com.ibm.collaboration.realtime.notes.connector.feature	X	X
com.ibm.collaboration.realtime.notes.messages.feature	X	X
com.ibm.collaboration.realtime.notifications.feature	X	X
com.ibm.collaboration.realtime.palettes.feature	X	X
com.ibm.collaboration.realtime.people.feature	X	X
com.ibm.collaboration.realtime.people.impl.feature	X	X
com.ibm.collaboration.realtime.quickfind.feature	X	X
com.ibm.collaboration.realtime.rtc.core.feature	X	X
com.ibm.collaboration.realtime.rtc.st.feature	X	X
com.ibm.collaboration.realtime.rtcadapter.feature	X	X
com.ibm.collaboration.realtime.spellchecker.feature	X	X
com.ibm.collaboration.realtime.stjavatk.feature	X	X

Table 3. Lotus Notes and Lotus Expeditor Default Features (continued)

Feature	Expeditor	Notes
com.ibm.collaboration.realtime.telephony.base.feature	X	X
com.ibm.collaboration.realtime.telephony.prod.feature	X	X
com.ibm.collaboration.realtime.ui.feature	X	X
com.ibm.content.operations.registry.feature		X
com.ibm.content.operations.registry.infra.feature		X
com.ibm.csi.feature		X
com.ibm.csi.notes.feature		X
com.ibm.directoryservices.feature		X
com.ibm.notes.aaf.feature		X
com.ibm.notes.branding.feature		X
com.ibm.notes.client.feature		X
com.ibm.notes.client.win32.feature		X
com.ibm.notes.dip.feature		X
com.ibm.notes.esupport.client.product.SSKTWPR8.feature		X
com.ibm.notes.gettingstarted.feature		X
com.ibm.notes.ipc.feature		X
com.ibm.notes.java.api.feature		X
com.ibm.notes.links.feature		X
com.ibm.notes.pd.feature		X
com.ibm.notes.pim.feature		X
com.ibm.notes.sametime.installed.feature		X
com.ibm.notes.search.common.feature		X
com.ibm.notes.search.googledesktop.feature		X
com.ibm.notes.toolbox.ca.adapter.feature		X
com.ibm.notes.toolbox.feature		X
com.ibm.notes.toolbox.langware.feature		X
com.ibm.productivity.tools.base.win32.feature		X
com.ibm.productivity.tools.feature		X
com.ibm.productivity.tools.gallery.feature		X
com.ibm.productivity.tools.notes.branding.feature		X
com.ibm.productivity.tools.serviceability.feature		X
com.ibm.productivity.tools.share.feature		X
com.ibm.productivity.tools.standalone.feature		X
com.ibm.productivity.tools.template.feature		X
com.ibm.rcp.aaf.feature	X	X
com.ibm.rcp.contentspots.feature		X
com.ibm.rcp.feedreader.feature		X
com.ibm.rcp.feedreader.notes.feature		X
com.ibm.rcp.feedreader.notes.providers.feature		X
com.ibm.rcp.feeds.parser.feature		X

Table 3. Lotus Notes and Lotus Expeditor Default Features (continued)

Feature	**Expeditor**	**Notes**
com.ibm.rcp.g11n.feature		X
com.ibm.rcp.notes.util.feature		X
com.ibm.rcp.pim.typeahead.feature		X
com.ibm.rcp.pim.views.feature		X
com.ibm.rcp.pim.widgets.feature		X
com.ibm.rcp.realtime.livenames.feature	X	X
com.ibm.rcp.search.engines.google.feature		X
com.ibm.rcp.search.engines.yahoo.feature		X
com.ibm.rcp.search.feature		X
com.ibm.rcp.threading.monitor.feature		X
com.ibm.rcp.ui.browser.content.feature		X
com.ibm.siapi.feature		X

Appendix B. External resources and links

This Appendix contains a comprehensive list of URLs and references provided in each section of the document.

Business value of Notes 8.x application development

1. OpenNTF.org: http://www.openntf.org/Projects/pmt.nsf/852fcfa76eb36baa85256bae00100855/998edc82d1fb2f3686257399005fe8e0!OpenDocument
2. Business Intelligence and Reporting Tools: http://www.eclipse.org/birt/phoenix/
3. Sales lead manager sample: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/ibm-lead-manager
4. Sales lead manager sample articles: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/lead-manager-sample-composite-application-articles

Roles and tools

Composite Applications wiki: http://www-10.lotus.com/ldd/compappwiki.nsf

Getting started with Eclipse as a Notes developer

1. Eclipse: http://www.eclipse.org
2. Eclipse RCP: http://wiki.eclipse.org/index.php/Rich_Client_Platform
3. Eclipse Newcomers FAQ: http://www.eclipse.org/home/newcomers.php
4. Equinox: http://www.eclipse.org/equinox/
5. P. Carlson, 21 Aug 2007, IBM, Getting started with the IBM Lotus Expeditor Toolkit 6.1; available online at: http://www.ibm.com/developerworks/lotus/library/expeditor-toolkit/
6. OSGi Home page: at http://www.osgi.org/Main/HomePage
7. E. Clayberg, D. Rubel, 2004, Pearson Education Inc., Eclipse, Building Commercial-Quality Plug-ins, page 129
8. E. Clayberg, D. Rubel, 2004, Pearson Education Inc., Eclipse, Building Commercial-Quality Plug-ins, page 129
9. E. Clayberg, D. Rubel, 2004, Pearson Education Inc., Eclipse, Building Commercial-Quality Plug-ins, page 130
10. B. Marchal, 3 Feb. 2005, Working XML: Define and load extension points; available online at: http://www.ibm.com/developerworks/xml/library/x-wxxm29.html
11. E. Clayberg, D. Rubel, 2004, Pearson Education Inc., Eclipse, Building Commercial-Quality Plug-ins, page 653
12. Eclipse wiki, FAQ What is a plug-in fragment?: http://wiki.eclipse.org/FAQ_What_is_a_plug-in_fragment%3F
13. D. Glozic, D. Birsan, 27 September 2003, IBM, How To Keep Up To Date; available online at: http://www.eclipse.org/articles/Article-Update/keeping-up-to-date.html

14. Composite Application Wiki, IBM, Introduction to provisioning and updating: http://www.ibm.com/developerworks/wikis/display/appdev/Introduction+to+provisioning+and+updating
15. J. McAffer, J. Lemieux, 2005, Addison-Wesley, eclipse – Rich Client Platform, page 215
16. Personalities, IBM Lotus Expeditor InfoCenter: http://publib.boulder.ibm.com/infocenter/ledoc/v6r1/index.jsp?topic=/com.ibm.rcp.tools.doc.appdev/ui_usingpersonalities.html
17. M. Valenta, 20 September 2004, IBM, On the Job: The Eclipse Jobs API; available online at: http://www.eclipse.org/articles/Article-Concurrency/jobs-api.html
18. J. McAffer, J. Lemieux, 2005, Addison-Wesley, eclipse – Rich Client Platform, page 357
19. Eclipse.org, Extension Registry API's: http://help.eclipse.org/help32/topic/org.eclipse.platform.doc.isv/reference/api/org/eclipse/core/runtime/IExtensionRegistry.html

Building NSF components

1. Sales lead manager sample: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/ibm-lead-manager
2. Hide view navigator: http://www.ibm.com/developerworks/wikis/display/appdev/Manage+Real+Estate+for+NSF+Components+in+Composite+Applications
3. Special replica IDs: http://www.ibm.com/developerworks/wikis/display/appdev/Deployment+of+NSF+based+Composite+Applications+with+Components+in+the+same+NSF
4. Deployment of NSF components: http://www.ibm.com/developerworks/wikis/display/appdev/How+to+calculate+Links+to+NSF+Components+at+Runtime
5. Built in properties and actions: http://www.ibm.com/developerworks/wikis/display/appdev/Using+built-in+properties+and+actions+in+Notes+8.0.1
6. Simple tutorial: http://www.ibm.com/developerworks/lotus/library/tutorials/notes8-comp-apps/lz-dw-lz-notes8-comp-apps.html?S_TACT=105AGX13&S_CMP=LP
7. Usage of the property broker editor: http://www.ibm.com/developerworks/wikis/display/appdev/Property+Broker+Editor+for+NSF+Components
8. Access of static component preferences: http://www.ibm.com/developerworks/wikis/display/appdev/How+to+run+NSF+Components+in+Context
9. CAI URL: http://www.ibm.com/developerworks/wikis/display/appdev/Pass+Context+to+Components+when+opening+Composite+Applications
10. Open Notes documents on pages: http://www.ibm.com/developerworks/wikis/display/appdev/Open+Notes+Documents+on+Pages

Building Java components

1. Using Lotus Expeditor Toolkit with Notes 8 and Lotus Sametime: http://www.ibm.com/developerworks/lotus/library/expeditor-notes-sametime/
2. Developing Applications for Lotus Expeditor, 6.1.x: http://publib.boulder.ibm.com/infocenter/ledoc/v6r11/index.jsp

3. Notes.jar: http://www-128.ibm.com/developerworks/lotus/library/notes8-data/
4. Eclipse: www.eclipse.org
5. Lotus Notes 8 Sidebar/Toolbar: http://www-128.ibm.com/developerworks/lotus/library/notes8-context/
6. Drag and Drop: http://www.eclipse.org/articles/Article-SWT-DND/DND-in-SWT.html
7. Developing an Eclipse Component, part 1: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/developing-an-eclipse-component-for-lotus-notes-8-part-1
8. Developing an Eclipse Component, part 2: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/developing-an-eclipse-component-for-lotus-notes-8-part-2
9. Lotus Expeditor Run/Debug: http://www.ibm.com/developerworks/lotus/library/expeditor-notes-sametime/
10. Debugging with Eclipse: http://www.ibm.com/developerworks/java/library/os-ecbug/

Assembling Composite Applications and creating widgets

1. PIM component example: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/using-pim-components
2. Embedded Browser component example: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/using-the-embedded-browser-component
3. Symphony component example: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/using-the-symphony-view-component
4. Widgets: http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=/com.ibm.help.domino80x.doc/DOC/H_MY_WIDGETS_OVER.html
5. Page properties: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/advanced-page-properties
6. Component properties: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/advanced-component-properties

Deploying applications

1. White lists: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/white-lists-and-updates
2. Feature and match rules: http://www-10.lotus.com/ldd/compappwiki.nsf/dx/defining-feature-and-match-rules
3. My Widgets preferences: http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=/com.ibm.help.domino80x.doc/DOC/H_SETTING_CATALOG_PREFERENCES_USING_A_CUSTOMIZATION_FILE_OVER.html
4. Network Client Installer: http://publib.boulder.ibm.com/infocenter/ledoc/v6r11/index.jsp?topic=/com.ibm.rcp.tools.doc.admin/serverbasedclientinstallation.html
5. User-initiated updates: http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=/com.ibm.help.domino80x.doc/DOC/H_ENABLING_USER_INITIATED_UPDATE_FOR_NOTES_8_STANDARD_3064_OVER.html

Appendix C. Notices

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

IBM might have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

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

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

Trademarks

Printed in USA