Oracle® Application Server

Developer's Guide for Microsoft Office Interoperability

10*g* Release 3 (10.1.3)

B25781-01

March 2006

ORACLE®

Oracle Application Server Developer's Guide for Microsoft Office Interoperability, 10*g* Release 3 (10.1.3)

B25781-01

Primary Authors: Peter Lubbers, Susan Highmoor

Contributing Authors: Ingrid Snedecor, Lalithashree Rajesh, Pravin Prabhakar

Contributors: Abhinav Agarwal, Albert Tam, Andy Page, Beth Morgan, Bhagat Nainani, Carolyn Bruse, Celia Coakley, Christine Jacobs, Chuck Murray, Dan Hynes, Deanna Bradshaw, Don Gosselin, Dawn Tyler, Frank Knifsend, Frank Rovitto, Guus Ramackers, Harpal Kochar, Jacques Vigeant, James Owen, Joe Garcia, Jon Maron, Luke Kowalski, Mahasweta Dey, Marc Houle, Marcie Caccamo, Mark Kennedy, Marty Roth, Michael McGrath, Michele Cyran, Navneet Singh, Olaf Stullich, Orlando Cordero, Parsha Reddy, Philipp Weckerle, Promila Chitkara, Raj Gupta, Rajesh Ramachandran, Raji Mahalingam, Ranga Polisetti, Ravi Rangaswamy, Robin Clark, Rohit Marwaha, Saheli Dey, Susan Shephard, Tal Broda, Thomas Kurian, Tim Dexter, Tom Pfaeffle, Vasuki Ashok, Vinayak Hegde, Virginia Beecher

Contents

3 Understanding Oracle Application Server Interoperability with Microsoft Office

Part II Building Microsoft Office Interoperability Solutions

4 Creating Smart Documents That Interact with Self-Service Business Processes

5 Completing Forms and Entering Data Using Microsoft Office

List of Examples

List of Figures

List of Tables

Preface

This guide describes how to enable interoperability between the Microsoft Office suite of products and the Oracle Application Server set of components. This includes a description of the Microsoft Office-centric architecture, the Microsoft Office Extensibility technologies, the Oracle Application Server components that can interoperate, and many step-by-step procedures that describe how to build Microsoft Office interoperability solutions.

> **Note:** For the portable document format (PDF) version of this guide, when a URL breaks onto two lines, the full URL data is not sent to the browser when you click it. To get to the correct target of any URL included in the PDF, copy and paste the URL into your browser's address field. In the HTML version of this guide, you can click a link to directly display its target in your browser.

Audience

This guide is intended for Oracle Application Server developers and administrators who want to configure Oracle Application Server components to interoperate with Microsoft Office, and are familiar with Microsoft Office applications and Microsoft programming languages.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

You can find all documentation related to Oracle Application Server, including the release notes, on the Oracle Application Server documentation page of the Oracle Technology Network (OTN):
`http://www.oracle.com/technology/documentation/appserver.html`

For additional information and to post queries about Oracle Application Server and Microsoft Office interoperability, access the Microsoft Office Interoperability discussion forum on OTN:

`http://forums.oracle.com/forums/forum.jspa?forumID=266`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
CAPITALIZED	Capitalized text indicates procedure names.
< >	Angle brackets enclose user-supplied information.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

Accessing the Demonstration Support Files

The demonstration support files that are described in various chapters of this guide are available on the ORACLE FUSION MIDDLEWARE and Microsoft Interoperability page on Oracle Technology Network (OTN) at `http://www.oracle.com/technology/products/middleware/fusion-middleware-microsoft-interoperability.html`. The ZIP file `microsoft-interoperability-guide-demo-support.zip` (found in the Developer's Guide section) contains folders with support files for specific chapters in

this guide. These support files are necessary to run the example procedures. The individual chapters' prerequisites sections will have additional details about what must be done with the support files. The following table maps example folders to the chapters in which they are used.

Folder Name	Chapter Title
selfservice	Chapter 4, "Creating Smart Documents That Interact with Self-Service Business Processes"
fillingforms	Chapter 5, "Completing Forms and Entering Data Using Microsoft Office"
securingsmartdocs	Chapter 6, "Securing Smart Documents and Web Services"
identitymanagement	Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts"
smarttags	Chapter 11, "Accessing in-Context Web Information and Invoking an Enterprise Portal"
reports	Chapter 13, "Delivering Enterprise Reports to Microsoft Office with Oracle Reports"

See Also: Appendix A, "Code Examples" for details about code samples and template content that you may need to use in the sample demonstrations that are described in various chapters of this guide.

Part I

Overview

Part I describes the Microsoft Office-centric architecture, the Microsoft Office Extensibility technologies available in Microsoft Office 2003, and the Oracle Application Server components that can interoperate with Microsoft Office. It contains the following chapters:

- Chapter 1, "Microsoft Office in the Enterprise Architecture"
- Chapter 2, "Understanding Microsoft Office 2003 Extensibility Technologies"
- Chapter 3, "Understanding Oracle Application Server Interoperability with Microsoft Office"

1

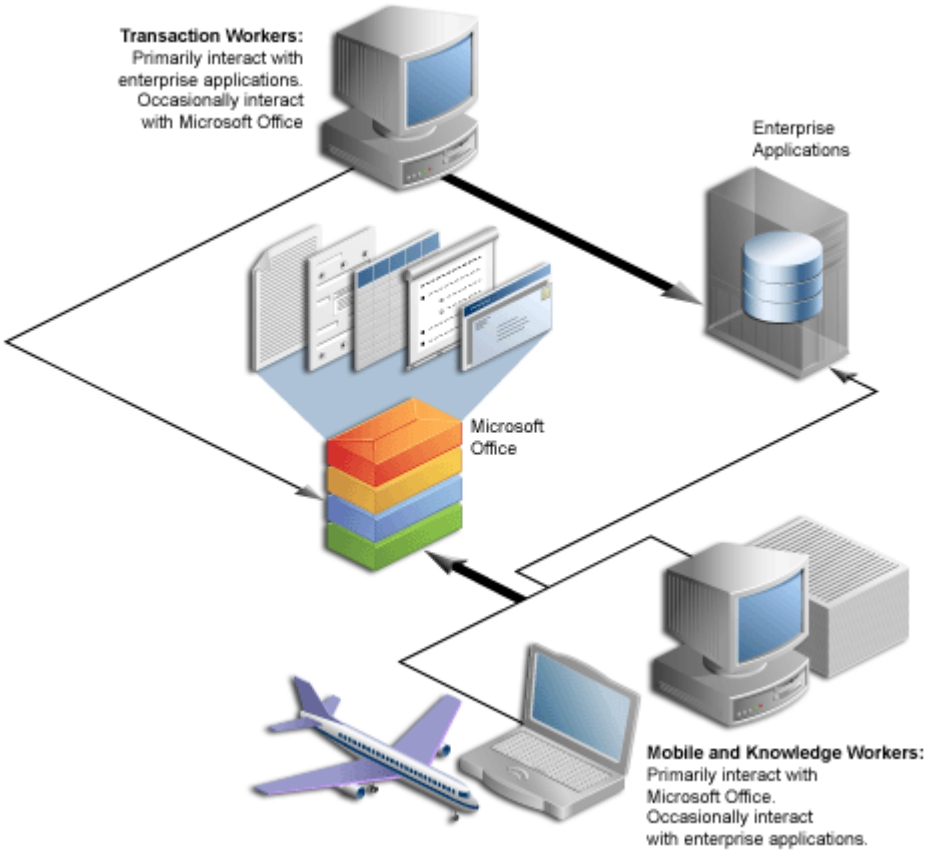
Microsoft Office in the Enterprise Architecture

This chapter contains the following topics:

1.1 Understanding the Need for Using Microsoft Office in an Enterprise Architecture

Microsoft Office is a product that is widely used, especially by knowledge workers, but it has not always been effectively used in conjunction with enterprise applications. Users of Microsoft Office and enterprise applications can often be thought of in two categories: transaction workers and knowledge workers. Transaction workers use one or more fixed transaction applications for work, for example, help desk, customer service, order entry, or inventory. Knowledge workers are involved in tasks like searching, organizing, storing, and analyzing information and then using the knowledge gained appropriately. Both have a need to work with Microsoft Office and enterprise applications, but their interactions vary as shown in Figure 1–1.

Figure 1–1 Microsoft Office Documents Accessed by Different Types of Users



Transaction workers work almost entirely in an enterprise application environment and use Microsoft Office sparingly. However, they frequently must get information created in the Microsoft Office environment into enterprise applications and send information, forms, reports, and so on to other people who prefer, or want, the information in a Microsoft Office format.

Knowledge workers on the other hand, spend most of their time using Microsoft Office and use enterprise applications sparingly, or may even be mobile users who do not always have a connection to an enterprise application. For the knowledge workers, it may be much more familiar and comfortable to do most of their work in the context of the Microsoft Office environment rather than having to copy or repeat work in the enterprise application.

Given the different needs of these different users, it would be better if Microsoft Office and enterprise applications could be better integrated, making the overall experience more seamless. For example:

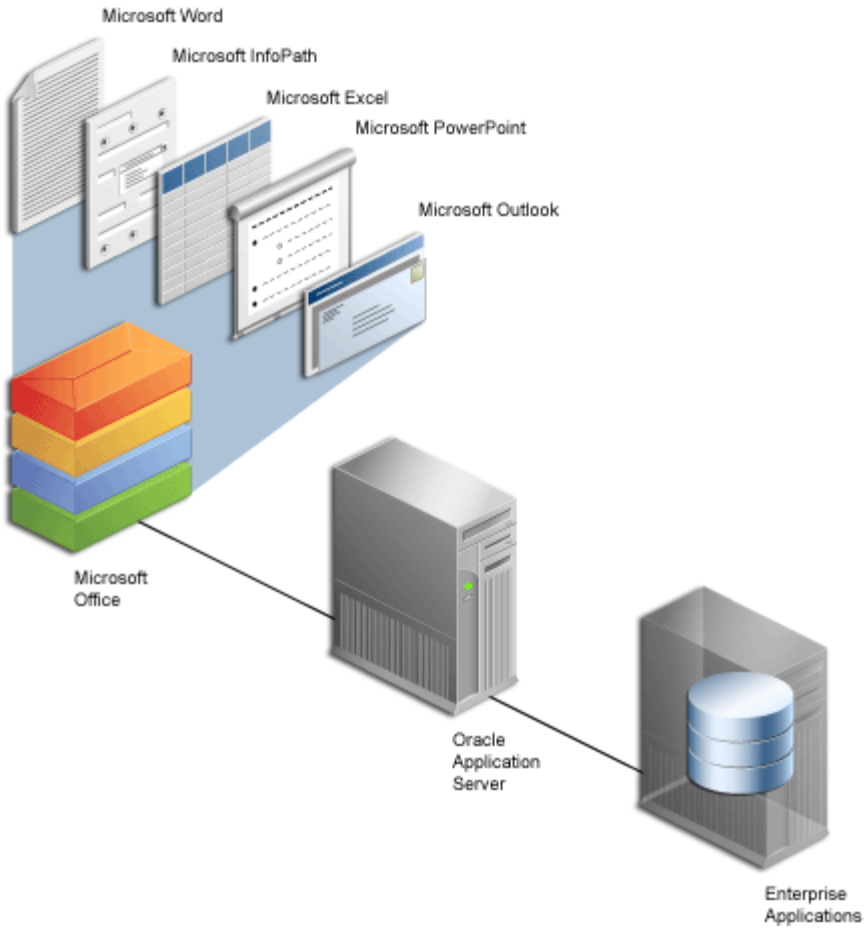
- Transaction workers do not need to learn the more advanced features of Microsoft Office and can continue to work in the environment with which *they* are familiar, yet still providing the preferred documents to their audience at the same time.
- Knowledge workers can continue to use the Microsoft Office environment with which they are familiar, without having to spend time learning a new, and potentially complex, enterprise application that is not designed to meet their specific working methods.
- Mobile workers can work offline in Microsoft Office when necessary and then update the enterprise application later when they are able to connect.

1.2 Meeting the Need for Using Microsoft Office in an Enterprise Environment

Oracle Application Server provides the opportunity to improve productivity by presenting the right information, at the right time, in a familiar tool.

Oracle Application Server enables developers to build enterprise applications that automate transaction processing, streamline business processes, and access and deliver information within the enterprise as shown in Figure 1–2.

Figure 1–2 Oracle Application Server Interoperability with Microsoft Office



Enabling interoperability between Microsoft Office suite of products and the Oracle Application Server set of components provides the following benefits:

- Simplifying and Improving the User Experience
- Streamlining and Automating Enterprise Business Processes
- Synchronizing User Information with Enterprise Information

1.2.1 Simplifying and Improving the User Experience

It is important to simplify and improve the user experience associated with enterprise applications interaction. Specifically, today most enterprise applications have Web-based user interfaces, but knowledge workers often find that these user

interaction models are less intuitive to use and interfere with their daily work in Microsoft Office.

Using Oracle Application Server, developers can create Microsoft Office solutions that manipulate enterprise information. This combines the familiar Microsoft Office interface with the power of the underlying enterprise application. For more information, refer to the following chapters in this guide:

- Chapter 4, "Creating Smart Documents That Interact with Self-Service Business Processes"
- Chapter 5, "Completing Forms and Entering Data Using Microsoft Office"

1.2.2 Streamlining and Automating Enterprise Business Processes

To streamline and automate enterprise business processes, there are two requirements:

- When tasks within the context of enterprise business processes require human participation, users would like this participation to occur from within the Microsoft Office suite of applications. For example: handling workflow alerts and notifications received in e-mail.

  Refer to Chapter 7, "Delivering Business Activity Monitoring Alerts and Reports to Microsoft Outlook" for information about how Oracle Application Server enables this.

- When tasks within the context of enterprise business processes require the use of Microsoft Office documents, users would like the ability to direct enterprise business processes from the Microsoft Office environment itself. For example: managing a travel request or an absence template form in Microsoft Word or Microsoft Excel.

  Oracle Application Server provides this ability. Refer to Chapter 4, "Creating Smart Documents That Interact with Self-Service Business Processes" to find out how.

1.2.3 Synchronizing User Information with Enterprise Information

It is critical that identity information is always up-to-date so that all enterprise users and applications are synchronized. In many enterprise environments, however, a number of systems have to be configured to make this work. Additionally, developers can use Oracle Application Server components to configure their systems to send notifications about changes in identity information directly to Microsoft Outlook.

Refer to Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts" for information about how this can be done.

1.3 Getting Started with Microsoft Office

Prerequisites for using Microsoft Office for the extensibility and interoperability tasks discussed in this guide are as follows:

- Microsoft Office. The version of Microsoft Office required may change from chapter to chapter.

  In this guide, the focus is primarily on Microsoft Office version 2003. This is to leverage different technologies that Microsoft Office 2003 offers for extensibility and interoperability. Refer to Chapter 2, "Understanding Microsoft Office 2003 Extensibility Technologies" for more information.

- Information about how to configure Microsoft Office and Microsoft Exchange.

The general Microsoft Office setup is straight-forward, because it is a simple desktop installation and not a complex distributed enterprise deployment.

Notes:

- In addition to the requirements in this section, each chapter has its own prerequisites.
- For some cases, an earlier version of Microsoft Office will work, and this will be specifically mentioned in the Prerequisites section of the relevant chapters.
- For some procedures, the .NET framework is required. It is important that the .NET framework be installed *before* installing Microsoft Office.

1.3.1 Microsoft Office Versions and Editions

As mentioned earlier, this guide primarily focuses on the extensibility technologies supported in Microsoft Office 2003.

Microsoft Office is available in various editions such as, Office Professional Edition, Office Small Business Edition, Office Student and Teacher Edition, and Office Standard Edition. All these editions include the following applications:

- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint
- Microsoft Outlook

Solutions that use the extensibility technologies discussed in this guide require Microsoft Office Professional Edition 2003 at run time in most cases. For example, a few applications, like Microsoft Access and Microsoft InfoPath, are available only in Microsoft Office Professional Edition 2003.

1.3.2 Microsoft Office Components

The following Microsoft Office suite of products can interoperate with the Oracle Application Server set of components:

- Microsoft Word

  Microsoft Word supports *smart documents*, which help in building interactive documents that can access disparate data sources and communicate with Web services. This minimizes the need to switch between browsers and applications.

- Microsoft Excel

  Output from various Oracle Application Server reporting tools can be saved in Microsoft Excel format.

- Microsoft PowerPoint

  Microsoft PowerPoint supports the use of *smart tags*, which can start enterprise applications from Microsoft PowerPoint presentations. It also provides support for Oracle Real-Time Collaboration Add-in for Microsoft Office that supports chatting with other Oracle Messenger users and starting instant Web conferences from within the Microsoft PowerPoint application.

- Microsoft Outlook

  Some of the extensibility and interoperability options possible with Microsoft Outlook include support for Oracle Real-Time Collaboration Add-in for Microsoft Office, and provisioning of user identity information in Microsoft Outlook from Oracle Internet Directory.

- Microsoft Infopath

  Web services built in Oracle JDeveloper can be invoked in Microsoft InfoPath.

1.3.3 Microsoft Software Development Kits, Utilities, and References

To perform the tasks discussed in this guide, the following Microsoft Software Development Kits, toolkits, and utilities may need to be installed:

- Microsoft Office 2003 .NET Framework Software Development Kit
- Microsoft Office 2003 VBA Language References
- Microsoft Office 2003 XML Reference Schemas
- Web Services Enhancements
- Microsoft Office 2003 Web Services Toolkit
- Microsoft PKI Utilities
- Office 2003 Update: Redistributable Primary Interop Assemblies
- Microsoft Office Schema Tag Lists
- Microsoft Office WordprocessingML Transform Inference Tool

Downloading the Software

Perform the following tasks to download the required software from the Microsoft Web site:

1. Access the Microsoft downloads page at

   `http://www.microsoft.com/downloads/`

2. In the search criteria, enter the name of the Software Development Kit, toolkit, utility, or reference document, and click **GO**.
3. Select an appropriate version of the software.
4. Follow the instructions on the page, and click **Download**.

Microsoft Office 2003 .NET Framework Software Development Kit

The .NET Framework programming model simplifies development and deployment and allows for integration with different programming languages.

To download this, perform the steps mentioned in "Downloading the Software" by entering **.NET Framework Software Development Kit** as the search criteria on the Microsoft downloads page.

Note: You must install the .NET Framework before you install Microsoft Office. If you have installed Microsoft Office first, then you must refer to the "Getting the Office 2003 PIAs When Installing .NET Framework 1.1 After Installing Office 2003" section in the article titled "Installing and Using the Office 2003 Primary Interop Assemblies" at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dno2k3ta/html/OfficePrimaryInteropAssembliesFAQ.asp.

Microsoft Office 2003 VBA Language References

The appropriate VBA language reference must be downloaded depending on the Microsoft Office application in which Visual Basic editor is used. This contains programming references, technical articles, and so on, and may contain tools and sample code to help customize and extend the Microsoft Office application to enable interoperability with other applications.

To download this, perform the steps mentioned in "Downloading the Software" by entering **VBA Language Reference** as the search criteria on the Microsoft downloads page. Click the appropriate link depending on the Microsoft Office application used, and then click **Download**.

Microsoft Office 2003 XML Reference Schemas

XML reference schemas represent the structure of Microsoft Word documents, Microsoft Excel spreadsheets, and Microsoft InfoPath form templates. Microsoft Office 2003 Edition XML Schema References and related documentation are available as part of this download.

To download this, perform the steps mentioned in "Downloading the Software" by entering **XML Schemas** as the search criteria on the Microsoft downloads page. Click Office 2003: XML Schemas, and then click **Download**.

Web Services Enhancements

Web Services Enhancements enables developers to leverage the latest Web services protocol specifications and develop secure Web services. Web Services Enhancements 3.0 for Microsoft .NET is an add-on to Microsoft Visual Studio 2005 and the Microsoft .NET Framework 2.0.

To download this, perform the steps mentioned in "Downloading the Software" by entering **Web Services Enhancements** as the search criteria on the Microsoft downloads page. Click Web Services Enhancements (WSE) 3.0 for Microsoft .NET, and then click **Download**.

Microsoft Office 2003 Web Services Toolkit

Microsoft Office 2003 Web Services Toolkit is used to provide XML Web services features with Microsoft Office 2003 applications.

To download this, perform the steps mentioned in "Downloading the Software" by entering **Microsoft Office 2003 Web Services Toolkit** as the search criteria on the Microsoft downloads page. Click **Microsoft Office 2003 Web Services Toolkit** 2.01, and then click **Download**.

Microsoft PKI Utilities

The `makecert.exe` (Certificate Creation tool) is used to generate X.509 certificates for testing purposes, the `cert2spc.exe` (Software Publisher Certificate Test Tool) for digital code signing, and the `pvkimprt.exe` utility to import digital certificate files.

`makecert.exe` and `cert2spc.exe` are available with the installation of Microsoft .NET Framework SDK.

To download this, perform the steps mentioned in "Downloading the Software" by entering **PVK Digital Certificate Files Importer** as the search criteria on the Microsoft downloads page. Click Office 2000 Tool: PVK Digital Certificate Files Importer, and then click **Download**.

Office 2003 Update: Redistributable Primary Interop Assemblies

To download this, perform the steps mentioned in "Downloading the Software" by entering **Redistributable Primary Interop Assemblies** as the search criteria on the Microsoft downloads page. Click **Office 2003 Update: Redistributable Primary Interop Assemblies**, and then click **Download**.

Microsoft Office Schema Tag Lists

Microsoft Office Smart Tag List (MOSTL) enables creation of smart tags as XML files.

To download this, perform the steps mentioned in "Downloading the Software" by entering **XML Schema for Smart Tag Lists** as the search criteria on the Microsoft downloads page. Click **Office XP: XML Schema for Smart Tag Lists**, and then click **Download**.

Microsoft Office WordprocessingML Transform Inference Tool

This tool is also known as the XSLT Inference tool, and it enables transformation of XML files into WordprocessingML by creating XSL transformations.

To download this, perform the steps mentioned in "Downloading the Software" by entering **WordprocessingML Transform Inference Tool** as the search criteria on the Microsoft downloads page. Click **Office 2003 Tool: WordprocessingML Transform Inference Tool**, and then click **Download**.

1.3.4 Coding Languages Supported in Microsoft Office 2003

When performing the tasks outlined in this guide, the following languages can be used to develop code in Microsoft Office applications:

Microsoft Visual Basic for Applications

Microsoft Visual Basic for Applications provides an integrated development environment that can be used to develop rich client applications and integrate them with existing data and systems. Microsoft Visual Basic for Applications is similar to Microsoft Visual Basic, because the elements and programming tools are based on the Microsoft Visual Basic development system.

The Microsoft Visual Basic for Applications development environment is included inside Microsoft Office applications, and can be invoked easily. For example, click ALT+F11 to invoke Microsoft Visual Basic for Applications in Microsoft Word. However, Microsoft Visual Basic for Applications has certain limitations. For example, it is not possible to create a DLL similar to the one created in Section 6.3.3, "Creating the Smart Document DLL". This can be done by using Microsoft Visual Studio, which is described in the next section.

Microsoft Visual Studio

Microsoft Visual Studio enables to efficiently build richer, more interactive applications. Microsoft Visual Studio provides a single integrated development environment for all development tasks, and includes programming languages like Microsoft Visual Basic and Microsoft Visual C# .NET.

These programming languages make it easier to create smart documents and other Microsoft Office interoperability tasks. For example, creating or securing a DLL for a smart document is a lot easier in Microsoft Visual C# .NET.

See Also:

- An example showing how Microsoft Visual Basic for Applications is used to invoke a Web Service in a Microsoft Word document in Section 5, "Completing Forms and Entering Data Using Microsoft Office".
- An example showing how Microsoft Visual C# .NET is used to create a DLL in Chapter 4, "Creating Smart Documents That Interact with Self-Service Business Processes".

2

Understanding Microsoft Office 2003 Extensibility Technologies

This chapter identifies and describes the different extensibility and interoperability technologies that Microsoft Office 2003 has to offer. This chapter requires a good working knowledge of Microsoft Office technology.

> **See Also:** The Microsoft Office Web page for more information about the various Microsoft Office technologies at
>
> `http://www.microsoft.com/office/`

This chapter contains the following sections:

Compared to previous versions, Microsoft Office 2003 has made a lot of progress when it comes to extensibility and interoperability. Enhancements as compared to earlier versions are as follows:

- Microsoft Office 2003 provides Extensible Markup Language (XML) and Web services-based technologies that developers can use to build solutions that extend the Microsoft Office Suite and allow Microsoft Office users to interact easily with enterprise-wide business processes and data. Solutions built in this way provide enhanced quality of decision making, improved document handling capabilities, and smoother flow of information.
- Microsoft Office 2003 offers smart documents, smart tags, and document distribution and management solutions for easier deployment.
- Efficient and productive developers tools and software enable secure and strong solution development.

This chapter describes the specific extensibility technologies and interfaces that Microsoft makes available in Microsoft Office 2003.

2.1 XML Schemas

Microsoft Office 2003 supports XML in two ways:

- XML Reference Schemas: World Wide Web Consortium (W3C)-compliant XML schemas defined by Microsoft to represent the structure of Microsoft Word documents, Microsoft Excel spreadsheets, and Microsoft InfoPath form templates.
- Custom-Defined XML Schemas: Custom-defined XML schemas to define any schema that represents business information so that the associated data can be used within Microsoft Office 2003.

2.1.1 XML Reference Schemas

Definition

XML reference schemas are fully documented and published schemas, which contain data from Microsoft Office documents and enable developers to understand and manipulate documents using any software that can process industry standard XML, without using programs in Microsoft Office 2003. This is advantageous to application developers because it allows server-based document creation, indexing and searching of documents, sharing data across different systems, and formatting information as required. All of this can now be done without the need for Microsoft Office on the server, or the use of complex document object model programming. The different XML reference schemas available are the following:

- WordprocessingML: schema for Microsoft Word 2003
- SpreadsheetML: schema for Microsoft Excel 2003
- FormTemplate Schemas: schema for Microsoft InfoPath 2003

Technology

Each of the following Microsoft Office applications supports XML reference schemas:

Microsoft Word 2003 WordprocessingML, the Microsoft Word 2003 XML format or schema, is available in all versions of Microsoft Word 2003. This schema saves all the information that is saved in the Microsoft Word binary format and has a format similar to the `.doc` binary file format. Microsoft Word or specific templates in Microsoft Word can be configured to save in the WordprocessingML format by default. When a document is saved as WordprocessingML and reopened, all document features are retained by Microsoft Word. Opening a WordprocessingML file from Microsoft Windows Explorer or Microsoft Internet Explorer will open the XML file directly in Microsoft Word. This is because Microsoft Word saves the WordprocessingML file with a processing instruction at the top of the file. When a Microsoft Word document is saved as XML, a detailed XML file with several name spaces is created. However, the structure of a simple WordprocessingML document has only five elements and one namespace.

Most of the document's details are stored as text. Items such as images, Microsoft ActiveX controls, and Microsoft Visual Basic for Application projects that do not provide a mechanism for Microsoft Word to save as text, are stored using base-64 encoding.

Microsoft Excel 2003 SpreadsheetML, the Microsoft Excel 2003 XML format or schema, is available in all versions of Microsoft Excel 2003. Unlike WordprocessingML, SpreadsheetML does not save all details associated with a workbook. When a user opens a SpreadsheetML file from Windows Explorer or Internet Explorer, the XML file

is opened in Microsoft Excel 2003. This is because Microsoft Excel saves the SpreadsheetML file with a processing instruction at the top of the file for this purpose. The content in this schema includes information about workbooks, worksheets, formulas, formatting, and so on.

Microsoft InfoPath 2003 Information captured through InfoPath is saved in the XML format defined by an InfoPath solution developer. Therefore, InfoPath does not define a file format, but uses FormTemplate Schemas, which is saved with a file extension of `.xsf`, to define the layout for InfoPath forms. InfoPath automatically creates and manages files for solutions created by the InfoPath designer. Information, like user interface customizations, XML schemas, views, business logic, events, deployment settings, and so on, which is used within an InfoPath form, is defined in a FormTemplate. The `.xsf` file holds information about how the form is constructed, used, and deployed.

Benefits

In earlier versions, the use of document content in a Microsoft program was restricted. This is because binary file formats were used and as a result, to interpret content, the appropriate Microsoft Office program had to be started and automated through its object model. Also, there was no support for writing or running Microsoft Office programs from a server. With the introduction of XML file formats, it is now possible to read and manipulate XML files from outside of the Microsoft Office programs by using tools and techniques that support XML standards.

This can be useful for developers in the following ways:

- A Microsoft Word document, Microsoft Excel spreadsheet, or InfoPath form template can be assembled on a server by using XML. An XML transform (XSLT) can be applied to information from a database or Web service to create a rich document by adding format and structure to it. This technique is useful, for example, in creating a report periodically such as an on-demand, customized report, or a document with customized information.
- Microsoft Office 2003 onward, information is reused when creating plans and schedules for similar projects. Users often copy information from existing documents, particularly those created from the same template. For example, project plans and schedules are created frequently and large part of the documents are often the same as previous versions. In the past, finding the right document with related information was difficult, and if found, the information had to be copied manually into the new document. Also, building a solution to help the user was difficult as information was difficult to access from the binary file formats. By using the XML file formats, Microsoft Office 2003 documents are saved as text files. These files can then be processed to find the required information and then extract only the desired section from an existing document. This takes away from all the difficulties and shortcomings faced in earlier versions. By using the XML-based solution, even users who are not familiar with XML can now complete a task more efficiently.
- Because XML documents are text-based, they can be read on all platforms. Binary documents cannot be shared easily as they are platform-dependent. XML enables cross-platform computer-based processing. Documents now have a satisfactory transfer format for solutions.
- When information is presented in a Microsoft Word document, formatting is a key aspect that must be considered, and many times content from databases may be displayed without any specific format, for example, when generating reports. By using XML transforms, developers can associate XML data streams with specific

formatting so that, the transformed XML data is displayed in Microsoft Word in the required format. This makes it easier to perform routine formatting work, which can otherwise be very tedious.

2.1.2 Custom-Defined XML Schemas

Definition

The type of data and structure for each element in an XML document is defined in a schema. Developers can specify a custom format for information needed for a particular application by defining it in a schema. This schema can then be used to ensure that valid XML data is captured and formatted.

Technology

The new user interface in Microsoft Office 2003 supports mapping data elements from a schema to specific sections in Microsoft Word and Microsoft Excel documents, and to controls in InfoPath forms. XML tags are invisible to the user of the solution. XML features can be defined from the user interface and by using object model support. Custom-defined XML schema information is retained in the XML document formats to enable further processing specific to the custom XML data. Custom-defined XML schema support in each major Microsoft Office Application is enabled as follows:

Microsoft Word 2003 In addition to retaining the ability to use the rich editing features, including spell check, change tracking, AutoCorrect and more, XML adds structure to a Microsoft Word document in the following ways:

- **Microsoft Word 2003 Task Pane**: Users can map schema elements to specific sections in a Microsoft Word document or template by using the task pane. When editing the document, the structure can be compared against the intended view by toggling between the tagged view and the standard view. Microsoft Word validates the document data against the attached schema as it is entered or edited, flags the errors through the user interface, and raises validation error events. XML tags in the document reveal exact locations of the XML elements. The elements must exist in the same order in the document as they are in the schema. To use a schema that does not have the same structure as the desired document layout, developers must first create a schema to match the document layout, and then change the format of the data when bringing it in or sending it out of the document, by using XSLT transforms.
- **WordprocessingML**: An input data stream can be transformed to become valid WordprocessingML, and can then be inserted in Microsoft Word. Documents using a custom-defined schema can be saved as *data-only*, where Microsoft Word saves only the XML data in a file based on the custom schema's structure. Alternatively, documents can be saved by using the full Microsoft Word XML schema (WordprocessingML). By saving in the WordprocessingML format, all custom XML elements are incorporated, allowing standard XML techniques to be used against the saved XML file to access and change any of the WordprocessingML or custom XML markup. WordprocessingML can be inserted in a document with any valid Microsoft Word formatting characteristics. There are two ways of inserting only data into an existing Microsoft Word template with custom XML tags: transform the XML to WordprocessingML and replace a selection with this, or, copy the XML data element by element.
- **Schema Library**: The Schema Library provides features to manage multiple custom schemas on a computer, and to configure Microsoft Word to process XML by applying a transform. This is possible as the schemas are organized according

to namespaces and a mechanism is provided to associate transforms (XSLT files) and smart documents with a namespace.

Microsoft Excel 2003 Using XML, it is possible to build Microsoft Excel solutions that collect data and provide analytical capabilities. In Microsoft Excel, developers can map the elements of any custom-defined W3C-compliant XML schema within the structure of one or more spreadsheets.

- **Visual Data Mapping Tool**: A visual data-mapping tool is available in a task pane in Microsoft Excel. In Microsoft Excel, mapped elements are designated with blue, nonprinting cell borders. Individual elements are mapped as defined by the schema to single cells, and repeating cells are mapped to the new List feature.
- **Microsoft Excel Lists**: Microsoft Excel Lists can be accessed programmatically by using the object model. However, it is not easy to work with XML that includes repeating elements within other repeating elements.
- **Mapping Multiple Schemas**: Multiple elements can be mapped to the same cell location. Multiple schemas can be mapped to a workbook. Microsoft Excel makes it very easy to bring XML data into a schema map, or to export it. For example, data can be imported into one schema and exported from another.
- **SpreadsheetML**: If an XML file that does not have a reference to a schema is opened in Microsoft Excel, then Microsoft Excel assumes a schema from the XML data and structure. When a workbook is saved as SpreadsheetML, Microsoft Excel saves complete copies of the respective schemas and also saves the map details in the file format.
- **Data Validation and Web Services**: With the two-dimensional layout of a spreadsheet, it is difficult to identify where the user last worked, and therefore, real-time validation against any attached schemas is not performed when using Microsoft Excel. However, based on a request, validation can be performed at any time to notify a user of errors at specific stages of a solution. Microsoft Excel does not provide native support for Web services, but it is easy to receive data from a Web service by using code, because the XML map in a worksheet acts like a display transform. Data can then be placed directly into a map by using one method in the Microsoft Excel object model.

Microsoft InfoPath 2003 Most Microsoft InfoPath form solutions are based on custom-defined XML schemas. XML elements defined by the schema map to controls on Microsoft InfoPath Forms, and Microsoft InfoPath implements the validity of the captured information according to the schema. Information captured in Microsoft InfoPath is stored in an XML file in the format of the custom-defined XML schema. Microsoft InfoPath has no special file format. A form can be based only on one schema. Microsoft InfoPath provides the following benefits:

- When designing a form from a custom-defined schema, InfoPath provides controls based on the data type specified by the schema, for example, a date control is suggested for the XSD date type.
- When captured information is saved as an XML file in InfoPath, to ensure that the document is always connected with that specific form template, InfoPath includes an XML processing instruction at the top of the document that finds the appropriate template solution. Similarly, it is possible to open any XML file in that namespace with the InfoPath form template.

Benefits

By using custom-defined XML schemas in Microsoft Office 2003, developers can work with data that is marked according to certain business rules defined in a schema, instead of plain text items in Microsoft Office documents. Earlier, when custom-defined XML schemas were not supported, this data was available only as numbers in the cells of a spreadsheet or somewhere within a Microsoft Word document region. When an item was moved to another position in the document, it was tedious to find and change every line of code that used the value. This is resolved by mapping an XML schema to the document. It is easy to read or set the value by referring to an XPath statement based on the schema. To access data in this way, it is not necessary to know where the data appears in the document because it directly references the XML structure.

These benefits of working with XML schemas enable any of the following:

- Access data in documents programmatically
- Format data being integrated in Microsoft Office 2003
- Incorporate data from external sources
- Extract data from a document or file

In addition, Microsoft Word and Microsoft Excel enable creation of more intelligent solutions by providing events relating to the custom-defined XML structure in a document. An example is the smart document solution model described subsequently in this guide. Microsoft Word and Microsoft Excel-based solutions can more directly exchange information with Web services. Integration of a document with corporate data and business processes is simplified by using XML and custom-defined XML schemas.

2.2 Smart Technology

Smart technology refers to technology used to build an adaptive, responsive, and rich interactive experience, and to provide the capability to connect to disparate data sources. Smart technology includes smart documents, smart tags, and smart clients. This section contains the following subsections:

2.2.1 Smart Documents

Definition

Smart documents are solutions that enhance the user experience when working with Microsoft Office documents. Smart documents enable developers to perform any of the following:

- Enter data automatically in documents
- Access external data automatically and place it appropriately in a document
- Provide contextual help to guide knowledge workers in the preparation of complicated documents
- Share information in a smart document across disparate systems and applications

Smart documents provide a user with help and additional information, various actions, custom tools, and even custom-designed ActiveX controls to specified sections in the document. When a user works in that section of the document, these items are presented dynamically through the Document Actions task pane. This is possible by making use of programming logic that defines the way documents are used and thereby controlling how data in the documents can be manipulated. Smart document solutions are supported in Microsoft Word 2003 and Microsoft Excel 2003, and make use of the custom-defined XML schema capabilities in these products.

Technology

The following subsections describe how smart documents work.

Overview Smart documents can be used for many tasks like managing a travel request or an absence template form in Microsoft Word or Microsoft Excel.

Developing Smart Documents A smart document solution can be built by using an existing document or by starting from scratch.

The steps involved in developing a smart document are as follows:

1. Attach the document to an underlying XML schema. The document then uses the XML schema as the basis for marking it with corresponding XML elements.
2. Write code to respond to events that start when the user's insertion point enters a document range mapped to an XML element. This code can provide the user with a customized user interface in the task pane.
3. Create an XML expansion pack, which contains information about the solution code, its version, and how it should be installed for the solution to work (that is, on the server, on the client, or in a particular directory).
4. Place the smart document code, expansion pack, and all the files used by the smart document in a trusted location.
5. From the user interface, reference an XML expansion pack and attach the solution to the document or workbook. When the user opens this type of smart document, the expansion pack technology in Microsoft Office 2003 inspects the expansion pack to ensure that the entire solution is available, operational, and secure, and that any new files are downloaded as needed.
6. Make this smart document or workbook available as a template. By using this template, the smart document and any supporting files used by the smart document are downloaded and registered on the user's computer.

The document or template and the code that controls the solution are independent of each other and as a result the deployment and maintenance of smart documents is simplified. It is easier to create new versions and update these documents separately.

Smart documents can be created by using different techniques including:

- ISmartDocument Interface
- Visual Studio Tools for Microsoft Office
- Information Bridge Framework

Each technique is based on a different technology and is suitable for the needs of a different solution.

Deploying Smart Documents Smart document solutions are deployed by using a mechanism that downloads the solution code on-demand when a user opens a document.

Smart document solutions can implement security that is subject to Microsoft Office security settings, demand trusted servers for deployment sites, and require all solution code to be signed.

Smart documents can also update themselves from a trusted server location automatically, making solution upgrades easier. There is no need to install or manage the client-side code directly.

Benefits

Smart documents have the following important benefits:

- Productivity: Users become more productive with smart documents because content is presented in the task pane as they navigate through a document, reducing the time spent searching for or filling in data, or looking for help.
- Familiarity: Users find it convenient and easy as they continue to work in Microsoft Word and Microsoft Excel, which they are familiar with.
- Extensibility: Users can create solutions that enable documents to interoperate with other processes and systems seamlessly, offer a varied user interface, and provide content that is relevant to the specific task that a user must perform within a specific section of a document. This is possible because the programmable task pane can contain any variation of data, help, and common controls (such as buttons, check boxes, option buttons, and list boxes), hyperlinks, images, free text, and more. It is also possible to manage task pane events to perform actions on behalf of the user. All of this allows an enterprise to manage its information with a higher level of integrity.

  See Also: The following chapters, which implement smart documents:

  - Chapter 4, "Creating Smart Documents That Interact with Self-Service Business Processes"
  - Chapter 5, "Completing Forms and Entering Data Using Microsoft Office"
  - Chapter 6, "Securing Smart Documents and Web Services"

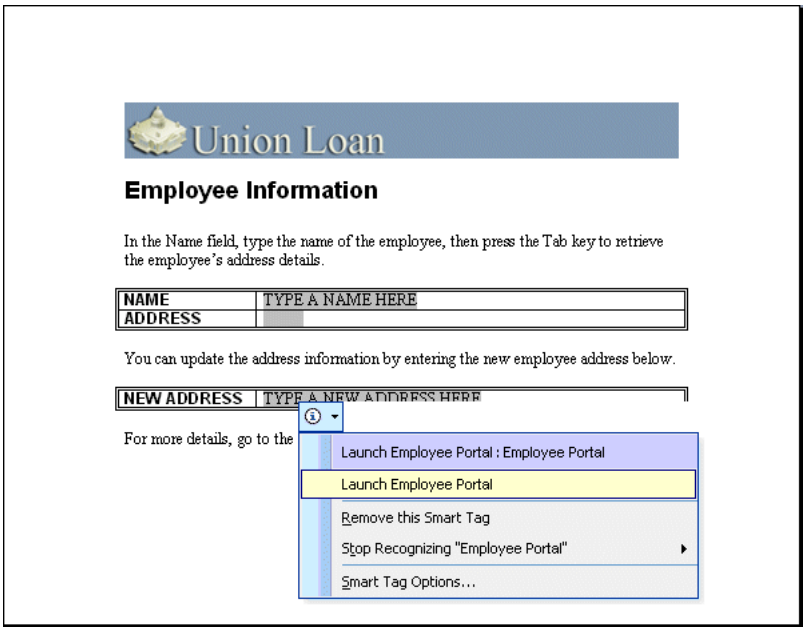
2.2.2 Smart Tags

Definition

Smart tags are pointers in a document that are programmed to identify terms and regions that are of interest to those working with the document. A smart tag appears as a dotted underline, next to which a menu icon appears when the user clicks or moves the mouse cursor over the marked region. The context-specific menus for smart tags present users with actions related to the text, cells, or regions of a document where they appear. Smart tags can be used with names of people by using the Personal menu in Microsoft Outlook and all task panes in Microsoft Office 2003. Developers can also use smart tag actions in Microsoft Internet Explorer for smart tags that are embedded in Web pages.

Figure 2–1 shows an example of a Microsoft Word document that uses a smart tag.

Figure 2–1 Example of a Document with Smart Tags



Technology

Within Microsoft Office 2003, the smart tag feature is available in Microsoft Word, Microsoft Excel, Microsoft Access, and in Microsoft PowerPoint, and in Microsoft Outlook when Microsoft Word 2003 is used as the e-mail editor.

Implementing smart tags requires the following:

- A *recognizer* for recognizing and marking text
- An *action handler* for executing actions

Along with recognizers for dates and times, stock ticker symbols, person names, and addresses, the Microsoft Office system includes a recognizer that identifies terms stored in special text files on the local hard disk. These text files can be edited to include or delete words and phrases that will be recognized by Microsoft Office applications.

Microsoft Office 2003 supports building smart tags by using the following technologies:

- COM DLLs
- Primary Interop Assemblies
- XML files with Microsoft Office Smart Tag List (MOSTL)

Recognizers and action handlers can be implemented using a combination of these technologies.

Benefits

Smart tags provide the ability to recognize key data terms inside documents and e-mail messages, allowing users to efficiently perform custom actions associated with the specific data elements directly from their document. Smart tags can link users to relevant corporate data, streamline cumbersome tasks, or perform any operations that are associated with the tagged data.

See Also: Chapter 11, "Accessing in-Context Web Information and Invoking an Enterprise Portal"

2.2.3 Difference Between Smart Documents and Smart Tags

Though smart documents are based on the Smart Tag Application Programming Interface (API), they are functionally different from smart tags in that the focus is on the document structure, unlike smart tags that focus on content structure. Smart documents are useful when working with data, that is, retrieving it from a database or any other location and saving it.

In smart documents, schema elements are accessed at run time and corresponding elements in specific regions in the document are populated. Users can create solutions that extend documents to interoperate with other processes and systems and provide content that is relevant to the specific task that a user must perform within a specific section of a document.

In smart tags, the smart tag recognizer examines the content of the document for information to which smart tags can be mapped. The context-specific menus present users with actions related to the text, cells, or regions of a document where they appear. Smart tags can link users to relevant corporate data, streamline cumbersome tasks, or perform any operations that are associated with the tagged data.

2.2.4 Smart Clients

Smart clients are client applications that optimize local resources to provide an adaptive, responsive, and rich interactive experience, and connect to disparate data sources. Smart clients are easy to deploy and manage.

To provide a rich user experience, a smart client application includes the following:

Local resources

Smart client applications provide various features, one such being the ability to optimize local resources such as hardware for storage, processing, or data capture. Smart client solutions enhance all that the Microsoft Windows platform offers. Examples of well known smart client applications are Microsoft Word and Microsoft Excel. Smart client applications reside on a user's local computer and can be used online or offline. Smart client applications work efficiently offline and online, but provide a richer experience when online.

Connectivity

By using smart client applications, it is easy to connect to systems across the enterprise or the internet, and exchange data with these systems. Smart client solutions with Web Services, utilize industry standard protocols such as XML, HTTP, and SOAP to exchange information with remote systems. Smart clients also enable synchronization and exchange of information between various back-end systems.

Offline capabilities

Smart client applications, for example, Microsoft Outlook, can work both, offline and online. Local caching and processing capabilities are enhanced to enable operation during periods of no network connectivity or intermittent network connectivity. This is a highly significant feature of smart clients.

Desktop solutions can update back-end systems by taking advantage of offline architecture, and thereby keeping the user interface responsive and improving the

overall user experience. This architecture provides a cost-effective, high-performance system.

2.3 Task Panes

When using Microsoft Office applications, there may be need for more information or options than a toolbar can provide. To address this, Microsoft Office 2003 provides task panes, which provide a common area for additional information and options. This section describes task panes and how to use them.

Description

Task panes appear as fixed dialog boxes on the right-hand side of the workspace in Microsoft Office applications like Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Microsoft Outlook, and Microsoft Access. Most task panes provide access to the Microsoft Office online Web site and look up Microsoft Office templates and Help. Task panes provide features for creating and locating documents, searching for information, tools, and services, formatting documents, researching, and collaborating with peers. The following task panes appear in one or more of these Microsoft Office programs:

- Startup
- Search
- Clipboard
- Insert Clip Art

In addition, Microsoft Office 2003 includes a new task pane called *Research and Reference*, which allows users to research information while working, and to look up references from both internal and external resources.

The Research and Reference task pane, also known as the Research task pane, shares cookies and caching with Microsoft Internet Explorer.

The Research and Reference task pane uses a discovery Web service to provide Research and Reference services. Microsoft Office 2003 provides support for certain basic tasks such as displaying results and copying items to the clipboard. Such tasks do not require any client-side code. However, by performing certain additional tasks, a Research and Reference service can be configured to contain smart tag actions for extended document interactions. To include smart tags in a document, a Dynamic Link Library (DLL) must be installed locally.

2.4 Research and Reference Services

Definition

Research and Reference Services is a new feature introduced with Microsoft Office 2003. This feature is provided by means of a task pane, which allows users to search for a word or phrase in a number of information sources from within the Microsoft Office application. Research and Reference services provide Microsoft Internet Explorer-based search capabilities from within the Microsoft Office applications. This eliminates the need to switch between applications intermittently to search for information when working on Microsoft Office documents.

The Research and Reference task pane is the same in all Microsoft Office applications.

Technology

By default, a few research services are registered and available with Microsoft Office 2003 applications. This includes Search options that come with Microsoft Office 2003, basic resources such as thesauruses and dictionaries in multiple languages, language translation, an online encyclopedia, and Web searching.

In addition, custom research services can be created to expose information from enterprise applications into the Research and Reference task pane. This requires a Web service and a Research and Reference task pane add-in.

Searches can be configured to access local and remote data sources. These data sources can be behind either a corporate firewall or on the Internet. There are no specific security considerations required when using the Research and Reference services. However, for services that may require authentication (for example, if the response contains a link to an installation program for integrated smart tag functions), Microsoft Windows authentication or Internet passwords can be used.

Benefits

As the Research and Reference Services feature is available in most Microsoft Office programs, users can look up of information from within their applications. This service provides a convenient way to look up terms within the context of user documents, and insert content into the documents. Enterprises can benefit by making corporate information available in the task pane to all appropriate users. The services available can be configured and updated from a server or through a policy.

2.5 Microsoft Office 2003 Web Services Toolkit

Definition

Web services are an industry standard used for the following purposes:

- Enabling the exchange of data between applications or systems
- Moving data across heterogeneous systems
- Communicating between distributed systems

Web services use open, XML-based standards and transport protocols to exchange data with applications.

Technology

Microsoft Office applications can be made to interoperate with Web services by using the Microsoft Office 2003 Web Services Toolkit. This toolkit helps in searching for Web services and integrating them with Microsoft applications, by creating the code necessary to interface with basic Web services. Simple Object Access Protocol (SOAP) is one of the main protocols defining the type of communication to a Web service. Developers can interact with Web services by manually interpreting the SOAP messages.

Benefits

By using Web services, data from legacy systems can be exposed to authorized users within the enterprise. Earlier, before Web services were used in such solutions, these users found it difficult to access such useful information.

2.6 Primary Interop Assemblies

Solutions built using managed code, that is, code that runs on the .NET framework can be very productive. For managed code to interoperate with COM interfaces, such as the Microsoft Office object models, developers must use a managed assembly that describes the COM interface types.

Primary Interop Assembly is one such managed assembly, which provides a run time interface for the .NET Framework, and allows applications to bind to the COM types at compile time. To make it easier for developers to write code using Visual Studio .NET to automate Microsoft Office applications, Microsoft provides Primary Interop Assemblies in Microsoft Office 2003 installations, but, for this it is necessary to have installed Microsoft .NET Framework version 1.1. Primary Interop Assemblies are also available for the smart tag and ISmartDocument interfaces.

2.7 Network Deployment of Documents

Code solutions are easier to manage when deployed from a network server. This is a primary reason why Web-based solutions are so popular. Microsoft provides support for deploying Microsoft Office documents from a network server to Microsoft Office clients.

Technology

Microsoft Office 2003 provides two technologies for deploying code for the following types of solutions from a network:

- Smart document solutions. These solutions are built using the ISmartDocument interface with an XML solution manifest file.
- Solutions built using the Visual Studio Tools for Microsoft Office.

Both technologies support document-based solutions for Microsoft Word and Microsoft Excel only. The solution code is placed on a trusted server and the user is given the document or template for the solution. When the network user opens the document, appropriate security checks are made. Custom code is downloaded to the local computer once all security checks are completed and no violations are found. At regular intervals, when the user opens the document, the server is checked for updates.

Server deployment of smart documents built with ISmartDocument interface A smart document built with ISmart document interface will contain a custom document property that has a reference to the XML solution manifest file. Microsoft Office 2003 verifies this document for the following:

- The manifest file is digitally signed
- The code components are signed

Along with the smart document, the code must also exist on a trusted server location. In addition, if the smart document solution is built with managed code, it also undergoes .NET-based security checks.

Once the document passes the security checks, depending on how the solution was configured, the solution code can either be saved on the server or installed locally. If it is installed locally and the solution does not need other network resources, then the solution can be used offline. Depending on the version numbers for each individual file in the XML solution manifest file, solutions will be updated. An XML element in the solution provides information about the frequency at which the server is checked for updates.

Server deployment of documents built with Visual Studio Tools for Microsoft Office
A document built in this way will also contain a custom document property, which, in this case, points to the main interfacing assembly. Once the document passes the security checks, the solution code is downloaded into the Internet Explorer cache. The solution can be accessed from the cache and will work offline. When the system is connected, the local cached copy is updated with a new version, if available.

Benefits

There are three important benefits of network deployment of Microsoft Office documents:

- It is easier to manage a solution from a server as newer versions can be published on the server and the local copy gets updated automatically. In earlier versions, deploying client-side code was an arduous task.
- More of the application and data access privileges in a server-based deployment are managed from a centralized server within a trusted zone, thereby improving the security of desktop solution code.
- A more robust security protocol can be used with solutions that are built with managed code.

3

Understanding Oracle Application Server Interoperability with Microsoft Office

This chapter highlights how different components of Oracle Application Server and Oracle Collaboration Suite interoperate with Microsoft Office. This interoperability ranges from sending e-mails to Microsoft Outlook to complete smart document solutions using Web services and BPEL processes.

This chapter contains the following sections:

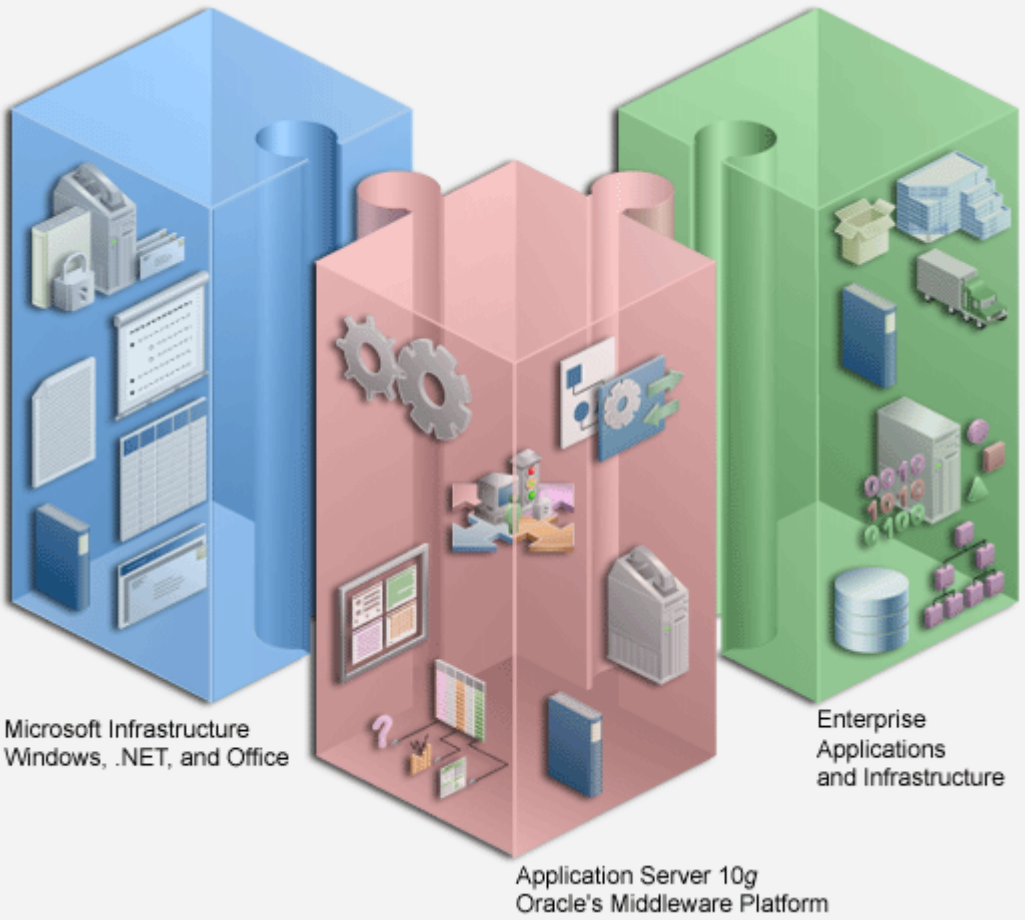
3.1 Overview of Microsoft Office Interoperability with Oracle Application Server

Oracle Application Server can be used to develop enterprise applications that automate transaction processing; streamline business processes; and access and deliver information within the enterprise. With the seamless interoperability capabilities of Oracle Application Server, this information can then be configured to be accessed using Microsoft Office within the context of enterprise applications. There are many ways in which Oracle Application Server components interoperate with the Microsoft Office suite of products.

There is an extensive range of capabilities provided in Oracle Application Server to enable the communication between Microsoft applications and enterprise applications and infrastructure. These capabilities leverage, among other things, the functionality provided within Microsoft Office to make it easier to use Microsoft Office along with XML and Web Services.

Figure 3–1 shows at a high level, how Oracle Application Server interoperates with enterprise applications as well as Microsoft Office applications. It is Oracle Application Server that enables Microsoft applications to communicate with enterprise applications and infrastructure services.

Figure 3–1 Microsoft Office Interoperation with Oracle Application Server



You can see how components of Oracle Application Server described in this chapter enable interoperability between the Microsoft infrastructure (Microsoft Office, Microsoft Active Directory, Microsoft Exchange, .NET applications, and so on.) and disparate enterprise applications, such as packaged and legacy applications, databases, directories, and application servers.

This chapter describes how Oracle Application Server enables enterprise application technology to work with Microsoft Office. Oracle Application Server effectively forms the bridge between the Microsoft components and the back-end applications by taking advantage of a set of standards-based features in Microsoft Office.

3.2 Oracle Application Server Component Support for Microsoft Office Interoperability

This section lists, in alphabetical order, the Oracle Application Server components that interoperate with Microsoft Office including a brief description of the components and their integration points with relevant documentation links. The following Oracle Application Server components are described:

- Oracle Application Server Forms Services
- Oracle Application Server Integration B2B
- Oracle Application Server Integration Business Activity Monitoring
- Oracle Application Server Portal

- Oracle Application Server Web Services
- Oracle Application Server Wireless
- Oracle Business Intelligence Beans
- Oracle Business Intelligence Discoverer
- Oracle BPEL Process Manager
- Oracle Collaboration Suite
- Oracle Identity Management
- Oracle Internet Directory
- Oracle JDeveloper
- Oracle Mobile Collaboration
- Oracle Reports
- Oracle Secure Enterprise Search
- Oracle Xellerate
- Oracle XML Publisher

3.2.1 Oracle Application Server Forms Services

OracleAS Forms Services interoperates with Microsoft office at several points including the following:

- WebUtil interoperates with Microsoft Office in several areas.

  WebUtil is designed for developers who are migrating client/server applications from Microsoft Windows desktops to the Web, but still need some interoperability between their Oracle Forms applications and external packages such as the Microsoft Office Suite running on the client browsers' computers.

- Forms Builder provides several Oracle Forms Built-ins that help in enabling interoperability between Microsoft Office functions and OracleAS Forms Services.

OracleAS Forms Services provides an interface for interoperating with Oracle Application Server-side Microsoft OLE objects. By using WebUtil, client-side objects can also be made interoperable.

> **See Also:** *Oracle Forms Developer WebUtil User's Guide* at
>
> `http://www.oracle.com/technology/products/forms/htdocs/webutil/web_util.pdf`

3.2.2 Oracle Application Server Integration B2B

OracleAS Integration B2B enables business partners to exchange data across networks such as the Internet, and incorporates the partners' host applications and business processes. As business messages are exchanged, instance data exists in the run-time repository. The OracleAS Integration B2B user interface tool enables to query this information to see what is occurring, and to perform business analysis.

Depending on the type of report being generated, this information can be queried in several different ways. Generated reports can be saved in XML files, or in a comma-delimited format. Reports saved in a comma-delimited format can be viewed by using Microsoft Excel.

See Also: *Oracle Application Server Integration B2B User's Guide*

3.2.3 Oracle Application Server Integration Business Activity Monitoring

OracleAS Integration Business Activity Monitoring provides real-time visibility into enterprise operations, which enables business users to cut costs and improve processes while business events, such as a drop in inventory levels, occur. The OracleAS Integration Business Activity Monitoring architecture utilizes messaging, data integration, advanced data caching, analytics monitoring, alerting, and reporting technology to deliver requested critical information within seconds of an event or change in status.

Alerts can be sent when data changes in a report, or periodic reports can be sent to users daily or at set intervals. It is possible to create solutions to send alerts from OracleAS Integration Business Activity Monitoring directly into users' Microsoft Outlook e-mail clients. These alerts can be regular links that open in a browser, or more sophisticated with embedded Microsoft Office documents sent as e-mail attachments.

See Also: Chapter 7, "Delivering Business Activity Monitoring Alerts and Reports to Microsoft Outlook"

3.2.4 Oracle Application Server Portal

To make information accessible and easy to find, the ideal solution is to save the data in one central content repository, such as the one provided by OracleAS Portal. For simple, distributed, low-volume file transfer, the Portal schema in the Oracle Application Server Metadata Repository can be mapped as a Web Folder.

There are several WebDAV tools for saving Microsoft Office documents to the OracleAS Portal content repository. For example, the Portal schema can be mapped as a drive by using Oracle Drive. It is then possible to work with Microsoft Office files within that drive.

See Also: Chapter 12, "Saving Microsoft Office Documents to the OracleAS Portal Content Repository"

3.2.5 Oracle Application Server Web Services

Oracle Application Server Web Services include a set of messaging protocols, programming standards, and network registration and discovery facilities. When they are used together, these features enable the publication of business functions to authorized parties over the Internet from any device connected to the Web.

A Web service supports direct interactions with other software applications using XML-based messages and Internet-based products.

By following a set of guidelines discussed in this guide, developers can use Oracle JDeveloper to create enterprise Web services that can be invoked from Microsoft Office applications, specifically Microsoft Word, Microsoft Excel, and Microsoft InfoPath.

Developers can use the Visual Basic editor included with these Microsoft Office applications to invoke a proxy class to these Web services. This proxy class can be generated by using the Microsoft Office 2003 Web Services Toolkit (a separate download). Alternatively, a proxy class can be created using Microsoft Visual C# .NET and Microsoft Visual Studio.

Representational State Transfer (REST) Web services architecture conforms to the Web architecture defined by W3C, and leverages its architectural principles. REST Web

services use XML documents, not SOAP envelopes, for sending messages. Unlike SOAP Web Services, REST is a "style" and has no standards or tools support from vendors. REST Web services can be deployed similar to OracleAS Web Services. The OracleAS Web Services platform will transform the SOAP response on the server into a REST response before sending it to the client. The REST response will be an XML document whose root element is the first child element of the SOAP body.

OracleAS Web Services can assemble REST Web services only where the use, or encoding mechanism, is literal (`use=literal`). It does not support REST Web services where the message format is encoded.

The Create Java Web Service wizard in Oracle JDeveloper provides an option for enabling REST functions for a Web service. For more information on using Oracle JDeveloper to enable REST functionality in a Web Service, see the Oracle JDeveloper on-line Help.

See Also:

- Chapter 5, "Completing Forms and Entering Data Using Microsoft Office"
- Details about assembling REST Web services in the *Oracle Application Server Web Services Developer's Guide*.

3.2.6 Oracle Application Server Wireless

OracleAS Wireless provides a complete set of Web-based tools, which provide functions for developing and publishing mobile applications, creating mobile users, providing help desk support, and managing the OracleAS Wireless server. OracleAS Wireless enables users to connect from a wireless device to their company's e-mail, calendar, and files systems.

OracleAS Wireless provides support for accessing, searching, and faxing Microsoft Office documents from the users mobile devices.

See Also: *Oracle Application Server Wireless Developer's Guide*

3.2.7 Oracle Business Intelligence Beans

Oracle Business Intelligence Beans enables developers to productively build business intelligence applications that take advantage of the extensive Online Analytical Processing (OLAP) functions in the Oracle Database. OracleBI Beans includes presentation beans: graph and crosstab, data beans: query and calculation builders, and persistence services, which may be deployed in both HTML client and Java client applications. OracleBI Beans is seamlessly integrated into Oracle JDeveloper to provide the most productive development environment for building custom business intelligence applications.

In an OracleBI Beans application, the application developer can let users export data from a crosstab to a text file or to an HTML file that can be read by Microsoft Excel 2000 and later.

The Oracle Business Intelligence Spreadsheet Add-In, which is based on OracleBI Beans, is an add-in to Microsoft Excel and enables users to display data from Oracle OLAP in Microsoft Excel spreadsheets. This add-in is available as part of Oracle Developer Suite, and can also be downloaded from Oracle Technology Network (OTN) at

`http://www.oracle.com/technology/products/bi/spreadsheet_addin/index.html`

See Also: Chapter 8, "Delivering Business Intelligence Information to Microsoft Excel"

3.2.8 Oracle Business Intelligence Discoverer

Oracle Business Intelligence Discoverer is an intuitive ad-hoc query, reporting, analysis, and Web-publishing tool that empowers business users at all levels of the enterprise to gain immediate access to information from data marts, data warehouses, online transaction processing systems and Oracle E-Business Suite.

Using OracleBI Discoverer, a OracleBI Discoverer workbook can be saved as a Microsoft Excel Spreadsheet. The following options are available when exporting to Microsoft Excel:

- Excel worksheet with formatting preserved.
- Excel worksheet with an Excel Pivot Table created. This option is available for OracleBI Discoverer crosstabs.
- Comma-separated values (CSV). This option is suitable when it is not necessary to format information, and when there is a need to conserve the file size.
- Microsoft Excel Web Query (.IQY). This option means that end users can access dynamic OracleBI Discoverer worksheets in Microsoft Excel. It is possible to export data to Microsoft Excel Web Query format from both OracleBI Discoverer Plus Relational and OracleBI Discoverer Viewer.

See Also:

- Chapter 8, "Delivering Business Intelligence Information to Microsoft Excel"
- *Oracle Business Intelligence Discoverer Plus User's Guide*

3.2.9 Oracle BPEL Process Manager

Oracle BPEL Process Manager provides a framework for easily designing, deploying, monitoring, and administering processes based on Business Process Execution Language (BPEL) standards.

Oracle BPEL Process Manager adds value and ease of use to BPEL functionality by providing support in Oracle JDeveloper BPEL Designer for Transformations, workflows, worklists, notifications, sensors, technology adapters, and third-party adapters.

Using Oracle BPEL Process Manager, it is possible to set up a BPEL process, which can create and receive organization alerts that trigger when users' identity information changes. These alerts can be in the form of e-mail notifications with Microsoft Office XML documents sent to the Microsoft Outlook e-mail client of appropriate users.

See Also:

- Chapter 4, "Creating Smart Documents That Interact with Self-Service Business Processes"
- Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts"

3.2.10 Oracle Collaboration Suite

The following Oracle Collaboration Suite components support Microsoft Office interoperability in different ways.

3.2.10.1 Oracle Calendar

Oracle Calendar is scalable scheduling software, based on open standards, for efficiently scheduling people, resources, and events. Among other features, it offers real-time lookups and free-time searches; multiple time zone support, and UTF-8 encoding to support international deployments; e-mail and wireless alerts; multiplatform support and an extensible authentication, compression, and encryption (ACE) framework for enhanced security.

The Oracle Calendar server is the back end to an integrated suite of scheduling and scheduling products. Networked users can use Microsoft Outlook to manage their calendars.

> **See Also:** Overview of Oracle Calendar in the *Oracle Calendar Administrator's Guide* available at
>
> `http://www.oracle.com/pls/cs101/vbook_subject?subject=calendar`

3.2.10.2 Oracle Connector for Outlook

Oracle Connector for Outlook extends Microsoft Outlook to provide a unified environment for e-mail, voicemail, fax, Web conferencing and real-time calendaring. As a MAPI service provider, Oracle Connector for Outlook communicates directly with the calendar and e-mail servers, converting e-mail, fax, voicemail, and calendar data into MAPI constructs for display in the Microsoft Outlook interface.

> **See Also:**
>
> - Chapter 9, "Managing Tasks and Collaborating in Microsoft Outlook"
> - `http://my.oracle.com/portal/page?_pageid=100,4501033&_dad=myo&_schema=PHP`

3.2.10.3 Oracle Drive

Oracle Drive is the desktop client for Oracle Content Services. Oracle Drive enables access to content (files) and file properties through a mapped drive in Windows Explorer, from any Windows applications, and Microsoft Office applications. Content is also accessible through a Web browser.

> **See Also:**
>
> - Section 12.3.3, "Using Oracle Drive as a WebDAV Client"
> - *Oracle Content Services Administrator's Guide* at
>
>   `http://download-west.oracle.com/docs/cd/B25553_01/content.1012/b25275/protocol.htm#sthref396`

3.2.10.4 Real Time Collaboration Add-in for Outlook

The Oracle Real-Time Collaboration Add-in for Microsoft Office provides a convenient way to schedule Web conferences, start instant conferences, or chat with Oracle Messenger users from within Microsoft Office applications such as Microsoft Excel, Microsoft Outlook, Microsoft PowerPoint, or Microsoft Word.

See Also:

- Chapter 9, "Managing Tasks and Collaborating in Microsoft Outlook"
- Troubleshooting information and FAQ at `http://www.oracle.com/technology/products/cs/user_info/ortc/office_addin_index.html`

3.2.11 Oracle Identity Management

Using Oracle Identity Management, it is possible to reduce administrative time and costs by enabling applications and directories to interoperate with Oracle Internet Directory. This includes third-party Lightweight Directory Access Protocol (LDAP) directories. It does this by using Oracle Directory Integration and Provisioning.

Throughout the interoperability process, Oracle Directory Integration and Provisioning ensures that the applications and other directories receive and provide the necessary information in a reliable way. Oracle provides centralized security administration by integrating components with Oracle Identity Management. Similarly, Microsoft provides centralized security administration in Microsoft Windows by integrating all Microsoft applications with Microsoft Active Directory. If the environment uses both Oracle Identity Management and Microsoft Active Directory, then, for these two systems to interoperate, their data must be synchronized. Active Directory Connector that is part of Oracle Directory Integration and Provisioning is used for this purpose.

See Also: Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts"

3.2.12 Oracle Internet Directory

Oracle Internet Directory is a critical component of Oracle Application Server management and security infrastructure. It ensures that user accounts and groups are managed centrally through the LDAP Version 3 standard. Oracle Application Server enables users to be created centrally in Oracle Internet Directory and shared across all components in Oracle Application Server. When users log in, they are authenticated once by Oracle Application Server Single Sign-On against their Oracle Internet Directory credentials, and can thereby access multiple applications seamlessly.

If Oracle Internet Directory is the central directory, and if Microsoft Exchange with Microsoft Active Directory is used, then to ensure up-to-date identity information in Microsoft Outlook contacts, Oracle Internet Directory and Microsoft Active Directory must be synchronized by using Oracle Directory Integration and Provisioning's Active Directory Connector.

See Also: Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts"

3.2.13 Oracle JDeveloper

Oracle JDeveloper is a free, integrated development environment (IDE) with end-to-end support for modeling, developing, debugging, optimizing, and deploying Java applications and Web services.

Oracle BPEL Process Manager can be used to set up BPEL processes for creating alerts, which will be sent to users as e-mail notifications. Oracle BPEL Process Manager is available as part of the Oracle JDeveloper installation, and uses the Oracle JDeveloper BPEL Designer and BPEL Console to build, deploy and test the BPEL processes.

Developers can use Oracle JDeveloper to create enterprise Web services that can be invoked from Microsoft Office applications, specifically Microsoft Word, Microsoft Excel, and Microsoft InfoPath.

See Also:

- Chapter 5, "Completing Forms and Entering Data Using Microsoft Office"
- Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts"

3.2.14 Oracle Mobile Collaboration

Oracle Collaboration Suite 10*g* provides a complete collaborative platform to enterprise customers, including services such as mail, calendar, files (Oracle Content Services), and Web conferences. Because users need access to these services while they are away from their desks, Oracle Collaboration Suite provides the Oracle Collaboration Suite 10*g* Mobile Collaboration. Oracle Mobile Collaboration provides users with a continuous connection to the enterprise, enabling them to access company e-mail, voice mail, calendars, address books, tasks, online files, and directories from any location using any mobile device, including those with voice access.

Oracle Mobile Collaboration provides the mobile and voice access to such browser-based applications as Oracle Collaboration Suite 10*g* Mail, Calendar, Search and Content Services through the following services:

- Oracle Collaboration Suite 10*g* Mobile Access
- Oracle Collaboration Suite 10*g* Mobile Push Mail
- Oracle Collaboration Suite 10*g* Mobile Data Sync
- Oracle Collaboration Suite 10*g* Mobile Device Management

Oracle Mobile Collaboration enables accessing, searching, and faxing Microsoft Office documents from mobile devices.

3.2.15 Oracle Reports

Oracle Reports is a powerful enterprise reporting tool that enables developers to rapidly develop and deploy sophisticated Web and paper reports against any data source, including XML, Microsoft Excel (through JDBC), Text, and so on. Reports built with Oracle Reports can be delivered to Microsoft Office. Report output can be saved as a Microsoft Excel spreadsheet, or a Microsoft Word document, or sent as an e-mail attachment.

See Also: Chapter 13, "Delivering Enterprise Reports to Microsoft Office with Oracle Reports"

3.2.16 Oracle Secure Enterprise Search

Oracle Secure Enterprise Search (OSES) provides uniform search capabilities over multiple repositories. OSES supports searching the following Microsoft documents with the built-in Web, file, and OracleAS Portal source types:

- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint

- Microsoft Access
- Microsoft Words Database
- Microsoft Works Word Processor
- Microsoft Write

> **See Also:** Oracle Secure Enterprise Search documentation page on OTN at
> http://www.oracle.com/technology/products/oses/index.html

3.2.17 Oracle Xellerate

Oracle Xellerate provides interoperability with Microsoft Active Directory, similar to the Oracle Directory Integration and Provisioning synchronization with Microsoft Active Directory, discussed in Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts".

Oracle Xellerate includes the following Microsoft connectors:

- Microsoft Exchange 5.5
- Microsoft Exchange 2000
- Microsoft Active Directory
- Microsoft Active Directory Password synchronization

3.2.18 Oracle XML Publisher

Oracle XML Publisher enables customers to utilize a set of familiar desktop tools to create and maintain their own report formats. These reports formats are based on XML data extracts from their existing Oracle applications. At run time, Oracle XML Publisher merges the custom templates with the concurrent request data extracts to generate output in PDF, HTML, RTF, or Microsoft Excel (HTML). Oracle XML Publisher can interoperate with Microsoft Office Applications as follows:

- **Report layouts** - Use Microsoft Word to build reports layouts. A plug-in to Microsoft Word helps in building the report structures. Microsoft Word functions can then be used to add look and feel to the report layout. This layout is interpreted to an XSL formatting object (XSL-FO) format, and a formatting engine on the server then generates documents in PDF, RTF, Microsoft Excel, and so on.
- **Report output** - Output can be generated in Microsoft Word (RTF) and Microsoft Excel. For both Microsoft Word and Microsoft Excel, Java engines on the server side including the RTF to XSLFO compiler, a binary Excel engine (read/write) and an Excel calculation engine is available. No Microsoft Office components are required on the server to create this output format.

> **See Also:** The Oracle XML Publisher page on OTN at
> http://www.oracle.com/technology/products/applications/publishing/index.html

Part II

Building Microsoft Office Interoperability Solutions

Part II contains a number of step-by-step procedures that describe how you can make Oracle Application Server components interoperate with Microsoft Office. It contains the following chapters:

4

Creating Smart Documents That Interact with Self-Service Business Processes

This chapter describes how you can create a smart document that interacts with a BPEL process that sends an e-mail message with a Microsoft Word XML file as an attachment.

It contains the following sections:

4.1 Overview

You can perform self-service tasks from within Microsoft Office applications by using Smart documents, avoiding the need to switch between browsers and your applications. Smart documents can automatically enter data, making it easier for users to complete forms or work with templates. Smart documents can also automatically access external data and place it in the right place in a document, or provide contextual help to assist you in the preparation of documents. For more details, see section Section 2.2.1, "Smart Documents".

Oracle BPEL Process Manager provides a user-friendly and reliable solution for designing, deploying, and managing BPEL business processes. The built-in integration services let you use advanced connectivity and transformation capabilities of standard BPEL processes, which support XSLT and XQuery transformation, and bindings to legacy systems through J2EE Connector Architecture (JCA) adapters and native protocols. A user task service is provided as a built-in BPEL service to enable you to participate in BPEL workflows. For more details, see `http://www.oracle.com/technology/bpel`.

4.2 Prerequisites

To perform the steps outlined in this chapter, the following software must be installed:

- Oracle BPEL Process Manager 10*g* Release 2 (10.1.2.0.2)
- Microsoft Office 2003 Professional

Note: You must install the .NET Framework before you install Microsoft Office. If you have installed Microsoft Office first, then you must refer to the "Getting the Office 2003 PIAs When Installing .NET Framework 1.1 After Installing Office 2003" section in the article titled "Installing and Using the Office 2003 Primary Interop Assemblies" at `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dno2k3ta/html/OfficePrimaryInteropAssembliesFAQ.asp`.

- The .NET Framework and Software Developer Kit (version 1.1 or later)

  Refer to Section 1.3.3, "Microsoft Software Development Kits, Utilities, and References" for details about downloading this software.

- The Microsoft Office WordprocessingML Transform Inference Tool (wml2xslt.exe)

  Refer to Section 1.3.3, "Microsoft Software Development Kits, Utilities, and References" for details about downloading this software.

- E-mail Server Configuration. This business process sends e-mail notifications and requires e-mail settings. Refer to Section 4.3.1, "Configuring the E-Mail Server" for details.

- The support files in the `selfservice` demonstration folder. Refer to Accessing the Demonstration Support Files in the Preface for details about the demonstration support files. The support files and folders in the `selfservice` demonstration folder are listed and described in Table 4–1.

Table 4–1 Self-Service Files

File or Folder	Description
`README.html`	The readme file for this demonstration. Contains instructions on installation and configuration of files.
`build.xml`	ANT build script file
`LoanDemoWordSD` (folder)	This folder contains the code examples needed to build the smart document solution.
`AutoLoanCreditRatingService` (folder)	This folder contains the AutoLoanCreditRatingService BPEL files.
`AutoLoanFlow` (folder)	This folder contains the AutoLoanFlow BPEL files.
`UnionLoan` (folder)	This folder contains the UnionLoan BPEL files.
`UnionLoanUI` (folder)	This folder contains the UnionLoanUI BPEL files.
`WesternLoan` (folder)	This folder contains the WesternLoan BPEL files.

Note: Download the contents of the `selfservice` folder into a new folder named `MSOfficeIntegration` under the *`BPEL_ORACLE_HOME`*`\integration\orabpel\samples\demos` folder, for example, `C:\OraBPELPM_1\integration\orabpel\samples\demos\MSOfficeIntegration\`.

4.3 Step-by-Step Procedures

In this scenario, a loan applicant seeking a car loan from the Fast Loan broker submits a loan application using a Microsoft Word smart document. The Word document initiates a BPEL process workflow that first gets the applicant's credit rating.

If the applicant has a good credit rating, the BPEL process sends the loan application details to two different loan companies - Western Loan and Union Loan. In one company, the loan approval process is automated. In the other, loans are approved manually.

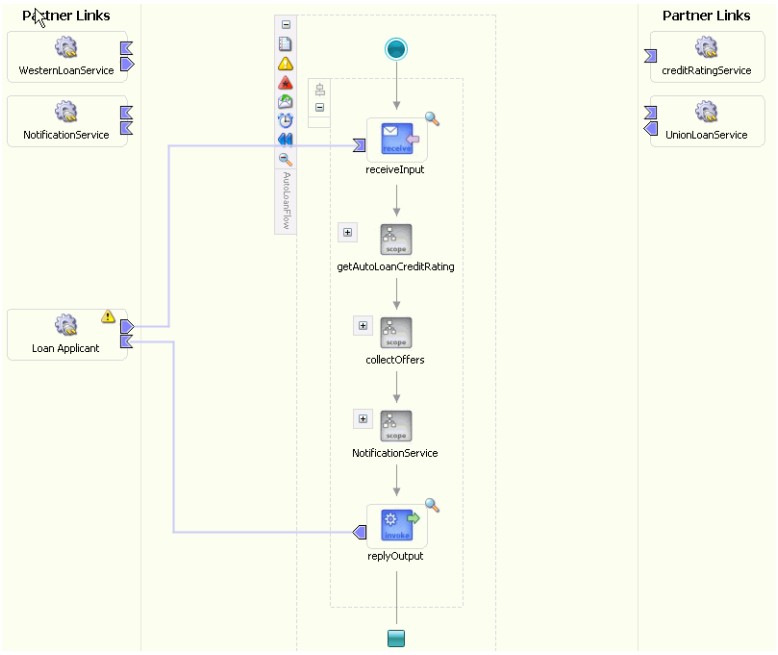
The BPEL process takes the replies from the two loan companies, and sends the one with the lowest APR to the loan applicant using a Microsoft Word 2003 XML document.

If the loan applicant has negative credit rating, the process sends an e-mail to the loan applicant with a loan rejection message.

This chapter describes how you can create a smart document loan application form, create a Microsoft Word template for the loan result notification, configure your e-mail server, and deploy required BPEL processes.

Figure 4–1 gives an overview of the BPEL process used by the smart document for processing this sample loan application scenario, which is explained in detail in this chapter. The smart document invokes a BPEL process and provides the expected input. The BPEL process takes the input and passes it to the getAutoLoanCreditRating function, which sends the details to the creditRatingService. The loan application details are then passed to WesternLoanService and UnionLoanService. The loan offers from the two loan services are evaluated and the best offer is passed to the NotificationService that sends the reply to the loan applicant.

Figure 4–1 BPEL Flow



Perform the steps in the following sections to create a smart document template that interacts with a BPEL workflow process:

- Configuring the E-Mail Server

- Deploying the BPEL Process
- Creating a Smart Document Form
- Creating the Microsoft Word Template for the Loan Result Notification
- Validating the Solution

4.3.1 Configuring the E-Mail Server

The `ns_emails.xml` file in the directory *`BPEL_ORACLE_HOME`*`\integration\orabpel\system\services\config` contains the configuration for e-mail accounts. Each `EmailAccount` element sets the configuration of a specific e-mail account. The `name` attribute in the `EmailAccount` element is the name of the account.

A default e-mail account is specified in the e-mail configuration file. This account is used when there is no account specified to which to send an e-mail notification. This account is also used for task-related notifications. A default e-mail account must always be specified in the configuration file.

The `EmailAccount` element contains the `OutgoingServerSettings` and `IncomingServerSettings` attributes. For notifications that require action in a workflow process, both `IncomingServerSettings` and `OutgoingServerSettings` attributes are required.

Table 4–2 describes the XML elements for the e-mail notification configuration stored in the `ns_emails.xml` file.

Table 4–2 XML Elements for E-Mail Server Configuration

Name	Description
EmailAccount/Name	Name of the account. This can be any name, but must be unique within this server.
EmailAccount/GeneralSettings/From Name	Name of the `From` e-mail address.
EmailAccount/GeneralSettings/From Address	E-mail address for the `From` e-mail address.
EmailAccount/OutgoingServerSettings /SMTPHost	Name of the outgoing SMTP server.
EmailAccount/OutgoingServerSettings /SMTPPort	Port of the outgoing SMTP server.
EmailAccount/IncomingServerSettings /Server	Name of the incoming e-mail server.
EmailAccount/IncomingServerSettings /Port	Port of the incoming e-mail server.
EmailAccount/IncomingServerSettings /UserName	User ID of the e-mail address.
EmailAccount/IncomingServerSettings /Password	User password.
EmailAccount/IncomingServerSettings /Password[encrypted]	Encrypted attribute of the password. It is `true` if the password is encrypted and `false` if it is not. Generally, you should set this to `false` when you first enter the password. The server automatically encrypts the password the first time it reads the configuration file and sets the attribute to `true`.

Table 4–2 (Cont.) XML Elements for E-Mail Server Configuration

Name	Description
EmailAccount/IncomingServerSettings/UseSSL	Secure sockets layer (SSL) attribute. It is `true` if the incoming server requires SSL and `false` if it does not.
EmailAccount/IncomingServerSettings/Folder	Name of the folder from which to read the incoming messages.
EmailAccount/IncomingServerSettings/PollingFrequency	Polling interval for reading messages from the incoming messages folder.

See Also: The information about configuring an e-mail server in the *Oracle BPEL Process Manager Developer's Guide*, located in Oracle Application Server 10*g* Release 2 (10.1.2.0.2) Documentation library at `http://www.oracle.com/technology/documentation/appserver101202.html`

Click **View Library** in the Oracle Application Server 10*g* Release 2 (10.1.2.0.2) table, and then click the **E-Business Integration** tab.

4.3.2 Deploying the BPEL Process

To deploy the BPEL process, perform the following steps:

1. Download the sample demonstration support files from OTN and save it in the *BPEL_ORACLE_HOME*`\integration\orabpel\samples\demos` folder, for example, `C:\OraBPELPM_1\integration\orabpel\samples\demos\`.

   Refer to "Accessing the Demonstration Support Files" in the Preface for details about downloading the demonstration ZIP file.

2. Start the Oracle BPEL Process Manager server. Click **Start**, **All Programs**, **Oracle - *ORACLE_HOME***, **Oracle BPEL Process Manager 10.1.2**, and then **Start BPEL PM Server**.

3. Open a command prompt window, or start the Oracle BPEL Process Manager developer prompt. Click **Start**, **All Programs**, **Oracle - *ORACLE_HOME***, **Oracle BPEL Process Manager 10.1.2**, and then **Developer Prompt**.

4. Navigate to the *BPEL_ORACLE_HOME*`\integration\orabpel\samples\demos\MSOfficeIntegration\` folder.

5. Run the following command:

   ```
   BPEL_ORACLE_HOME\integration\orabpel\bin\obant
   ```

   You will see a `BUILD SUCCESSFUL` message in the command prompt. For example:

   ```
   BUILD SUCCESSFUL
   Total time: 33 seconds
   ```

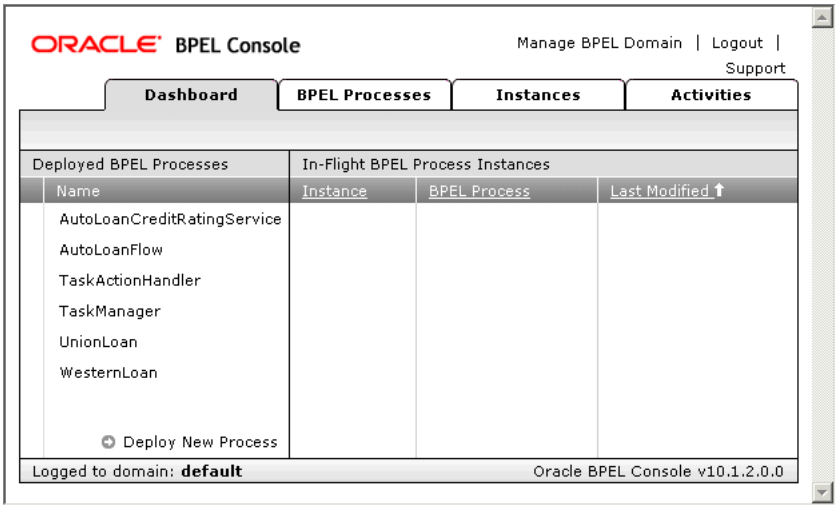
6. Verify and test that the BPEL processes are deployed by performing the following steps:

   a. Click **Start**, **All Programs**, **Oracle - *ORACLE_HOME***, **Oracle BPEL Process Manager 10.1.2**, and then **BPEL Console**.

   b. Log in by specifying the BPEL developer credentials.

You will see the BPEL Console page as shown in Figure 4–2. Ensure that the AutoLoanFlow, UnionLoan, WesternLoan, and AutoLoanCreditRatingService processes are displayed in the list of deployed processes.

Figure 4–2 BPEL Console Page



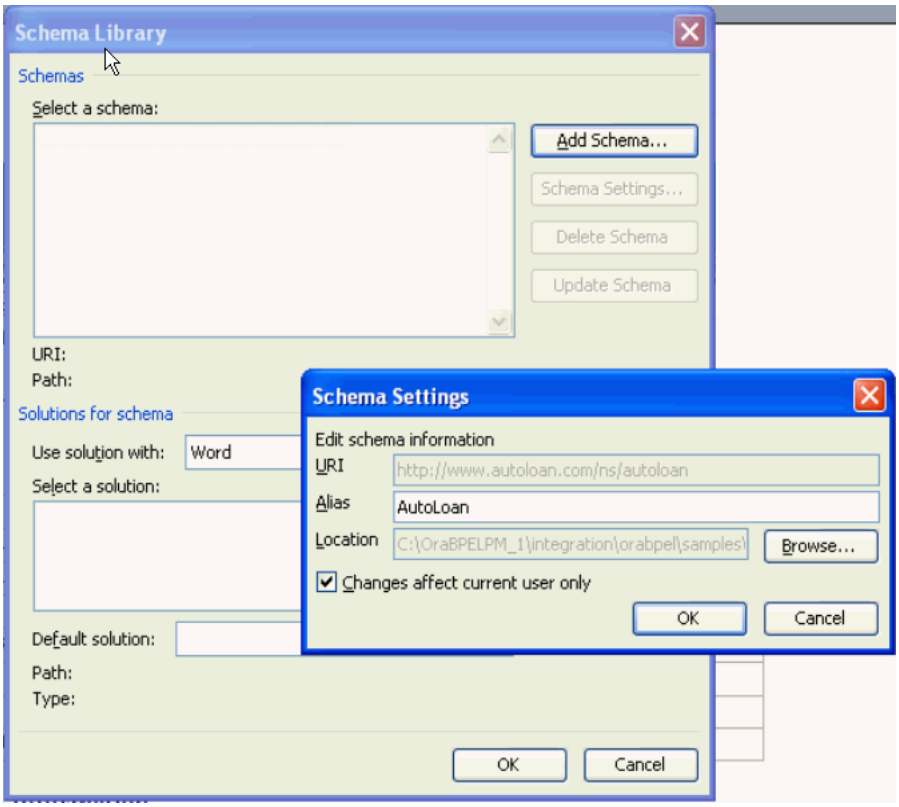
4.3.3 Creating a Smart Document Form

Smart documents are solutions that enhance the user experience when working with Microsoft Office documents. In this section, the steps involved in creating a loan application form that can be used by the auto loan process are discussed.

To create a smart document, perform the following steps:

1. Navigate to the *BPEL_ORACLE_HOME*\integration\orabpel\samples\demo\MSOfficeIntegration\LoanDemoWordSD folder. For this example, you can use the LoanDemo.doc file provided, or create a new document in Microsoft Word.
2. Start Microsoft Word.
3. Save a blank document as LoanDemo.doc in the LoanDemoWordSD folder, or use the LoanDemo.doc file provided.
4. Attach the AutoLoanTypes.xsd schema to LoanDemo.doc. To do this, perform the following steps:
   a. From the Microsoft Word menu bar, click **Tools**, and then select **Templates and Add-Ins**.
   b. In the Templates and Add-ins dialog box, select the **XML Schema** tab.
   c. Click **Schema Library**.
   d. Remove any schema that uses the namespace http://www.autoloan.com/ns/autoloan. If there is such a schema, then select it and click **Delete Schema**.
   e. Click **Add Schema**, and browse to *BPEL_ORACLE_HOME*\integration\orabpel\samples\demo\MSOfficeIntegration\LoanDemoWordSD\AutoLoanTypes.xsd. Name the schema **AutoLoan**, and click **OK** as shown in Figure 4–3.

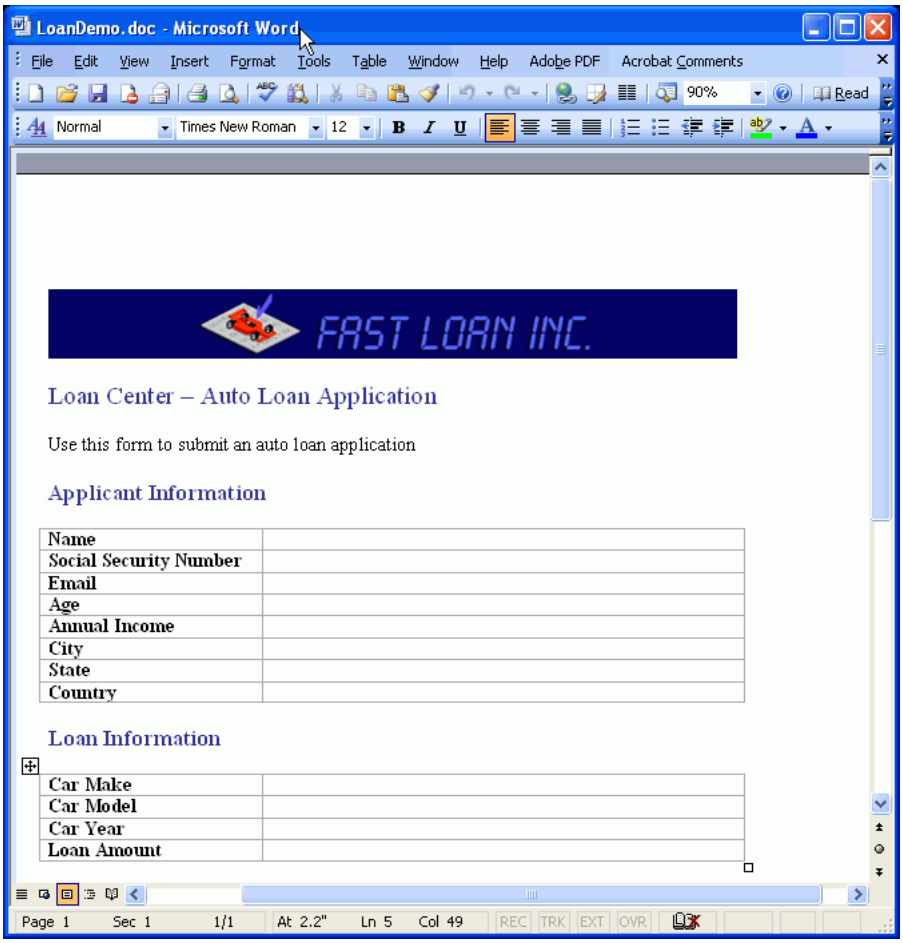
Figure 4–3 Adding a Schema



f. Click **OK**, and then **OK** again to exit.

5. Add content to the `LoanDemo.doc` file in the form of banners, tables, and so on, as shown in figure Figure 4–4.

Figure 4–4 Adding Content to LoanDemo.doc



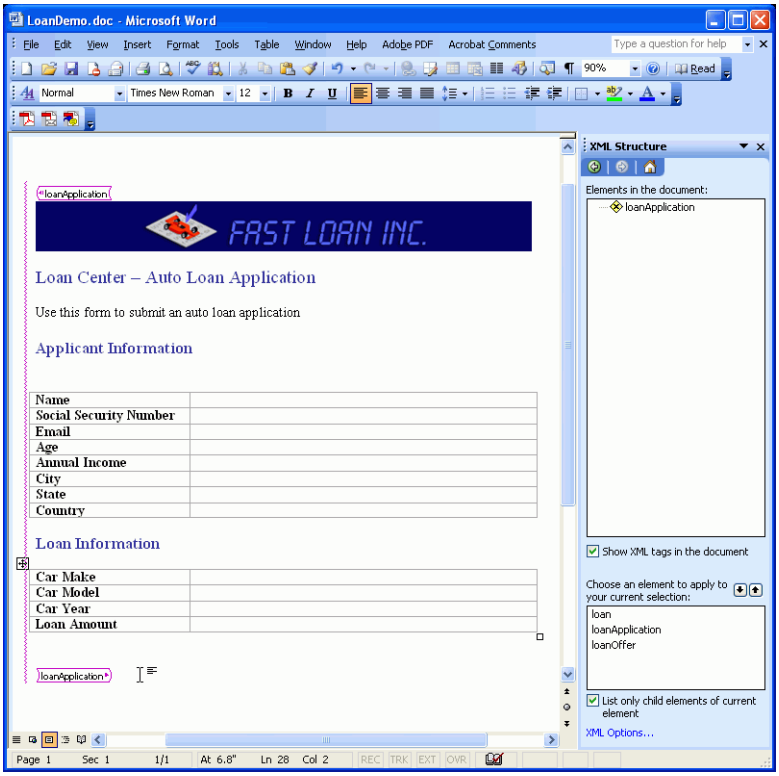
6. Add structure to your word document by mapping XML elements to the word document. To do this, perform the following steps:

   a. Click **View**, **Task Pane**, and then select **XML Structure** in the Task Pane.

   b. Add the correct elements into the document, as shown in Figure 4–5. Place the cursor just below the image. When the cursor is in this position, you can map only three elements that are the root elements defined in the XML Schema Document (XSD).

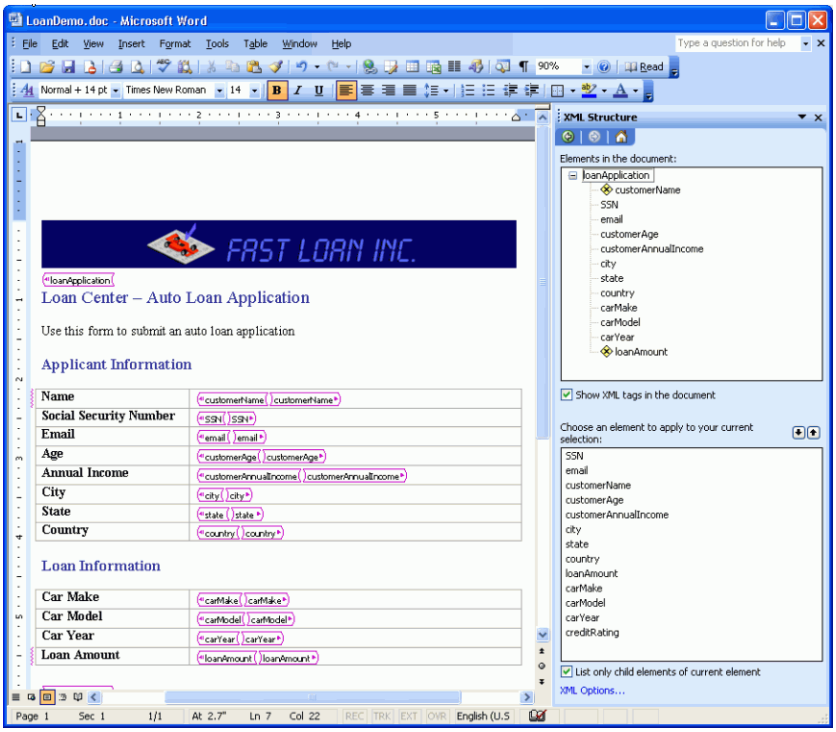
      Click the **loanApplication** element. When prompted, click **Apply to Entire Document**.

Figure 4–5 XML Structure Pane Showing Available Root Elements



c. When the cursor is placed anywhere within the `<loanApplication>` tag, the XML Structure pane shows the child elements of the `loanApplication` element. Add the child elements to the corresponding table cells, as shown in Figure 4–6.

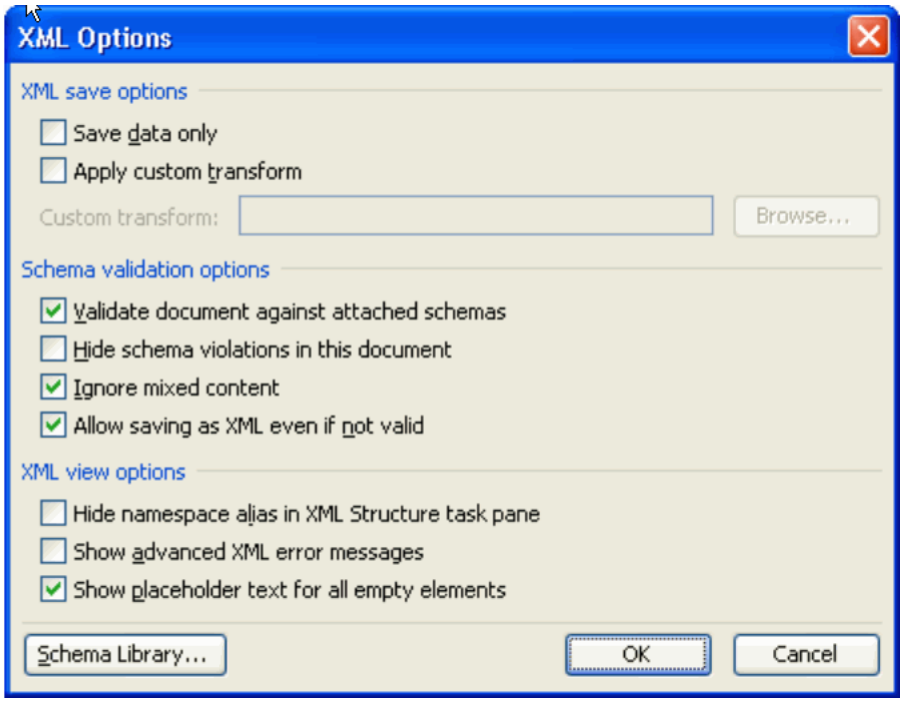
Figure 4–6 XML Structure Pane Showing Available Child Elements



7. Set XML options by performing the following steps:

   a. In the XML Structure pane, click the **XML Options** link. The XML Options dialog box is displayed as shown in Figure 4–7.

Figure 4–7 XML Options Dialog Box



b. If the document contains both unstructured text and XML elements, then select **Ignore Mixed Content**.

c. To hide the XML elements in the document, select **Show placeholder text for all empty elements**.

d. If you want to save XML that does not conform to the schema, then select **Allow saving as XML even if not valid**.

e. Click **OK**.

8. In the XML Structure pane, clear the **Show XML tags in the document** option to hide the XML structures.

9. Create an XML file called `ManagedManifest.xml`, and add the code from Section A–2, "ManagedManifest.xml for Chapter 4" to this file.

10. Enable or disable the manifest security check of the smart document manifest file. To enable the manifest security check, follow the steps described in Section 6.3.4.2, "Enabling Manifest Security Check".

    To disable the manifest security check, perform the following steps:

    Important: You must disable XML expansion pack manifest security checking within a testing environment only and not on the end users' computers. The option to disable XML expansion pack security checking helps developers to easily test smart documents in the development phase. For more information about security checking for XML expansion packs, see "Security for XML Expansion Packs" at http://msdn.microsoft.com/library/en-us/sdsdk/html/sdconSecurityXMLExpansionPacks_HV01074377.asp.

    a. Back up the Windows registry. In the Registry Editor, click **Registry**, and then click **Export Registry File**, and save it in a suitable location.

    b. Navigate to `HKEY_LOCAL_MACHINE/Software/Microsoft/Office`.

    c. Create a key called **Common**.

    d. Under Common, create a key called **Smart Tag**.

    e. In the right pane, right-click and select **New**, and then select **DWORD Value**.

    f. Enter `DisableManifestSecurityCheck` in the Name field.

    g. Right-click **DisableManifestSecurityCheck**, and select **Modify**.

    h. Enter `00000001` in the Value data field.

11. Save the document.

12. Generate a Web service proxy by using the Microsoft `WSDL.exe` tool. To do this, open the Windows command prompt window, and navigate to `C:\OraBPELPM_1\integration\orabpel\samples\demos\MSOfficeIntegration\LoanDemoWordSD` folder, and run the following command:

    ```
    "C:\Program Files\Microsoft.NET\SDK\v1.1\Bin\WSDL" /l:CS /protocol:SOAP
    http://localhost:9700/orabpel/default/AutoLoanFlow/1.0/AutoLoanFlow?wsdl
    ```

    This creates the proxy class file `AutoLoanFlow.cs` in the `LoanDemoWordSD` folder.

13. Include the root element of the WSDL in the `AutoLoanFlow.cs` file for serialization. Place the text in **bold**, just below the line as shown in Example 4–1:

Example 4–1 WSDL Root Element in the AutoLoanFlow.cs File

```
/// <remarks/>
[System.Xml.Serialization.XmlTypeAttribute(Namespace="http://www.autoloan.com/ns/a
utoloan")]
[XmlRoot (ElementName="loanApplication",
Namespace="http://www.autoloan.com/ns/autoloan")]
public class LoanApplicationType {
```

The smart document invokes this Web service proxy, which in turn invokes the Web service as if it were local.

14. The Web service proxy assumes that the BPEL server is running on the same host at port `9700`. If this is incorrect, then change it in the *`BPEL_ORACLE_HOME`*`\integration\orabpel\samples\demos\MSOfficeIntegration\LoanDemoWordSD\AutoLoanFlow.cs` file. The file must be updated with the correct port number, as shown in **bold**, in Example 4–2:

Example 4–2 Host and Port Entries in the AutoLoanFlow.cs File

```
public class AutoLoanFlowBinding :
System.Web.Services.Protocols.SoapHttpClientProtocol {

    public EndpointReferenceType ReplyTo;

    public AttributedURI MessageID;

    /// <remarks/>
    public AutoLoanFlowBinding() {
        this.Url = "http://localhost:9700/orabpel/default/AutoLoanFlow/1.0";
    }
```

If you change the values, then you must re-create the DLL file, as described in Step 16.

15. Create the smart document implementation class file `AutoLoanSmartDocument.cs` by pasting the code from Section A–1, "AutoLoanSmartDocument.cs" into a text file, and saving it in the `LoanDemoWordSD` folder. Alternatively, you can use the `AutoLoanSmartDocument.cs` file provided in the `LoanDemoWordSD` folder.

> **Note:** The AutoLoanSmartDocument class implements the `Microsoft.Office.Interop.SmartTag.ISmartDocument` interface as shown in `AutoLoanSmartDocument.cs`.

16. Run the following command from a command prompt window:

```
%WINDIR%\Microsoft.NET\Framework\v1.1.4322\csc /t:library
/reference:"%WINDIR%\assembly\GAC\Microsoft.Office.Interop.SmartTag\11.0.0.0__
71e9bce111e9429c\Microsoft.Office.Interop.SmartTag.dll";"%WINDIR%\assembly\GAC\
Microsoft.Office.Interop.Word\11.0.0.0__
71e9bce111e9429c\Microsoft.Office.Interop.Word.dll" AutoLoanSmartDocument.cs
AutoLoanFlow.cs
```

Verify that the `csc` path and .NET Framework SDK version are correct. See the text in **bold** in the preceding command.

This creates the `AutoLoanSmartDocument.dll` file, which makes up the brains of your smart document.

17. From the `LoanDemoWordSD` folder, run the following commands from a command prompt window:

```
%WINDIR%\Microsoft.NET\Framework\v1.1.4322\caspol -pp off -ag 1.1 -url
"C:\OraBPELPM_
1\integration\orabpel\samples\demos\MSOfficeIntegration\LoanDemoWordSD\*"
FullTrust -n LoanDemo

%WINDIR%\Microsoft.NET\Framework\v1.1.4322\caspol -pp on
```

Verify the path of the `MSOfficeIntegration\LoanDemoWordSD` directory and `caspol`, if required.

This gives full trust to the smart document library file.

18. Attach the XML expansion pack to the document. To do this, perform the following steps:

    a. From the Microsoft Word menu bar, click **Tools**, and then click '**Templates and Add-Ins**.

    b. In the Templates and Add-ins dialog box, select the **XML Expansion Packs** tab.

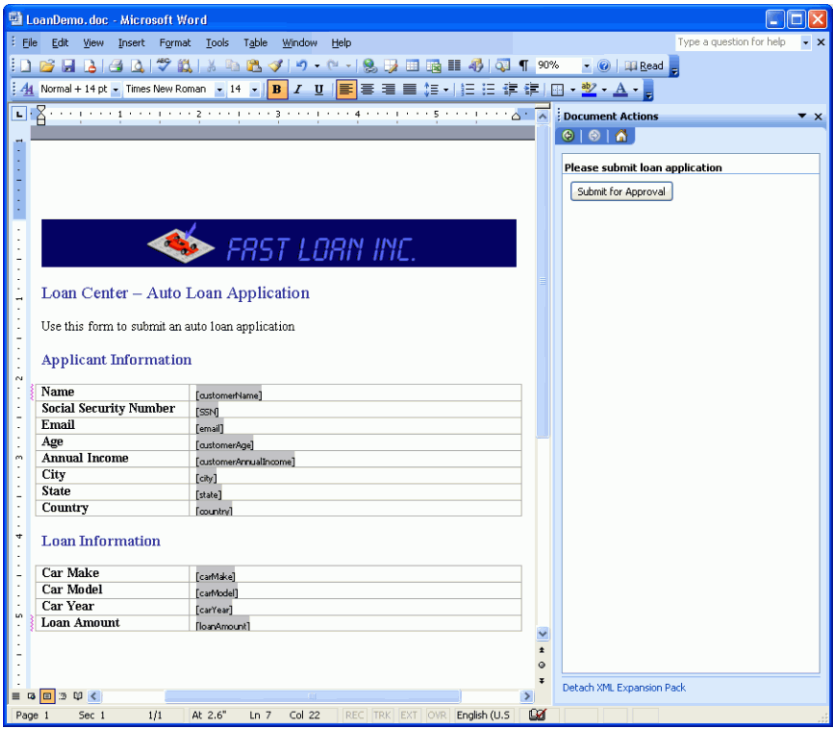
    c. Click **Add** and select *`BPEL_ORACLE_HOME`*`\integration\orabpel\samples\demos\MSOfficeIntegration\LoanDemoWordSD\ManagedManifest.xml`.

    d. In the security dialog box, decide whether you want to enable or disable XML expansion pack security. If you disabled the manifest security check earlier, then you must click **No**.

    e. Click **OK**.

19. Go to the Document Actions pane to see the document. Place the cursor within the `<loanApplication>` tags to see the Submit for Approval button, as shown in Figure 4–8.

Figure 4–8 Structured Smart Document



4.3.4 Creating the Microsoft Word Template for the Loan Result Notification

This section provides information on how to create a Microsoft Word template based on the smart document form created, as explained in Section 4.3.3, "Creating a Smart Document Form". This template is used by the AutoLoanFlow process to create a Microsoft Word document with the loan details and sends it to the e-mail address of the loan applicant.

To create a Microsoft Word template, perform the following steps:

1. Start Microsoft Word.
2. Save a blank document as `LoanResult.doc` in the *`BPEL_ORACLE_HOME`*`\integration\orabpel\samples\demo\MSOfficeIntegration\LoanDemoWordSD` folder. Alternatively, you can use the `LoanResult.doc` file provided in the `LoanDemoWordSD` folder.
3. Attach `AutoLoanTypes.xsd` to the document. To do this, perform the following steps:
   a. From the Microsoft Word menu bar, click **Tools**, and then select **Templates and Add-Ins**.
   b. In the Templates and Add-ins dialog box, select the **XML Schema** tab.
   c. Click **Schema Library**.
   d. Remove any schema that uses the namespace http://www.autoloan.com/ns/autoloan. If there is such a schema, then select it and click **Delete Schema**.
   e. Click **Add Schema**, and browse to *`BPEL_ORACLE_HOME`*`\integration\orabpel\samples\demo\MSOfficeIntegration\`

`LoanDemoWordSD\AutoLoanTypes.xsd`. Name the schema **AutoLoan**, and click **OK**.

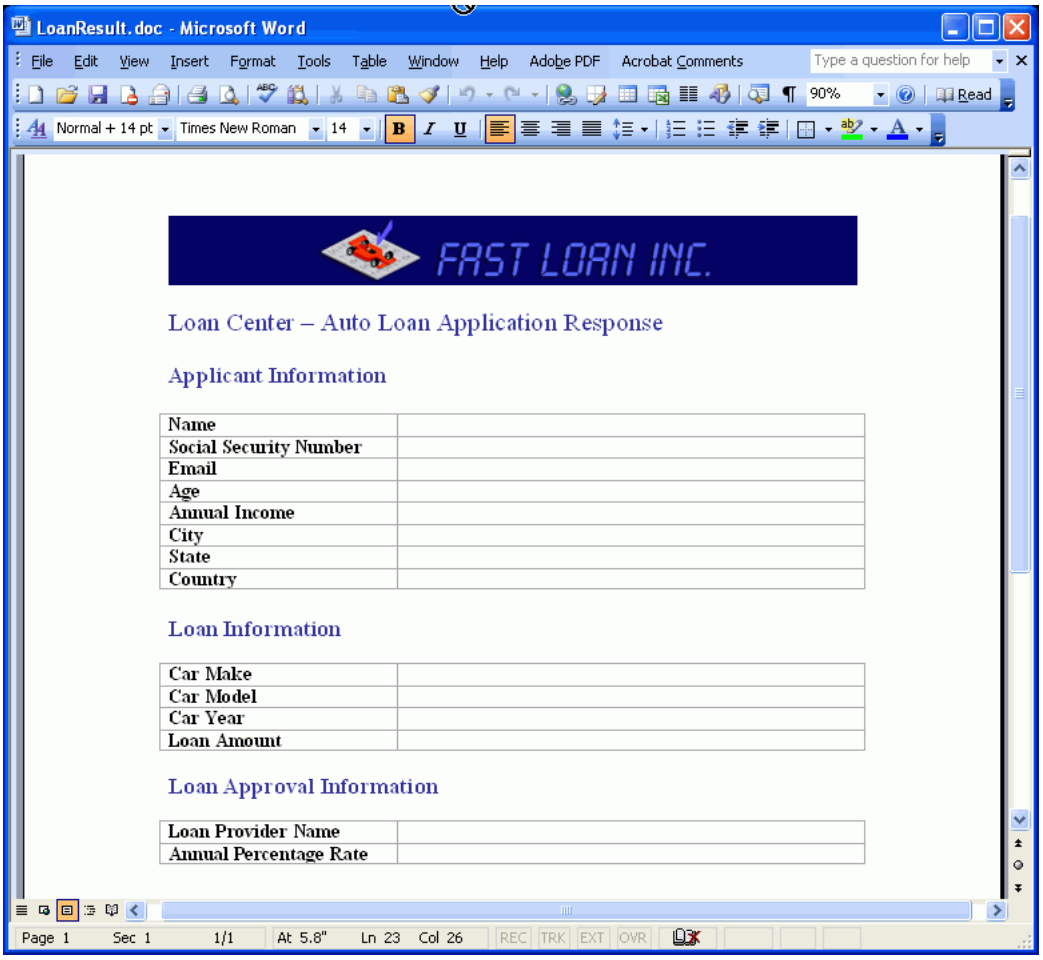
Note: Alternatively, you can create a new XSD file, copy the code from Section A.4, "Contents of the AutoLoanTypes.xsd File" into it, and save it as `AutoLoadTypes.xsd`. You can then add this schema.

---

f. Click **OK**, and then **OK** again to exit.

4. Add content to the `LoanResult.doc` file in the form of banners, tables, and so on, as shown in Figure 4–9.

Figure 4–9 Adding Content to LoanResult.doc



5. Add structure to your word document by mapping XML elements to the word document. To do this, perform the following steps:

a. Click **View**, **Task Pane**, and then **XML Structure**.

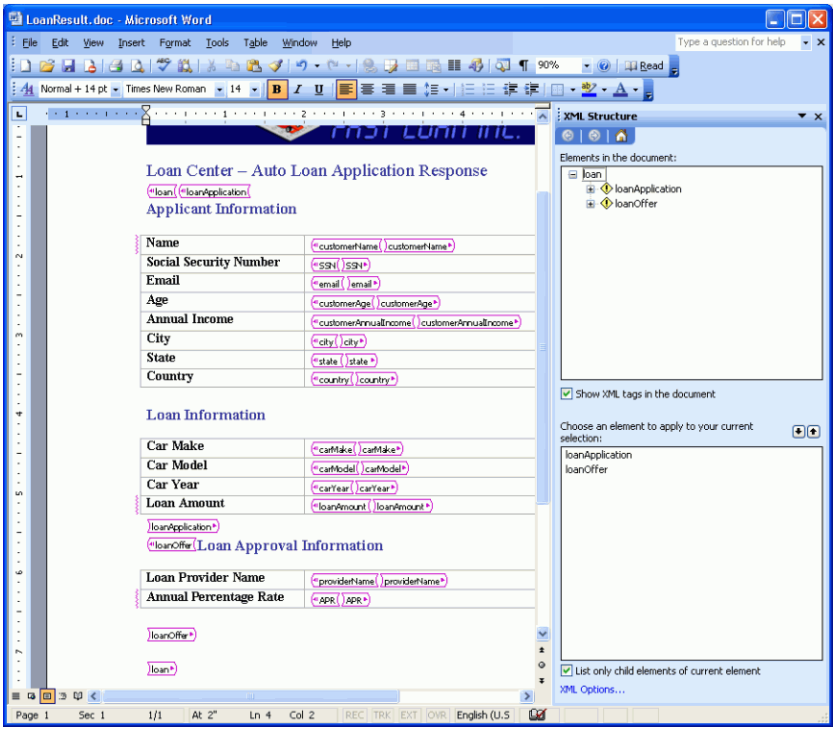
b. Place the cursor just below the banner image and click the **loan** element. When prompted, click **Apply to Entire Document**. The document displays data from both children of the `loan` element – `loanApplication` and `loanOffer`.

c. Place the cursor below the image and select the content up to the second table in the document. Click the **loanApplication** element.

d. Select the text Loan Approval Information and the subsequent table. Click the **loanOffer** element.

e. Add the child elements to the corresponding table cells, as shown in Figure 4–10

Figure 4–10 XML Structure of the Loan Result Document



6. Set XML options by performing the following steps:

   a. In the XML Structure pane, click the **XML Options** link. The XML Options dialog box is displayed as shown in Figure 4–7.

   b. If the document contains both unstructured text and XML elements, then select **Ignore Mixed Content**.

   c. To hide the XML elements in the document, select **Show placeholder text for all empty elements**.

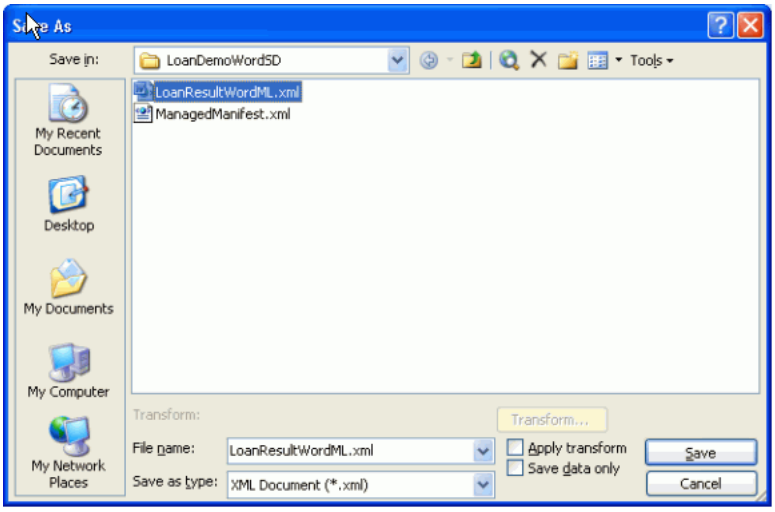
   d. If you want to save XML that does not conform to the schema, then select **Allow saving as XML even if not valid**.

   e. Click **OK**.

7. In the XML Structure pane, clear the **Show XML tags in the document** option to hide the XML structures.

8. Save the document as `LoanResultWordML.xml`, as shown in Figure 4–11.

Figure 4–11 Saving as WordML File



9. Navigate to *BPEL_ORACLE_HOME*`\integration\orabpel\samples\demos\MSOfficeIntegration\LoanDemoWordSD`, and run the following command in a command prompt window:

   ```
   wml2xslt LoanResultWordML.xml
   ```

   The XSLT for the `LoanResultWordML.xml` document is created. By default, `wml2xslt.exe` is located in the `C:\Program Files\Microsoft Office 2003 Developer Resources\` directory.

10. Copy the `LoanResultWordML.xsl` file from Step 9 to the *BPEL_ORACLE_HOME*`\integration\orabpel\samples\demos\MSOfficeIntegration\AutoLoanFlow` directory and run the `obant.bat` file from the `AutoLoanFlow` directory, as follows:

    ```
    BPEL_ORACLE_HOME\integration\orabpel\bin\obant
    ```

    You will see a `BUILD SUCCESSFUL` message in the command prompt. For example:

    ```
    BUILD SUCCESSFUL
    Total time: 15 seconds
    ```

4.3.5 Validating the Solution

The auto loan process is used to process a loan application. The user submits the loan application from a Microsoft Word document. The word document invokes a BPEL process that gets the credit rating of the loan applicant from the AutoLoanCreditRatingService process, and performs either of the following:

- If the applicant has a good credit rating, then the process sends queries to two loan processing business processes:
  - UnionLoan
  - WesternLoan

  Upon receiving replies from these two processes, the AutoLoanFlow process chooses the loan offer with the lowest APR and creates a Microsoft Word

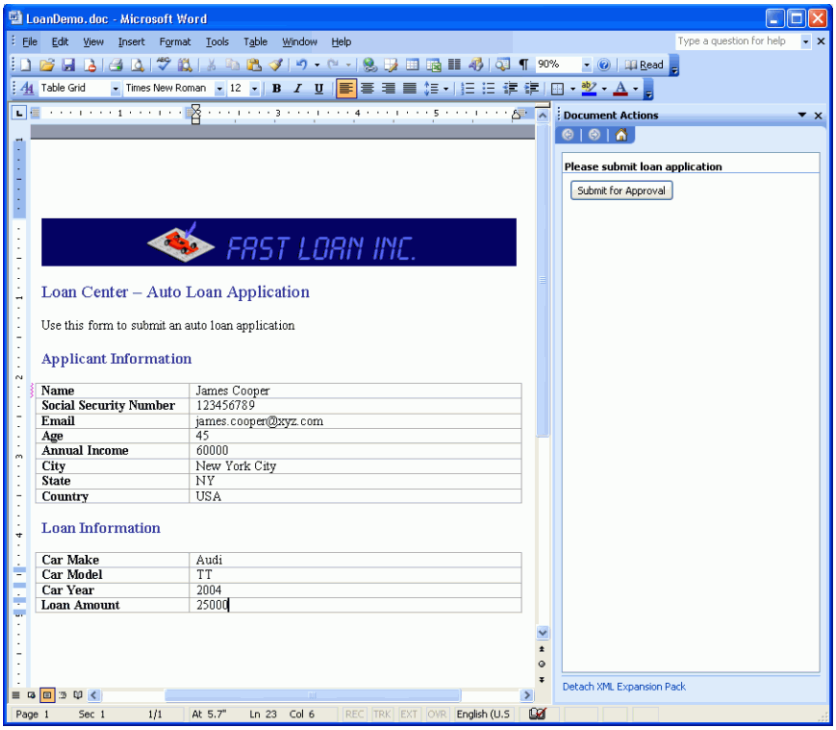
document with the loan details, and sends the document by e-mail to the loan applicant.

- If the loan applicant has a bad credit rating, the process sends an e-mail to the loan applicant with the loan application rejection message.

To test the loan application process using the smart documents created in Section 4.3.3, "Creating a Smart Document Form" and Section 4.3.4, "Creating the Microsoft Word Template for the Loan Result Notification", perform the following steps:

1. Open `LoanDemo.doc`.
2. Fill in the loan application details, as shown in Figure 4–12.

Figure 4–12 Filling In Loan Details



3. In the Task pane, click **Submit for Approval**.
4. You can monitor the BPEL process in the BPEL Console page.
5. A Loan Result document is created with the Loan Approval information, as shown in Figure 4–13.

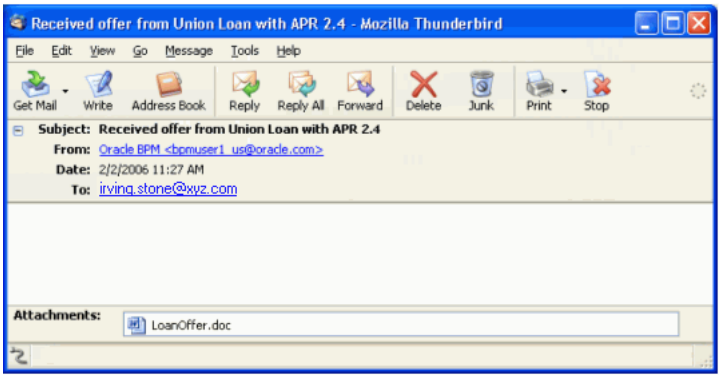
Figure 4–13 Loan Approval Information

Loan Approval Information

Loan Provider Name	Union Loan
Annual Percentage Rate	2.4

6. Log in to the Union Loan Console at `http://localhost:9700/bpelconsole`.
7. Check for e-mail messages from who was requesting the loan. A sample mail that includes a Word document as an attachment is shown in Figure 4–14.

Figure 4–14 Loan Approval Mail



4.4 Related Documentation

Refer to the following documents at the Oracle BPEL Process Manager home page at `http://www.oracle.com/technology/bpel/`:

- *BPEL: Learn by Example*
- *Quick Start Tutorial - JDeveloper 10g*
- *Quick Start Tutorial - Eclipse*

5

Completing Forms and Entering Data Using Microsoft Office

This chapter shows how to use Microsoft Office templates and Web services to create forms for entering data into enterprise applications.

This chapter contains the following sections:

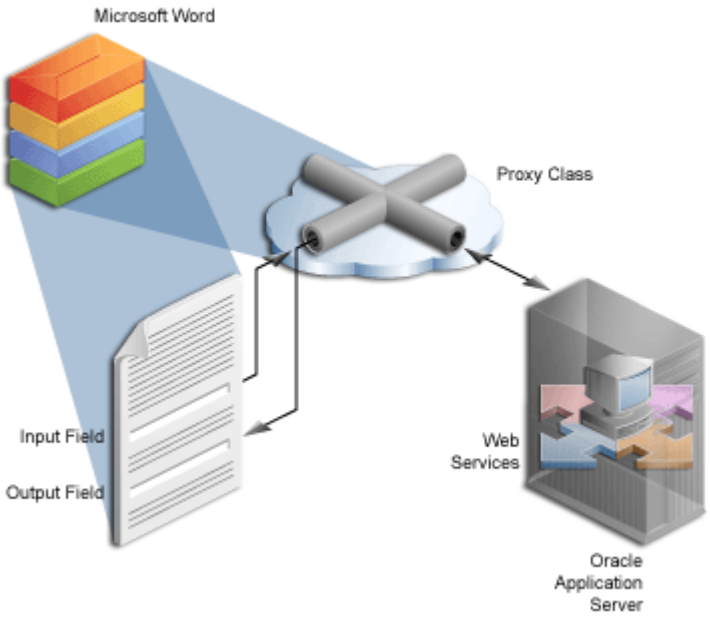
5.1 Overview

Microsoft Office 2003 Professional provides a familiar user interface for many users. Enterprise applications typically have their own specific user interface. For those types of users that only intermittently access enterprise information, Microsoft Office provides a set of facilities around smart documents (particularly templates, form fields, and Web service integration) that enable the development of Microsoft Office applications that manipulate enterprise information. Smart documents can automatically enter data, making it easier for users to complete forms or work with templates. Smart documents can also automatically access external data and place it appropriately in a document, or provide contextual help to assist users in the preparation of documents.

Oracle Application Server provides the ability to deploy and run JAX-RPC Web services that are based on Java, Enterprise Java Beans (EJB), or PL/SQL. Oracle JDeveloper provides a design-time environment for developing J2EE and Web service applications.

By following a set of guidelines discussed in this chapter, developers can use Oracle JDeveloper to create enterprise Web services that can be invoked from Microsoft Office applications, specifically Microsoft Word, Microsoft Excel, and Microsoft InfoPath (see Figure 5–1). Developers can use the Visual Basic editor included with these applications to invoke a proxy class to these Web services. This proxy class can be generated by using the Microsoft Office 2003 Web Services Toolkit as described in Section 5.2, "Prerequisites".

Figure 5–1 Manipulating Enterprise Information in Smart Documents



This chapter describes how to use Oracle JDeveloper, the Microsoft Office 2003 Web Services Toolkit, and Microsoft Office 2003 Professional to access enterprise applications.

> **Note:** These applications can also be developed by using Microsoft Visual Studio, instead of Visual Basic for Applications (VBA) code in Microsoft Word or Microsoft Excel. The steps described in this chapter (including usage of the Microsoft Office 2003 Web Services Toolkit) also apply in that context.

5.2 Prerequisites

To perform the steps outlined in this chapter, first install the following software:

- Oracle JDeveloper 10*g* Release 3 (10.1.3)

  Oracle JDeveloper contains an embedded Oracle Containers for J2EE that is sufficient for developing and testing the integration described in this chapter. For user testing and preproduction deployment, Oracle Application Server 10*g* Release 3 (10.1.3) is required.

- Microsoft Office 2003 Professional, specifically Microsoft Word 2003
- Microsoft Office 2003 Web Services Toolkit 2.01: refer to Section 1.3.3, "Microsoft Software Development Kits, Utilities, and References".
- Microsoft Internet Explorer 6.0, Mozilla Firefox 1.0, or equivalent browser
- The support files in the `fillingforms` demonstration folder. Refer to Accessing the Demonstration Support Files in the Preface for details about the demonstration support files. The support files in the `fillingforms` demonstration folder are listed and described in Table 5–1.

Table 5–1 Forms Files

Files	Description
`unionloan_banner.gif`	Banner graphic for Microsoft Word smart document and Microsoft InfoPath form. Download this file to your local file system. We recommend that you download it to: `C:\OfficeInt\samples\forms.`

5.3 Step-by-Step Procedures

This section describes two step-by-step procedures that illustrate how Microsoft Office can interoperate with Web services hosted on Oracle Application Server:

- Developing a Smart Document to Retrieve and Update Enterprise Information
- Developing a Microsoft InfoPath Form

5.3.1 Developing a Smart Document to Retrieve and Update Enterprise Information

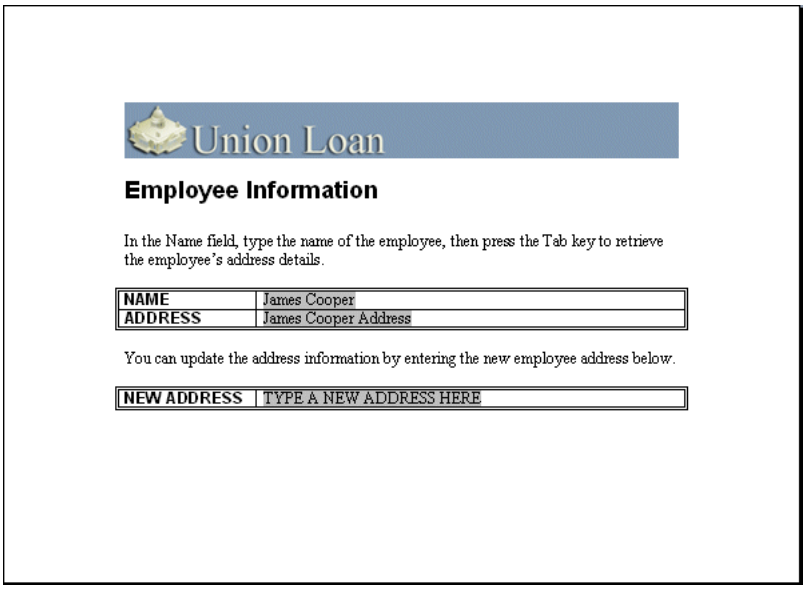
The example in this section shows how to develop a Web service in JDeveloper and integrate it with Microsoft Word by using Visual Basic for Applications code, together with a wrapper class that is generated with the Microsoft Office 2003 Web Services Toolkit. The example covers a Web service built from a simple Java class.

> **Note:** JDeveloper also offers facilities for developing EJB and PL/SQL Web services. The constraints and integration steps described subsequently also apply to these technologies.

This example is based on the custom HR enterprise application used by Union Loan. This application is used by HR and administrative personnel. Recently, the need has arisen for other users to be able to view employee addresses, and be able to update them. Rather than providing access to the (fairly complicated) HR system, the company decides to quickly develop a Microsoft Word 2003 Professional application shown in Figure 5–2, using JDeveloper to expose an existing Java implementation as a Web service that provides access to employee addresses. This Web service provides two operations in its interface: `GetAddress` and `SetAddress`.

> **Note:** The example in this section uses a Web service to access public data and therefore requires no security. If you require a more secure connection to your Web service, refer to Chapter 6, "Securing Smart Documents and Web Services".

Figure 5–2 The Microsoft Word Application



To develop a form that communicates with a Web service, perform the steps in the following sections:

- Developing a Web Service in Oracle JDeveloper
- Defining a Template Document in Microsoft Word
- Generating a Proxy Class with Microsoft Office 2003 Web Services Toolkit
- Mapping Template Fields to Web Service Parameters

There is also an optional fifth step, as shown in the following section:

- Automatically Loading and Saving Web Service Data

5.3.1.1 Developing a Web Service in Oracle JDeveloper

To create a Microsoft Office compatible Web service in Oracle JDeveloper 10*g* Release 3 (10.1.3), perform the following steps:

1. Start JDeveloper.
2. From the File menu, select **New**, select **General**, and then select **Application**.

   Tip: You may need to make sure that you select All Items from the Filter By list.

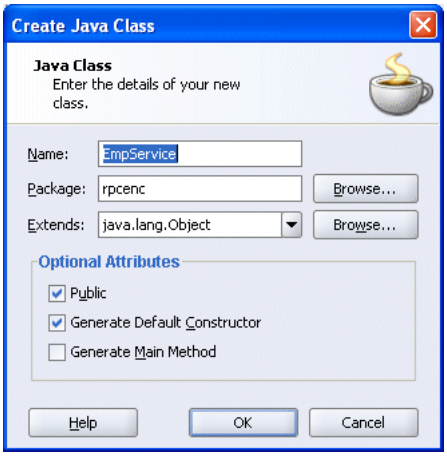
3. Click **OK.**
4. In the Application Name field, enter `MSOffice`.

   Tip: You may need to choose No Template [All Technologies] from the Application Template list.

5. Click **OK.**
6. In the Project Name field, enter `Rpc-enc`.
7. Click **OK.**

8. In the Applications Navigator, right-click the **Rpc-enc** project, and select **New** from the shortcut menu.
9. Select **General,** then select **Java Class.**
10. Click **OK.**
11. In the Name field, enter `EmpService` (see Figure 5–3).

Figure 5–3 Create Java Class Dialog Box



12. Click **OK.**

    The class is now visible in the code editor.
13. Replace the existing code with the following code sample:

Example 5–1 EmpService Java Class

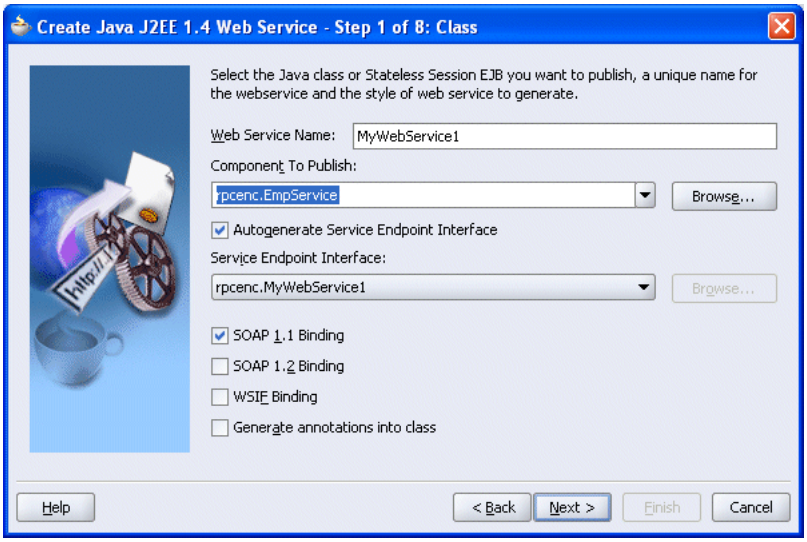
```
package rpcenc;

public class EmpService {

private String adr = null;

    public EmpService() {
    }

    public void setAddress (String address) {
        adr = address;
        return;
    }

    public String getAddress (String empno) {
        if (adr == null) {
        return empno + " Address"; }
        else return adr;
    }
}
```

This Java class declares an internal variable (`adr`) to hold the employee address data. The operation `setAddress` takes an input string and assigns it to this variable. The operation `getAddress` takes an employee number as input and returns a concatenated string consisting of the employee number and the address

string as output. In anything except this very simple example, this class would likely call a database or enterprise application API to retrieve the data.

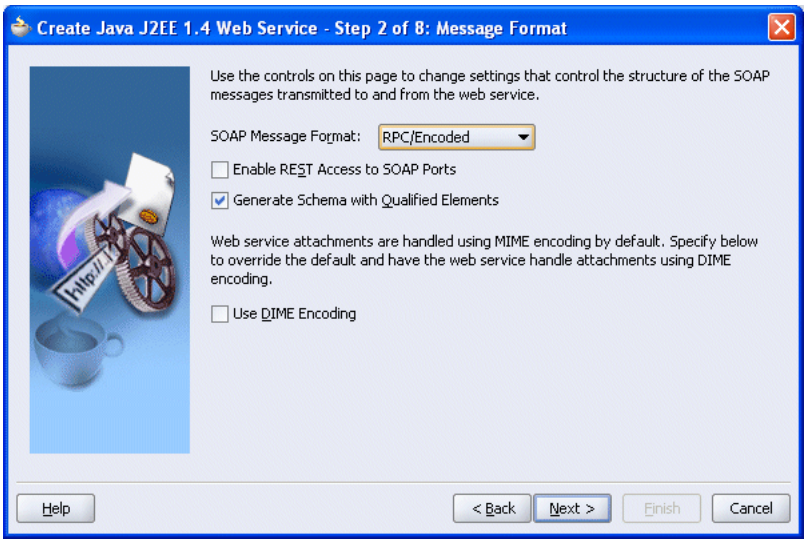
14. Save your application.
15. From the File menu, select **New**.
16. Expand the **Business Tier** node and select **Web Services,** then select **Java Web Service.**
17. Click **OK.**
18. Ensure that **J2EE 1.4 (JAX-RPC) Web Service** is selected, then click **OK.**
19. Click **Next**, if necessary, to move past the Welcome page of the wizard.
20. From the Component To Publish list, select the **EmpService** class (see Figure 5–4).

Figure 5–4 Selecting the Class



21. Click **Next.**
22. From the SOAP Message Format list, select **RPC/Encoded** (see Figure 5–5).

Figure 5–5 Specifying the Message Format



The SOAP message format can also be RPC/Literal. You can also use the document style.

> **Note:** If you are using REST for your Web service, select the Enable REST Access to SOAP Ports check box.

23. Click **Next.**
24. For this example, you do not have to specify a mapping file, so click **Next** again.
25. In the Available Methods list, select both the **getAddress** and **setAddress** methods.
26. Click **Next.**
27. Click **Next** again.
28. Select the **Stateful service** check box.

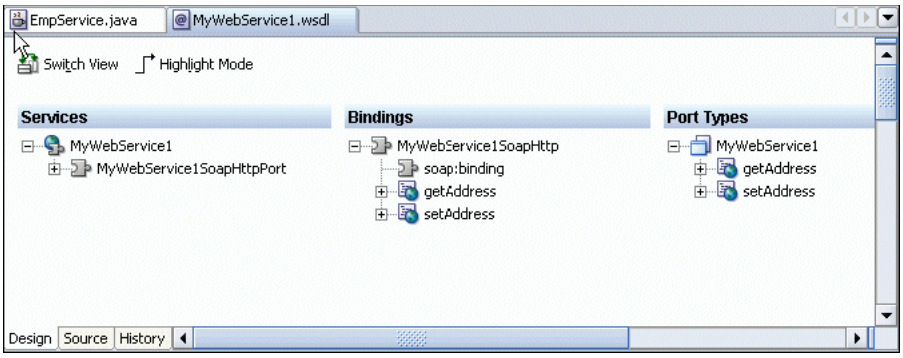
> **Note:** Web services called from Microsoft Office 2003 may be stateful or stateless. The Web service in this example is stateful.

29. From the State Scope list, select **Session**.
30. In the Session Timeout(s) field, enter `1000`.
31. Click **Finish.**

> **Note:** You can define further details for the JAX-RPC Web service, but these are optional and are not necessary for this example.

You can now see the Web Services Description Language (WSDL) document that is generated by the wizard (see Figure 5–6). In the Application Navigator, a Web Service node named MyWebService1 has been added, as well as a Java endpoint interface named `MyWebService1.java`. This file is required for JAX-RPC, but does not have to be edited for this example.

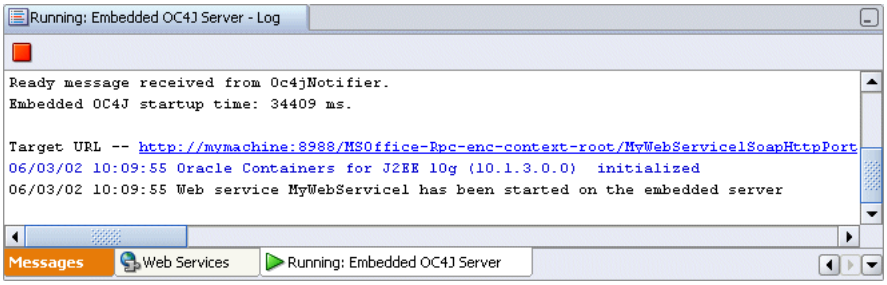
Figure 5–6 WSDL Document in JDeveloper



32. In the Application Navigator, right-click the **MyWebService1** node, and select **Run** from the menu.

JDeveloper automatically deploys the service to its embedded OC4J container and displays the message shown in Figure 5–7 in the log window.

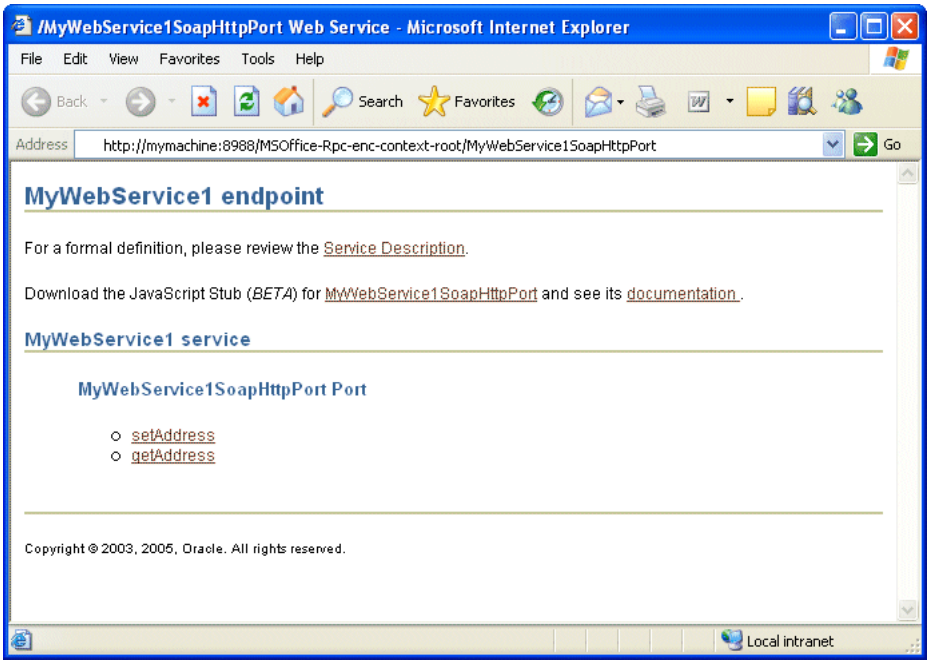
Figure 5–7 The Oracle JDeveloper Log Window



33. Click the URL in the log window.

The resulting page displays the running Web service (see Figure 5–8).

Figure 5–8 Running the Web Service



5.3.1.2 Defining a Template Document in Microsoft Word

To define a template document in Microsoft Word 2003 Professional that invokes the Web service developed in Section 5.3.1.1, "Developing a Web Service in Oracle JDeveloper", perform the following steps:

1. Start Microsoft Word.
2. From the Insert menu, select **Picture**, then select **From File**. Select the `unionloan_banner.gif` image you downloaded in Section 5.2, "Prerequisites".
3. Add the title `Employee Information` in bold font, size 18point.
4. Enter the following text to describe how to look up an employee address:

   ```
   In the Name field, enter the name of the employee, then press
   the Tab key to retrieve the employee's address details.
   ```

5. From the Table menu, select **Insert**, then select **Table**. Create a table with 2 rows and 2 columns.
6. In the first column of the first row, enter `NAME`.
7. In the first column of the second row, enter `ADDRESS` (see Figure 5–9).

Figure 5–9 Table for Looking Up an Employee's Address

NAME	
ADDRESS	

8. Enter the following text to describe how to update an employee address:

   ```
   You can update the address information by entering the new
   employee address below.
   ```

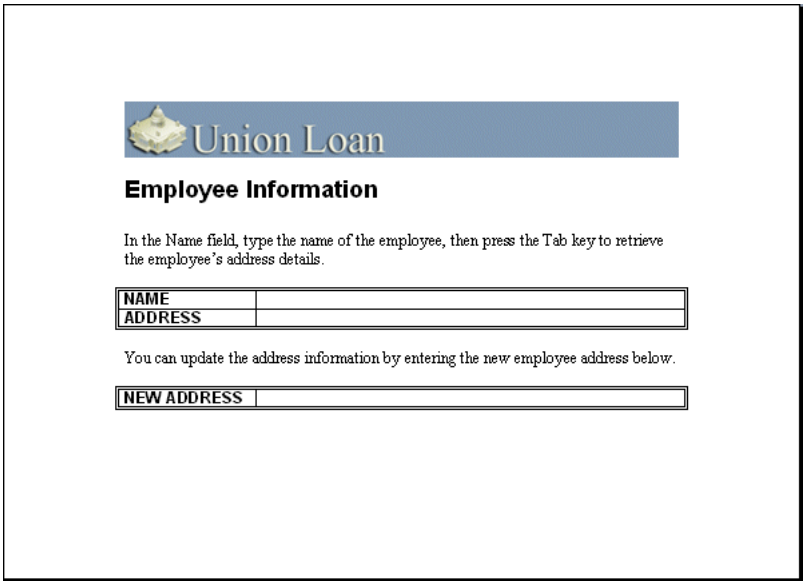
9. From the Table menu, select **Insert**, then select **Table**. Create a table with 1 row and 2 columns.
10. In the first column, enter `NEW ADDRESS` (see Figure 5–10).

Figure 5–10 Table for Specifying a New Employee Address

NEW ADDRESS	

Your document should now look something like Figure 5–11.

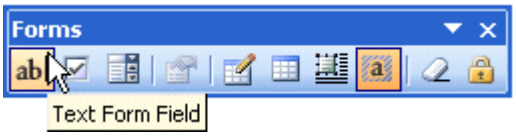
Figure 5–11 The Document Template



Next, add form fields to the document to display the live data from the Web service.

11. From the View menu, select **Toolbars**, then select **Forms**.
12. Place the cursor in the table cell that will contain the value of the employee name.
13. In the Forms Toolbar, click the **Text Form Field** button (see Figure 5–12).

Figure 5–12 The Text Form Field Icon on the Forms Toolbar



This inserts a grey area in the cell to indicate the position of the form field.

14. Place the cursor in the table cell that will contain the retrieved value of the employee address, and click the **Text Form Field** button.
15. Place the cursor in the table cell that will contain the new employee address, and click the **Text Form Field** button.

    Before you save the document, protect it to control the template definition, and to enable the form completion process for the user.

16. From the Tools menu, select **Protect Document**.
17. In the task pane that appears:
    a. In the Editing Restrictions section, select the **Allow only this type of editing in the document** check box.
    b. From the list, select **Filling in forms**.
    c. Click **Yes, Start Enforcing Protection.**
18. Enter a password to protect the document.
19. From the File menu, select **Save As**.
20. From the Save as type list, select **Document Template (*.dot)**.
21. Use the default file name.
22. Click **Save.**

    You will edit this document in Section 5.3.1.4, "Mapping Template Fields to Web Service Parameters" to connect the fields to the actual running Web service using VBA code.

5.3.1.3 Generating a Proxy Class with Microsoft Office 2003 Web Services Toolkit

To create VBA code that invokes the deployed Web service, you must use the Microsoft Office 2003 Web Services Toolkit (see Section 5.2, "Prerequisites") to generate a class that serves as a proxy, or wrapper, class to the Web service. The VBA code developed in Section 5.3.1.4, "Mapping Template Fields to Web Service Parameters" calls the proxy class generated in this section.

To generate the proxy class, perform the following steps:

1. Start Microsoft Word and open the template you created in Section 5.3.1.2, "Defining a Template Document in Microsoft Word".
2. From the Tools menu, select **Macro**, and then select **Visual Basic Editor**.

   Note: If you are using REST for your Web service:

   1. From the Tools menu of the Visual Basic Editor, select **References**.
   2. Select the Microsoft XML, v5.0 check box and click **OK.**

   You do not need to perform steps 4 through 9.

3. Make sure the Employee Information project is selected in the Project Navigator.
4. From the Tools menu of the Visual Basic Editor, select **Web Service References** to invoke the Microsoft Office 2003 Web Services Toolkit.
5. In the Microsoft Office 2003 Web Services Toolkit dialog, select the **Web Services URL** radio button.
6. In the URL field, enter the URL of the running Web service.

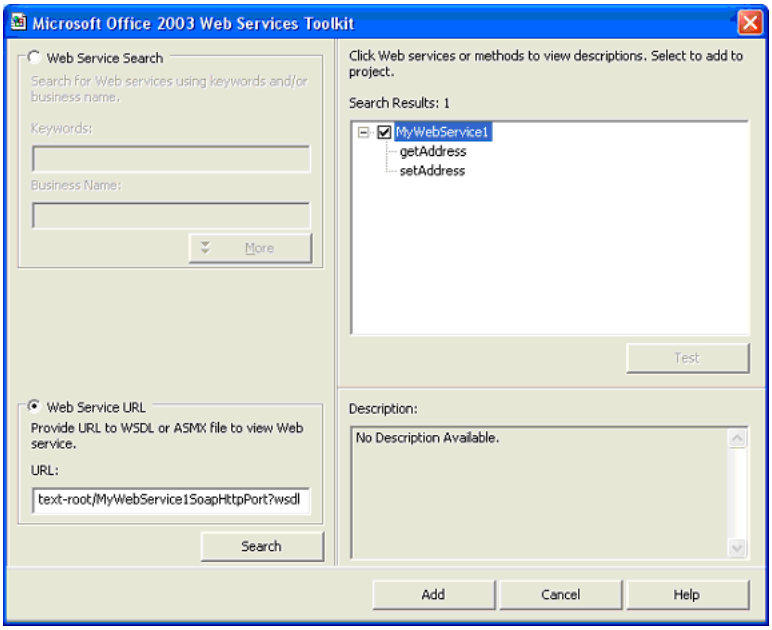
   Tip: You can copy this URL from the Address field of the browser window you opened at the end of Section 5.3.1.1, "Developing a Web Service in Oracle JDeveloper".

7. Add `?wsdl` to the end of the URL. The URL should look something like the following:

```
http://mymachine:8988/MSOffice-Rpc-enc-context-root/MyWebService1SoapHttpPort?w
sdl
```

8. Click **Search.** This adds the Web service to the top right pane (see Figure 5–13).

Figure 5–13 Selecting the Web Service



9. Select MyWebService1, then click **Add.**

   The Toolkit dialog closes and a generated class with the name `clsws_mywebservice1` is added to the Project Explorer under the Class Modules node.

10. From the Insert menu, select **Module**.

11. In the Properties window, change the name to `GetEmployeeInfo`.

12. Add the following code to the module, by entering it in the editor window:

Example 5–2 GetEmployeeInfo Module

```
Public Sub GetEmployeeInfo()

    Dim ename As String
    ename = ActiveDocument.Fields(1).Result.Text

    Dim employeeWS As clsws_MyWebService1
    Set employeeWS = New clsws_MyWebService1

    'Send the service the employee name
    Dim eadr As String
    eadr = employeeWS.wsm_getAddress(ename)
    ActiveDocument.Fields(2).Result.Text = eadr

End Sub
```

The `GetEmployeeInfo` subroutine declares an employee name string (`ename`) and assigns the value of the Name field in the document to this variable (`ename = ActiveDocument.Fields(1).Result.Text`). It then creates a reference to the

Web service and calls it. The output of the service is assigned to the Address field in the document (`ActiveDocument.Fields(2).Result.Text=eadr`).

If you are using REST the code should be as follows:

Example 5–3 GetEmployeeInfo Module for REST Services

```
Public Sub GetEmployeeInfo()

    Dim eid As String
    eid = ActiveDocument.Fields(1).Result.Text


    Dim query As String
    query =
"http://mymachine:8988/MSOffice-Rpc-enc-context-root/MyWebService1SoapHttpPort/get
Address?empno=" + eid

    'define xml and http components
    Dim queryResult As New MSXML2.DOMDocument
    Dim employeeService As New MSXML2.XMLHTTP

    'create HTTP request
    employeeService.Open "GET", query, False

    'send the request
    employeeService.send

    'parse result
    queryResult.LoadXml(employeeService.responseText)
    ActiveDocument.Fields(2).Result.Text =
queryResult.SelectSingleNode("//ns0:result").Text

End Sub
```

13. Add another subroutine with the name `SetEmployeeInfo` and enter the following code:

Example 5–4 SetEmployeeInfo Module

```
Public Sub SetEmployeeInfo()

    Dim eadr As String
    eadr = ActiveDocument.Fields(3).Result.Text

    Dim employeeWS As clsws_MyWebService1
    Set employeeWS = New clsws_MyWebService1

    'Send the service the new emp address
    employeeWS.wsm_setAddress(eadr)

End Sub
```

The `SetEmployeeInfo` subroutine declares a variable (`eadr`) and assigns it the value of the Address field in the document. It then initializes the Web service and calls the `setAddress` operation with this value.

If you are using REST, the code should be as follows:

Example 5–5 SetEmployeeInfo Module for REST Services

```
Public Sub SetEmployeeInfo()

    Dim address As String
    address = ActiveDocument.Fields(3).Result.Text

    Dim query As String
    query =
"http://mymachine:8988/MSOffice-Rpc-enc-context-root/MyWebService1SoapHttpPort/set
Address?address=" + address

    'define xml and http components
    Dim queryResult As New MSXML2.DOMDocument
    Dim employeeService As New MSXML2.XMLHTTP

    'create HTTP request
    employeeService.Open "GET", query, False

    'sent the request
    employeeService.send

End Sub
```

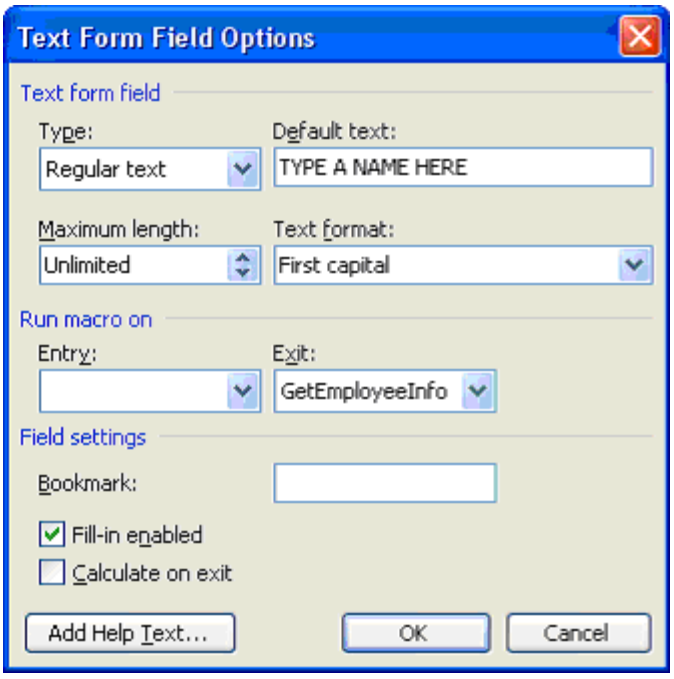
14. Save the project.
15. Exit the Visual Basic Editor and return to Microsoft Word.

5.3.1.4 Mapping Template Fields to Web Service Parameters

To link the VBA code developed in Section 5.3.1.3, "Generating a Proxy Class with Microsoft Office 2003 Web Services Toolkit" with the template document created in Section 5.3.1.2, "Defining a Template Document in Microsoft Word", perform the following steps:

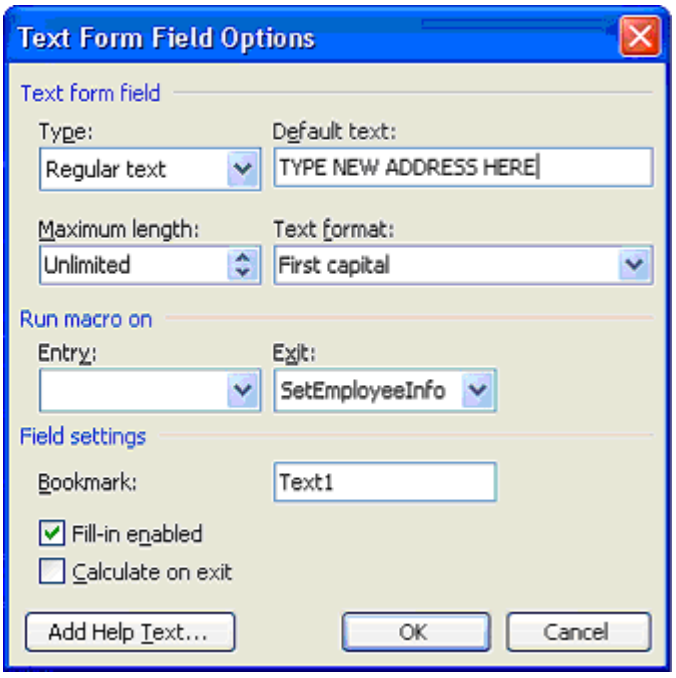
1. Start Microsoft Word.
2. Open the template you created in Section 5.3.1.2, "Defining a Template Document in Microsoft Word".
3. From the Tools menu, select **Unprotect Document**, and enter the password when prompted.
4. Right-click the form field where the user will enter a value for the employee name, and select **Properties** from the shortcut menu.
5. In the Default text field, enter `TYPE A NAME HERE.`
6. From the Text format list, select **First capital**.
7. From the Exit list, select GetEmployeeInfo (see Figure 5–14). This invokes the VBA code created in Section 5.3.1.4, "Mapping Template Fields to Web Service Parameters" when the user enters a value in the field and then leaves the field (typically by pressing the Tab key).

Figure 5–14 Options for the Employee Name Field



8. Enter similar properties for the form field where the user will update the address, but this time invoke the `SetEmployeeInfo` subroutine (see Figure 5–15).

Figure 5–15 Options for the New Address Field



9. Protect the document, as described at the end of Section 5.3.1.2, "Defining a Template Document in Microsoft Word".

10. Save and close the document.

11. In Windows Explorer, double-click the template to create a document based on the template.
12. The cursor will be placed in the first form field, so enter the name of an employee, for example, `James Cooper`.
13. Press the Tab key to move to the next field.

    The second field is populated with the value James Cooper Address.
14. Press Tab again to move to the field where you can specify a new address.
15. Enter a new address for the employee.
16. Press the Tab key again to move to the first field.

    The employee's address now reflects the new address.

5.3.1.5 Automatically Loading and Saving Web Service Data

You can automatically load and save data in your document. To do this in Microsoft Word, perform the following steps:

1. Start Microsoft Word.
2. Open the template you created in Section 5.3.1.2, "Defining a Template Document in Microsoft Word".
3. From the Tools menu, select **Unprotect Document**, and enter the password when prompted.
4. From the Tools menu, select **Macro**, and then select **Visual Basic Editor**.
5. In the Project Explorer, under your template project, double-click **ThisDocument.**
6. From the Object list, select **Document**.
7. From the Procedure list, select **Open**.

   An empty subroutine is added to the class module.
8. Add the following Visual Basic instructions to invoke the `getAddress` Web service.

Example 5–6 Invoking the getAddress Web Service

```
Private Sub Document_Open()

    Dim ename As String
    ename = ActiveDocument.Fields(1).Result.Text

    Dim employeeWS As clsws_MyWebService1
    Set employeeWS = New clsws_MyWebService1

    'Send the service the employee name
    Dim eadr As String
    eadr = employeeWS.wsm_getAddress(ename)
    ActiveDocument.Fields(2).Result.Text = eadr

End Sub
```

This code is identical to the `GetAddressInfo` subroutine you created in Section 5.3.1.3, "Generating a Proxy Class with Microsoft Office 2003 Web Services Toolkit", that is, it calls the `getAddress` Web service to retrieve the address of a given employee. Because this code is declared as an `onOpen` procedure for the

document, it will be automatically executed when the document is opened. The default value for the Name field will be passed into the Web service. The result is a default address value that is computed by adding the string address to the name. In a real application, you would not want to pass in default values. It is used in this example only for demonstrating automatic Web service invocation on opening a template document.

9. Save the project.
10. Exit the Visual Basic Editor and return to Microsoft Word.
11. Protect the document, as described at the end of Section 5.3.1.2, "Defining a Template Document in Microsoft Word".
12. Save and close the document.
13. To run the document, open the file in Microsoft Word.

    On opening, the Web service is invoked by passing in the default value for the Name field: `TYPE A NAME HERE`. In this example, the default value returned is therefore `TYPE A NAME HERE Address`.

5.3.2 Developing a Microsoft InfoPath Form

Microsoft InfoPath enables the form designer to quickly add fields to a form that are mapped to Web service invocations.

To develop a form that communicates with a Web service, perform the following tasks:

- Developing the Web Service in Oracle JDeveloper
- Defining a Form in Microsoft InfoPath

5.3.2.1 Developing the Web Service in Oracle JDeveloper

To create a Web service in JDeveloper to be used with Microsoft InfoPath, perform the following steps:

1. Follow the steps as described in Section 5.3.1.1, "Developing a Web Service in Oracle JDeveloper".
2. If you are not using Microsoft Visual Studio .NET, make the following change:

   In the Message Format step (step 2) of the Create Java J2EE 1.4 Web Service Wizard, make sure that the SOAP Message Format is set to Document/Wrapped.

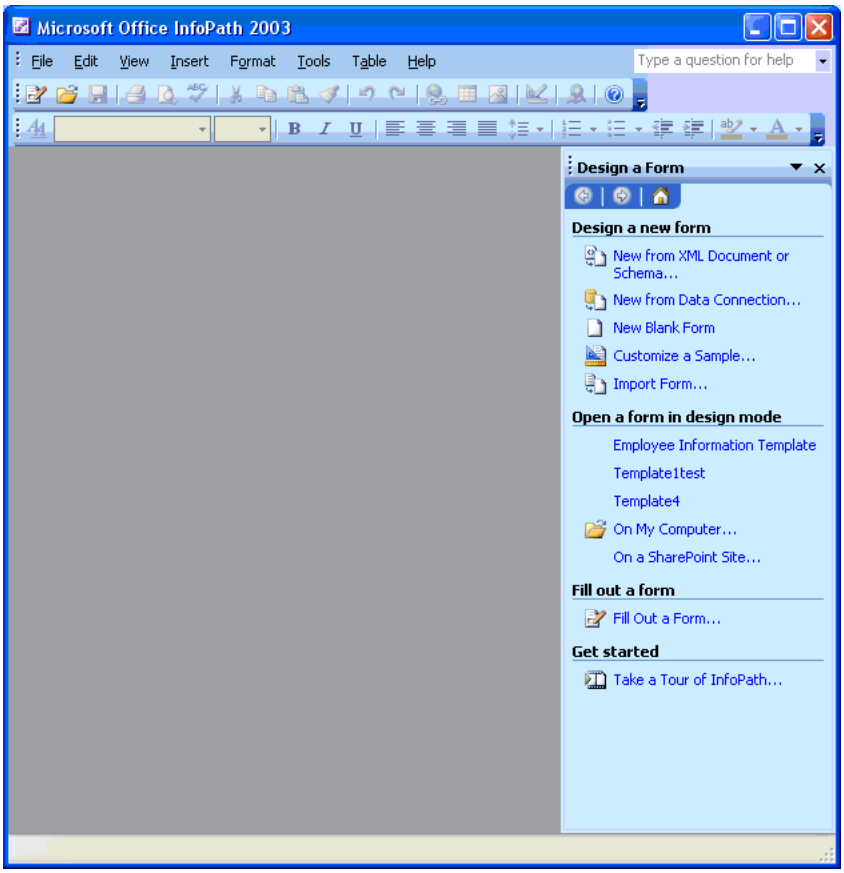
5.3.2.2 Defining a Form in Microsoft InfoPath

To define a form in Microsoft InfoPath that calls an Oracle Web service, perform the following steps:

1. Start Microsoft InfoPath.
2. On the left hand side of the dialog, select **Design a Form.**

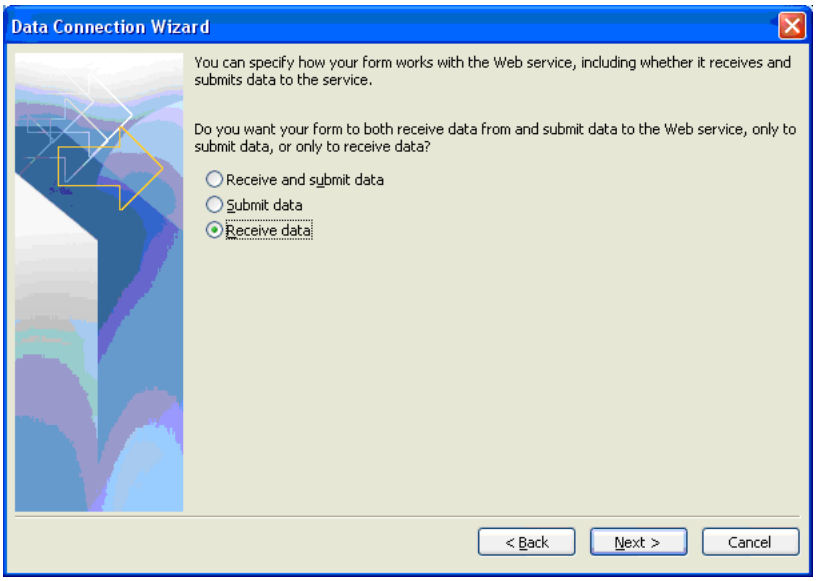
   The InfoPath main window displays as shown in Figure 5–16.

Figure 5–16 Microsoft InfoPath Main Window



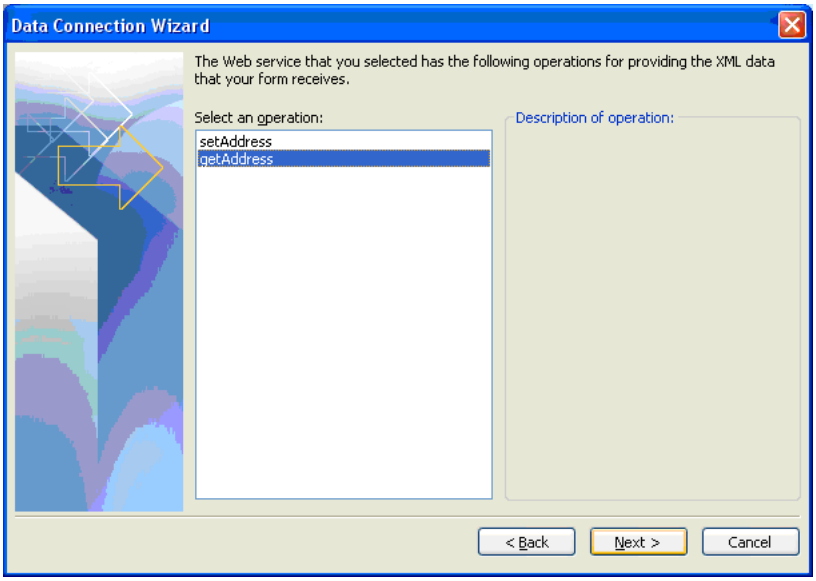
3. Under Design a new form, click **New from Data Connection.**
4. On the first step of the Data Connection Wizard, select **Web Service** as the type of connection.
5. Click **Next.**
6. Select **Receive data** (see Figure 5–17).

Figure 5–17 Receive Data from the Web Service



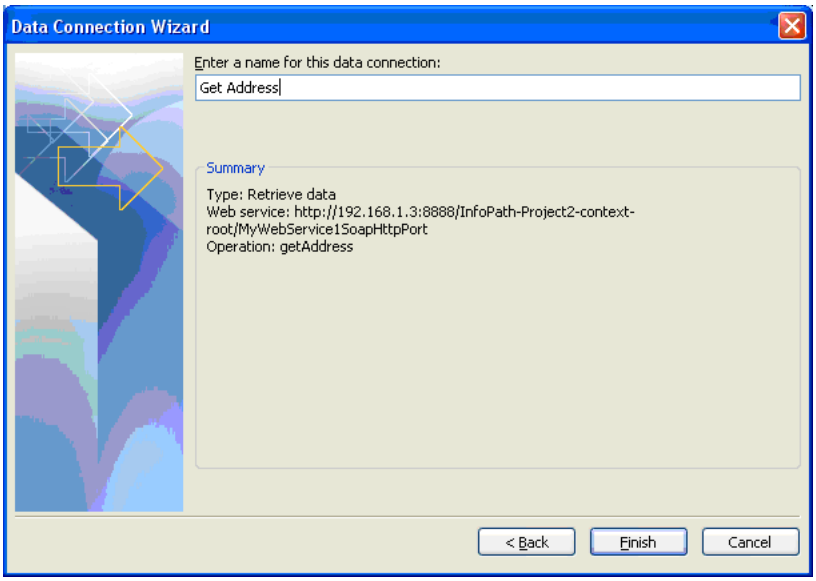
7. Click **Next.**
8. Enter the URL of the deployed Web service, adding `?WSDL` to the end.
9. Click **Next.**
10. In the Select an operation list, select **getAddress** (see Figure 5–18).

Figure 5–18 Select Web Service Operation



11. Click **Next.**
12. Enter `Get Address` as the name for the data connection (see Figure 5–19).

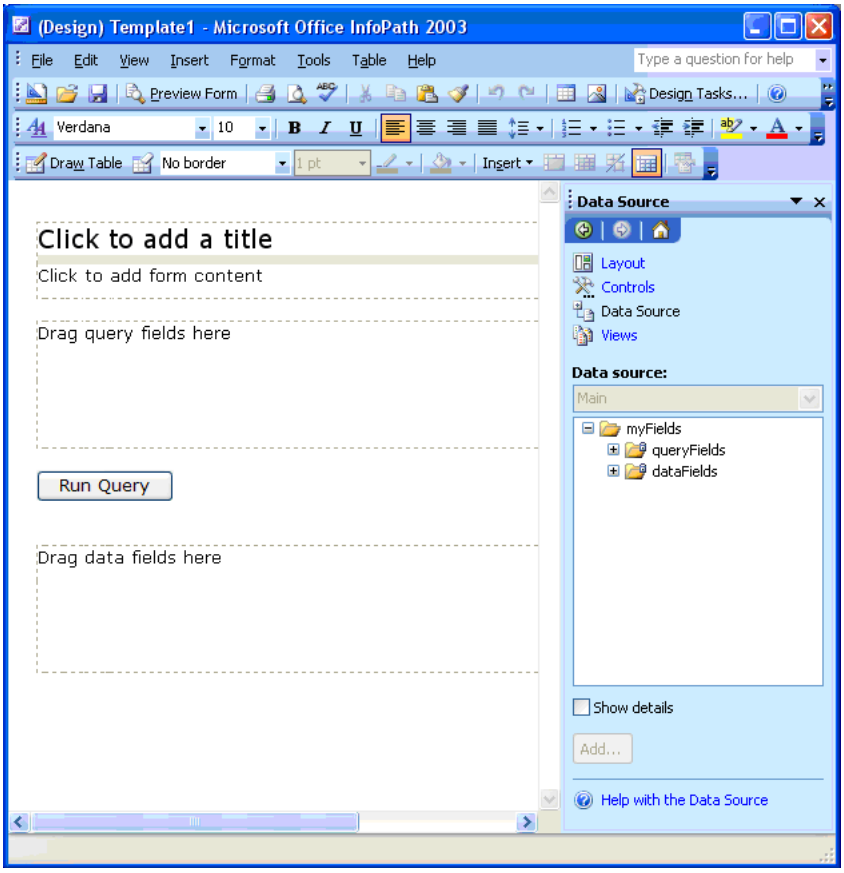
Figure 5–19 Data Connection Name



13. Click **Finish.**

Microsoft InfoPath generates a default form as shown in Figure 5–20.

Figure 5–20 Microsoft InfoPath Default Form

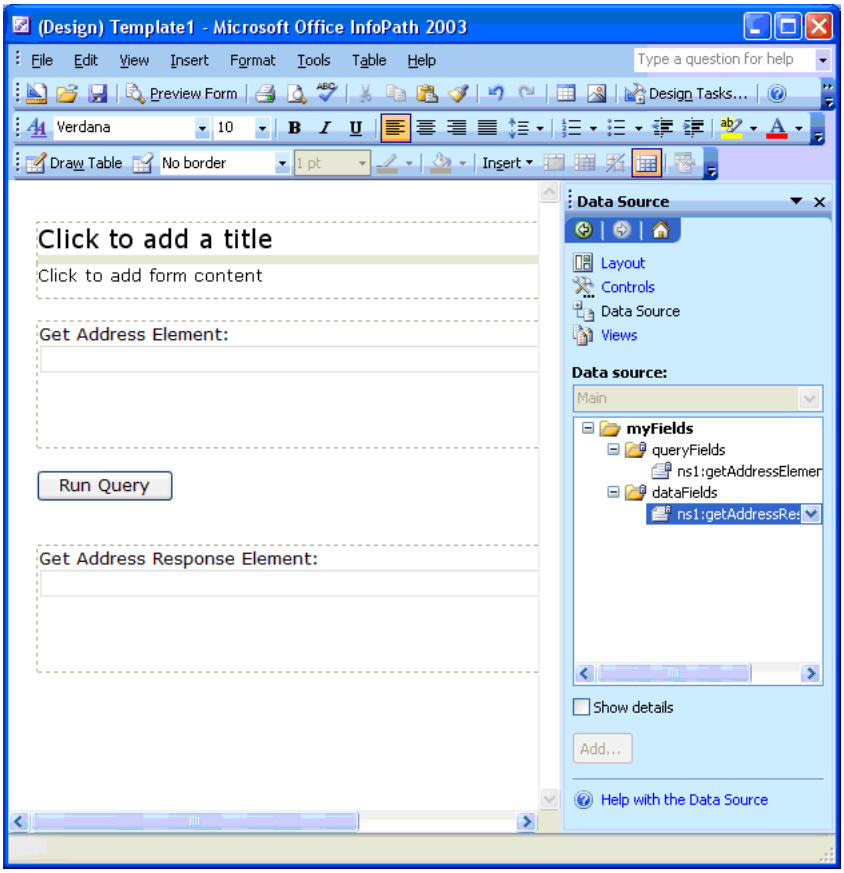


14. In the Data Source panel, expand the **queryFields** node.

15. Drag the `ns1:getAddressElement` element onto the area of the form labeled Drag query fields here, and select **Section with Controls**.
16. Expand the **dataFields** node.
17. Drag the `ns1:getAddressResponse` element onto the area of the form labeled Drag data fields here.

    The form now looks like Figure 5–21.

Figure 5–21 Form with Input and Output Fields



18. Place the cursor at the top of the form.
19. From the Insert menu, select **Picture**, then select **From File**. Select the image you downloaded in Section 5.2, "Prerequisites".
20. Change the title of the form to Employee Information.
21. Enter the following descriptive text in the area labeled Click to add form content:

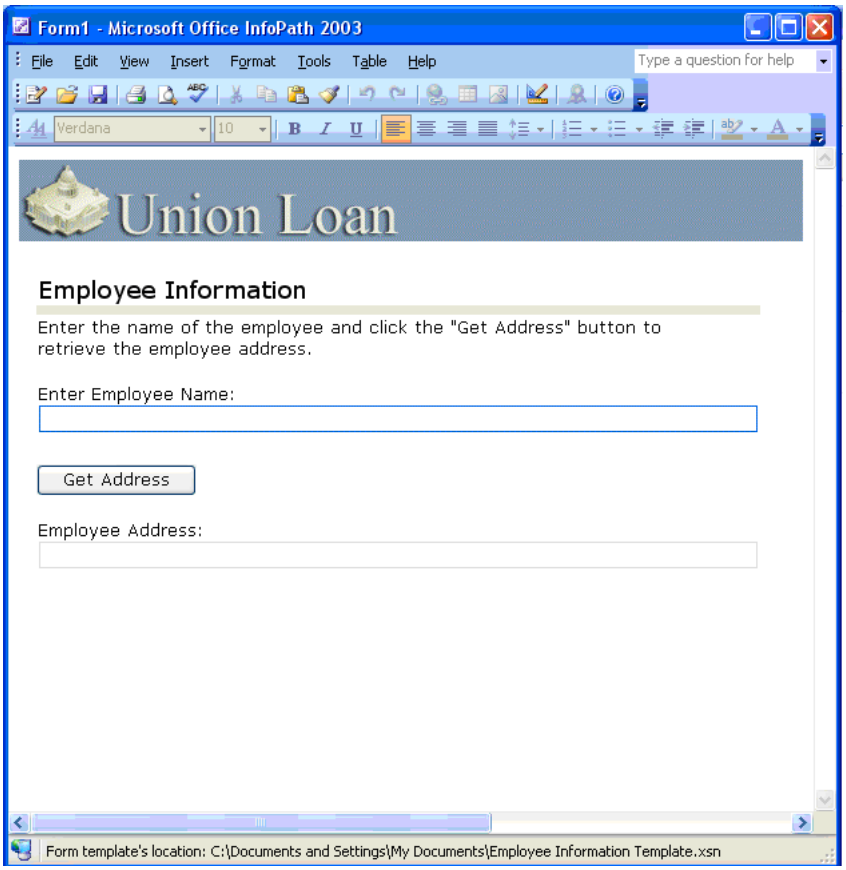
    ```
    Enter the name of the employee and click the Get Address
    button to retrieve the employee address.
    ```

22. Change the label of the input field (Get Address Element) to `Enter Employee Name.`
23. Change the label of the output field (Get Address Response Element) to `Employee Address.`
24. Double-click the **Run Query** button and change the label to `Get Address.`

25. Reduce the area of the form for the input and output fields by selecting them and dragging the outline with the mouse.

    The form should now look like Figure 5–22.

Figure 5–22 Formatted Form



26. Save the template form and close Microsoft InfoPath.
27. Open Microsoft InfoPath and select your new template form in the Fill out a Form dialog.
28. In the Enter Employee Name field, enter `James Cooper`.
29. Click **Get Address.**

    The Employee Address field is populated with the value James Cooper Address.

5.4 Troubleshooting

This section lists hints and tips that address potential issues that you might encounter.

Making sure the endpoint URL in the WSDL is pointing to the correct service

JDeveloper generates a default endpoint URL that will work with a standalone server. Since release 10.1.3, the application server to which the service is deployed will update this URL. It is therefore important to insert the WSDL reference to the running service in the Microsoft Office 2003 Web Services Toolkit and Microsoft InfoPath Data Connection Wizard. In the log window, JDeveloper will always report the URL to the

deployed service. Copy and paste this URL into dialog boxes, and append `?wsdl` to the endpoint. This can also be done in a browser to retrieve the WSDL.

Editing an existing template in Microsoft Word

To use a template, it must first be protected with a password (Tools menu, Protect Document option). When the document is opened, remember to remove the document protection (Tools menu, Unprotect Document option) first before attempting to edit the definition, otherwise edits will have unexpected results.

5.5 Related Documentation

- *Oracle Application Server Web Services Developer's Guide*

6

Securing Smart Documents and Web Services

This chapter demonstrates how you can configure smart documents to invoke enterprise Web services in a secure way. Communication from the smart document through the Web Services Proxy to the Web service and back must be secured. This requires configuration of the Web Service, the Web Service proxy, and the smart document. Additionally, it is important to digitally sign the manifest file that references the DLL file used by the smart document.

This chapter contains the following sections:

6.1 Overview

Smart documents greatly enhance the user experience of working with Microsoft Office documents. They allow automatic data entry in documents, access external data automatically and place it appropriately in a document. Smart documents can provide contextual help during the preparation of complicated documents, enabling information sharing across systems and applications.

Smart documents can communicate with Web services. If the Web service provides non-sensitive data, there is no need for securing the communication between the smart document and the Web service, but if the Web service provides sensitive and confidential data, security is a must. This chapter describes the necessary steps that must be performed to secure the communication between smart documents and Web services.

More information about smart documents and how they can be used in specific business cases is available in Chapter 4, "Creating Smart Documents That Interact with Self-Service Business Processes" and Chapter 5, "Completing Forms and Entering Data Using Microsoft Office".

6.2 Prerequisites

In order to perform the tasks outlined in this chapter, you must install the following applications and files:

- Oracle JDeveloper 10*g* Release 3 (10.1.3)

- Microsoft Visual Studio .Net 2003
- Microsoft Office 2003 Professional
- The following Microsoft SDKs and utilities:
  - Microsoft .NET Framework SDK (version 1.1 or later)
  - Microsoft Web Services Enhancements 2.0
  - Microsoft Office Smart Document SDK
  - Office 2003 Update: Redistributable Primary Interop Assemblies
  - Microsoft PKI utilities: `makecert.exe`, `pvkimprt.exe`, `cert2spc.exe`, `sn.exe`, and `signcode.exe`.
  - `xmlsign.exe` (part of Microsoft Office Smart Document SDK)

  Refer to Section 1.3.3, "Microsoft Software Development Kits, Utilities, and References"for information on obtaining these files.
- The Web service created in Chapter 5, "Completing Forms and Entering Data Using Microsoft Office".
- The support files in the `securingsmartdocs` demonstration folder. Refer to Accessing the Demonstration Support Files in the Preface for details about the demonstration support files. The demonstration files used in this chapter are described in Table 6–1.

Table 6–1 Example Smart Document Files

File	Description
`SecureSmartDocUtoken\SecureDocument.xsd`	The smart document XML Schema Definition file.
`SecureSmartDocUtoken\ManagedManifest.xml`	The XML manifest file that contains the location of the smart document's DLL.
`SecureSmartDocUtoken\SecureSmartDocument.doc`	The secure Microsoft Word smart document. **Note:** This file can also be created by following the steps given in Section 5.3.1.2, "Defining a Template Document in Microsoft Word".
`SecureSmartDocX509\SecureDocument.xsd`	The smart document XML Schema Definition file.
`SecureSmartDocX509\ManagedManifest.xml`	The XML manifest file that contains the location of the smart document's DLL.
`SecureSmartDocX509\SecureSmartDocument.doc`	The secure Microsoft Word smart document. **Note:** This file can also be created by following the steps given in Section 5.3.1.2, "Defining a Template Document in Microsoft Word".

6.3 Step-by-Step Procedures

You can use Oracle JDeveloper to expose an existing Java implementation as a Web service that provides access to confidential or restricted information. This Web service can be secured so that smart documents can safely communicate with it.

An example scenario could be the Star Loan Company that has exposed a Web service on its site to receive loan applications from people. The loan application, which is invoked by a smart document form requires a few confidential pieces of information from an applicant, like Social Security number, Annual Income, and so on. When an

applicant fills an application form and submits it to Star Loan Company, this transfer of information must be secured. A security implementation between smart documents and Web services addresses this situation appropriately.

In this chapter, we secure the sample Web service developed in Section 5.3.1, "Developing a Smart Document to Retrieve and Update Enterprise Information", as well as a smart document solution that uses a secure Web services proxy.

To develop this secure solution, perform the following steps:

- Copying the Demonstration Files
- Creating and Deploying the Web Service
- Creating the Smart Document DLL
- Attaching the XML Schema and the Expansion Pack to the Smart Document
- Securing Communication Between the Smart Document and the Web Service
- Testing the Smart Document Configuration

6.3.1 Copying the Demonstration Files

Unzip the demonstration support ZIP file to `C:\` on your computer. Refer to Table 6–1, " Example Smart Document Files" for more details.

6.3.2 Creating and Deploying the Web Service

Create a Web Service by performing the steps outlined in Section 5.3.1.1, "Developing a Web Service in Oracle JDeveloper".

6.3.3 Creating the Smart Document DLL

Create the smart document DLL using Microsoft Visual Studio .Net 2003, by performing the following steps:

1. Open Microsoft Visual Studio .Net 2003.
2. Create a new Visual C# Class Library project. Click **File**, **New**, and then **Project**.
3. In the New Project dialog box, select **Visual C# Projects** and then **Class Library**. Use the project name **SecureDoc**.
4. Right-click the project (SecureDoc) and select **Add Reference**.
5. In the Add Reference dialog box, add the following DLL files:
   - `System.dll`
   - `System.data.dll`
   - `System.Web.dll`
   - `System.Web.Services.dll`
   - `System.Windows.Forms.dll`
   - `System.Drawing.dll`
   - `System.XML.dll`
6. Add references to the `Microsoft.Office.Interop.SmartTag.dll` and `Microsoft.Office.Interop.Word.dll` files. These DLLs are located in the Global Assembly Cache (GAC) and therefore you cannot browse to them in the

Add Reference dialog box. You can add these DLLs by performing the following steps:

a. Close the SecureDoc project in Microsoft Visual Studio .Net 2003.

b. Using a text editor, open the SecureDoc project file (`SecureDoc.csproj`), in the location where you saved your project in the preceding steps.

c. Add the following lines to the file, within the <References> tag, for example:

```
<Reference
    Name = "Microsoft.Office.Interop.Word"
    AssemblyName = "Microsoft.Office.Interop.Word"
    HintPath =
"..\..\..\..\..\WINDOWS\assembly\GAC\Microsoft.Office.Interop.Word\11.0.0.0
__71e9bce111e9429c\Microsoft.Office.Interop.Word.dll"
/>

<Reference
    Name = "Microsoft.Office.Interop.SmartTag"
    AssemblyName = "Microsoft.Office.Interop.SmartTag"
    HintPath =
"..\..\..\..\..\WINDOWS\assembly\GAC\Microsoft.Office.Interop.SmartTag\11.0
.0.0__71e9bce111e9429c\Microsoft.Office.Interop.SmartTag.dll"
/>
```

d. Save and close the file.

e. Open the SecureDoc project in Microsoft Visual Studio .Net 2003.

7. Enable Web Services Enhancements for this Visual Studio project, by performing the following steps:

   a. In the Solution Explorer pane, right-click **SecureDoc**, and select **WSE Settings 2.0**.

   b. In the resulting dialog box, click the **General** tab, and select **Enable this project for Web Service Enhancements**.

   c. Click **OK**.

      This adds `Microsoft.Web.Services2.dll` to your reference list.

8. Generate the secure Web service proxy, by adding a Web reference to your `EmpService` Web service created using Oracle JDeveloper in Section 5.3.1.1, "Developing a Web Service in Oracle JDeveloper". To do this, perform the following steps:

   a. In the Solution Explorer pane, right-click **References**, and select **Add Web Reference**.

   b. In the Add Web Reference dialog box, enter the WSDL of the **EmpService** Web service. This is the URL of the WSDL file, shown at the end of the Web service creation in Oracle JDeveloper. You must append `?WSDL` to this URL.

   c. Change the Web reference name to **SecureWS**, and click **Add Web Reference**.

   d. Double-click the **SecureWS** link in the Solution Explorer pane, and expand `SecureDoc.SecureWS`. You should see a file `MyWebService1Wse`.

9. Create a class that implements the ISmartDocument interface. Right-click the project (SecureDoc), and click **Add**, and then **Add Class**. Name the class as `SecureSmartDocument.cs`. Copy the contents from Section A.6, "Contents of the SecureSmartDocument.cs File" into this file and save it.

10. Build the project. A DLL file is created in the `project_path/bin/debug` directory.
11. Copy the DLL file to the `SecureSmartDocUtoken` and `SecureSmartDocX509` demonstration support folders on your computer, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocUtoken\`.

6.3.4 Attaching the XML Schema and the Expansion Pack to the Smart Document

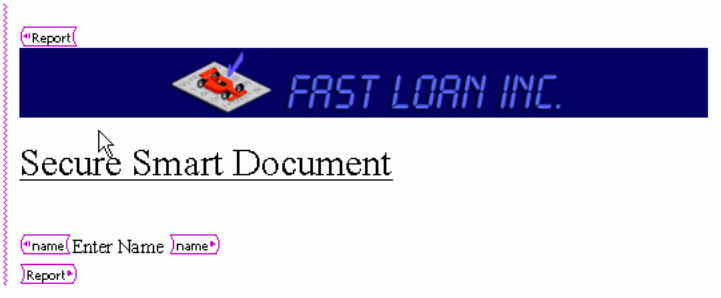
To attach the XML schema and the XML expansion pack to the smart document that you copied in Section 6.3.1, "Copying the Demonstration Files", perform the following steps:

1. Open the example document (`SecureSmartDocument.doc`) in the demonstration support folder on your computer, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocUtoken\SecureSmartDocument.doc`.
2. From the Microsoft Word menu bar, click **Tools**, and then click **Templates and Add-Ins**.
3. In the Templates and Add-ins dialog box, select the **XML Schema** tab.
4. Click **Schema Library**.
5. If there are schemas already attached, then select them and click **Delete Schema**.
6. Click **Add Schema**, and select `SecureDocument.xsd` from the demonstration support folder, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocUtoken`. Name the schema **SecureDoc**, and click **OK**.

> **Note:** Alternatively, you can create a new XSD file, copy the code from Section A.5, "Contents of the SecureDocument.xsd File" into it, and save it as `SecureDocument.xsd`. You can then add this schema.

7. Click **OK**.
8. Select **Show XML tags in the document** in the XML Structure pane.
9. Place the cursor just below the banner image, and select the **report** element from the XML Structure pane. When prompted, click **Apply to Entire Document**.
10. Select the text Enter Name and click the **name** element in the XML Structure pane. The document should now look as shown in Figure 6–1.

Figure 6–1 XML Structure of SecureSmartDocument.doc



11. Save the document.

To attach the XML expansion pack to the smart document that you created in the preceding steps, perform the following steps:

- Attaching the XML Expansion Pack
- Enabling Manifest Security Check
- Signing the Manifest Using XMLSign.exe

6.3.4.1 Attaching the XML Expansion Pack

To attach the XML expansion pack, perform the following steps:

1. Create an XML file called `ManagedManifest.xml`, and add the code from Example A–3, "ManagedManifest.xml for Chapter 6" to this file. Alternatively, you can use the `ManagedManifest.xml` file provided in the `securingsmartdocs\SecureSmartDocUtoken` demonstration folder.
2. Attach the XML expansion pack to the document. To do this, perform the following steps:
   a. From the Microsoft Word menu bar, click **Tools**, and then click **Templates and Add-Ins**.
   b. In the Templates and Add-ins dialog box, select the **XML Expansion Packs** tab.
   c. Click **Add** and select `ManagedManifest.xml` from the demonstration support folder, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocUtoken\ManagedManifest.xml`.

6.3.4.2 Enabling Manifest Security Check

To enable manifest security check of the smart document manifest file, perform the following steps:

1. Digitally sign the XML Expansion Pack with a trusted certificate using the XML Expansion Pack Signing Utility that comes with the Microsoft Office Smart Document SDK.
2. Create a trusted certificate for signing using the `makecert` utility, by running the following command:

   ```
   makecert -r -n "CN=mansign" -sv mansign.pvk mansign.cer
   ```

   `makecert.exe` is part of the Microsoft Visual Studio .Net 2003 installation. You can skip Step 2 through Step 6 if you have an existing trusted certificate.
3. Copy the certificate file (`mansign.cer`) and the private key file (`mansign.pvk`) to the demonstration support files directory, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\certs`.
4. Install the certificate in the personal store by performing the following steps:
   a. Right-click the `.cer` file and select **Install Certificate**. The Certificate Import Wizard is displayed.
   b. Click **Next**.
   c. In the Certificate Store wizard dialog box, select **Place all certificates in the following store** and browse to the Personal store, and click OK.

d. Click **Next**, and then click **Finish**.

5. Verify that your certificate was stored successfully. See "Step 3: Checking If Your Certificate Was Stored Successfully". In the last step, verify that a certificate with the alias **mansign** exists.

6. To make this certificate trusted, perform the following steps:

   a. In Windows, click **Start**, and then click **Run**.

   b. Enter `mmc`, and click **OK**. This starts the Microsoft Management Console (MMC) tool.

   c. Click **Console Root**, **Certificates**, and then click **Trusted Root Certification Authorities**.

   d. Right-click **Certificates**, **All Tasks** and click **Import**.

   e. In the Certificate Import Wizard, select **mansign.cer**.

   f. Click **Next** and then click **Finish**.

6.3.4.3 Signing the Manifest Using XMLSign.exe

To sign the manifest using `XMLSign.exe`, perform the following steps:

1. Navigate to the `C:\Program Files\Microsoft Office 2003 Developer Resources\Microsoft Office 2003 Smart Document SDK\Tools` directory.

2. Run the following command:

   ```
   xmlsign.exe -c mansign.cer -v mansign.pvk ManagedManifest.xml
   ```

   Where `ManagedManifest.xml` is the XML expansion pack of your smart document. `XMLSign.exe` is part of the Microsoft Office 2003 Smart Document SDK.

3. Use the `sn` utility to compile the managed Smart Document assembly with a Strong Name, which consists of a simple text assembly name, a version number, culture information, public key, and a digital signature. Run `sn` as shown here:

   ```
   sn -k PSS_SmartDoc.snk
   ```

   `sn.exe` is part of the Microsoft Visual Studio .Net 2003 installation.

4. In the `AssemblyInfo.cs` file of the SecureDoc visual project, set the `AssemblyKeyFile` attribute to point to the Strong Name key, as shown here:

   ```
   [assembly: AssemblyDelaySign(false)]
    [assembly: AssemblyKeyFile("C:\demos\certs\PSS_SmartDoc.snk")]
     [assembly: AssemblyKeyName("")]
   ```

   Provide the absolute path to the `PSS_SmartDoc.snk` file.

5. Rebuild your smart document project.

6. Digitally sign the strong-named assembly, using the `signcode` utility. To do this, perform the following steps:

   a. Create an SPC file for signing, using `cert2spc.exe`, as follows:

      ```
      Cert2spc mansign.cer mansign.spc
      ```

b. Using the `signcode` utility located at `C:\Program Files\Microsoft.NET\SDK\v1.1\Bin`, sign the smart document DLL, as follows:

```
signcode /spc mansign.spc /v mansign.pvk
C:/microsoft-interoperability-guide-demo-support/securingsmartdocs/SecureSmartDocUtoken/SecureDoc.dll
```

Where `SecureDoc.dll` is the assembly for the secured document. `signcode.exe` is part of the Microsoft Visual Studio .Net 2003 installation.

6.3.5 Securing Communication Between the Smart Document and the Web Service

This section describes how you can secure communication between smart documents and Web services. This chapter describes two methods for securing communication between smart documents and Web services - Username token and X.509 token. This section contains the following subsections:

- Securing the Web Service Proxy and the Web Service Using Username Token
- Securing the Web Service Proxy and the Web Service Using X.509 Token
- Securing the Web Service using OWSM Gateway
- Integrating with Oracle Identity Management

6.3.5.1 Securing the Web Service Proxy and the Web Service Using Username Token

You can use the username token to propagate user credentials to the Web service. If you choose this approach, users must perform authentication by entering their user name (optional) and password before using the Web service from within the smart document. The following two sections describe the steps to configure username token authentication:

- Securing the Client Side
- Securing the Web Service on the Server Side

6.3.5.1.1 Securing the Client Side

When giving access to confidential information, you must ensure that users are authenticated when they access a secure Web service. An authentication dialog box must be displayed to the user for entering valid credentials. To secure the client side, perform the following steps:

1. Open Microsoft Visual Studio .Net 2003.
2. Create a username token dialog class and add it to the visual project created in Section 6.3.3, "Creating the Smart Document DLL". Select the SecureDoc project in the Solution Explorer pane, and click **Add**, **Add Class**, and name it `UsernameTokenDialog.cs`.
3. Copy the contents from Section A.7, "Contents of the UsernameTokenDialog.cs File" into this file.
4. Integrate the smart document with the Web service proxy and pass the user name and password extracted in the previous step.

   In Microsoft Visual Studio .Net 2003, add the code shown in Example 6–1 to the `onTextboxContentChange()` method in your smart document class created in Section 6.3.3, "Creating the Smart Document DLL".

Example 6–1 Code to Add to the onTextboxContentChange() Method (Username Token)

```
{

   if (Value.Length > 0)
   {
      String subName = Environment.UserName;

      UsernameTokenDialog dialog = new UsernameTokenDialog();
      String uname = null;
      String pwd = null;

      dialog.setDefaultUsername(subName);

      if (dialog.ShowDialog() == DialogResult.OK)
      {
         uname = dialog.getUsername();
         pwd = dialog.getPassword();
      }

      if (uname == null || pwd == null)
      {
         System.Windows.Forms.MessageBox.Show("Missing username / password ");
      }
      else
      {

         SecureDoc.SecureWS.MyWebService1Wse proxy = new
SecureDoc.SecureWS.MyWebService1Wse();

         UsernameToken utoken = new UsernameToken(uname, pwd,
PasswordOption.SendPlainText);

         // Add the UsernameToken token to the SOAP message.
         proxy.RequestSoapContext.Security.Tokens.Add(utoken);

         String res = proxy.getAddress(uname);

         Console.WriteLine("response : " + res);
         Console.ReadLine();

         System.Windows.Forms.MessageBox.Show(res);

         String result = res;

         Microsoft.Office.Interop.Word.Range objRange =
(Microsoft.Office.Interop.Word.Range)Target;
         objRange.InsertAfter(result);

      }
   }
}
```

5. Rebuild the project to generate an updated DLL, and copy it to `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocUtoken`.

6.3.5.1.2 Securing the Web Service on the Server Side

To secure the Web service on the server side, perform the following steps:

1. Open the Web service generated using Oracle JDeveloper.
2. Right-click **MyWebService1** and select **Secure Web Service**. This displays a wizard.
3. Select authentication option as **Text Password**.
4. Select the **Authentication** tab under Security, and select **Expect Username token to Authenticate**.
5. In the Application Navigator, right-click the **MyWebService1** node, and select **Run**. Oracle JDeveloper automatically deploys the service to its embedded OC4J container.
6. Copy the URL in the console window, paste it in the Address field in a new Internet Explorer window, and press **Enter**.

   The resulting page displays the running Web service.
7. Add a user entry in the `system-jazn-data.xml` file by performing the following steps:
   a. Stop the embedded OC4J server by clicking **Run**, **Terminate**, and then **Embedded OC4J Server**.
   b. Navigate to the *JDEV_HOME*`/Jdev/mywork/MSOffice` directory and open the `MSOffice-jazn-data.xml` file for editing.
   c. Enter the following user entry under the `<users>` tag:

```
<user>
    <name>jcooper</name>
    <display-name>Smart document user </display-name>
    <description>Smart document user</description>
    <credentials>!password</credentials>
</user>
```

   Note: The user name and password should be the same username and password that the client sends.
8. Restart the OC4J Server.
9. Select **MyWebService1** and click **Run**.

6.3.5.2 Securing the Web Service Proxy and the Web Service Using X.509 Token

You can use X.509 token for propagating the user credentials to the Web service. If you choose this approach, a Public Key Infrastructure (PKI) trust has to be set up before exchanging the X.509 token. The following sections describe the steps to configure X.509 token authentication:

- Generating and Deploying Public and Private Keys
- Securing the Client Side
- Securing the Web Service on the Server Side

6.3.5.2.1 Generating and Deploying Public and Private Keys

Public Key Infrastructure (PKI) enables an organization to secure its communications and business transactions by using digital certificates that are exchanged between authenticated users and trusted resources. A private key is required for the current user that is used to sign the client message, and a public key is required that is used by

the OC4J Web service for verifying signature and asserting identity. The procedure for deploying PKI to secure smart documents includes the following high-level steps:

- Step 1: Creating X.509 Certificate and Corresponding Private Key File
- Step 2: Installing the Private Key Certificate
- Step 3: Checking If Your Certificate Was Stored Successfully
- Step 4: Using X.509 Certificate Tool for Viewing the Certificate
- Step 5: Importing the Public Key Certificate for Verifying Signature

Step 1: Creating X.509 Certificate and Corresponding Private Key File

To create an X.509 certificate and corresponding private key (pvk) file, perform the following steps:

1. Run the `makecert` utility as follows:

   ```
   makecert -r -n %alias% -sv  %pvkfilename%  %cerfilename%
   ```

   where `alias` is the alias of your certificate. For example, if the identity of the user logging in to the Windows system is `jcooper`, then create a certificate with alias `CN=jcooper`.

   For example:

   ```
   makecert -r -n "CN=jcooper" -sv  jcooper.pvk jcooper.cer
   ```

2. In the Create Private Key Password dialog box, specify and confirm the password as `oc4jnetsign`, and click **OK**.
3. Create the SPC file needed to create the PFX file by running the following command:

   ```
   Cert2spc.exe jcooper.cer jcooper.spc
   ```

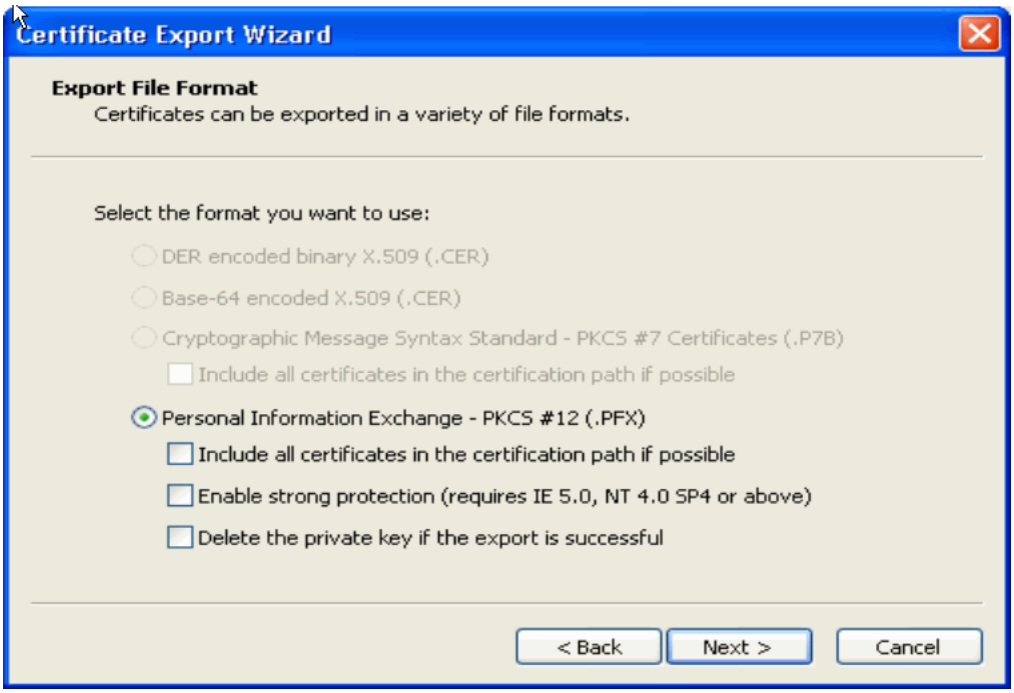
   This creates a `jcooper.spc` file.

4. Create the P12 file from the PVK and SPC files by running the following command:

   ```
   Pvkimprt.exe -pfx jcooper.spc jcooper.pvk
   ```

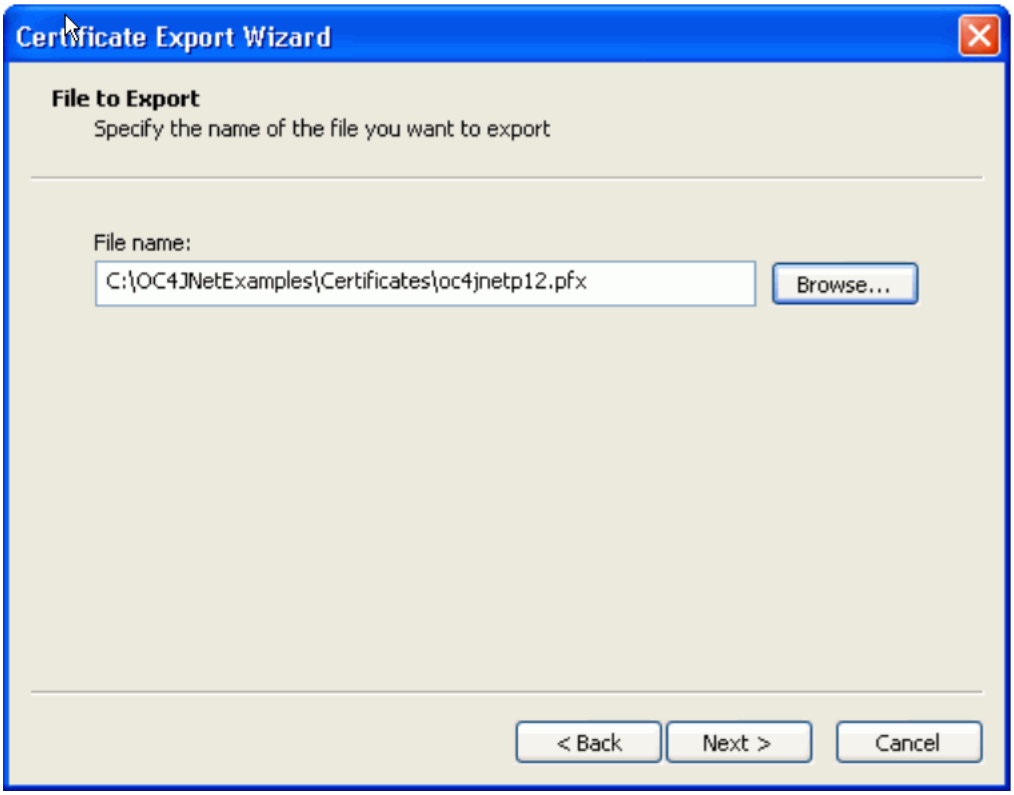
5. In the Enter Private Key Password dialog box, enter the `oc4jnetsign` password, and click **OK**.
6. The Certificate Export Wizard is displayed, which starts creating the PFX file from the certificate and the associated private key for this. Click **Next**.
7. Click **Yes** to export the private key with the certificate.
8. In the Export File Format wizard dialog box, shown in Figure 6–2, ensure that the **Enable strong protection** option is not selected, and click **Next**.

Figure 6–2 Export File Format



9. In the Password wizard dialog box, enter the password that you specified earlier (`oc4jnetsign`), confirm the same, and click **Next**.
10. In the File to Export wizard dialog box, shown in Figure 6–3, specify the file name as `oc4jnetp12.pfx` and store it in the `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocX509\Certificate` folder. Click **Next**.

Figure 6–3 File to Export



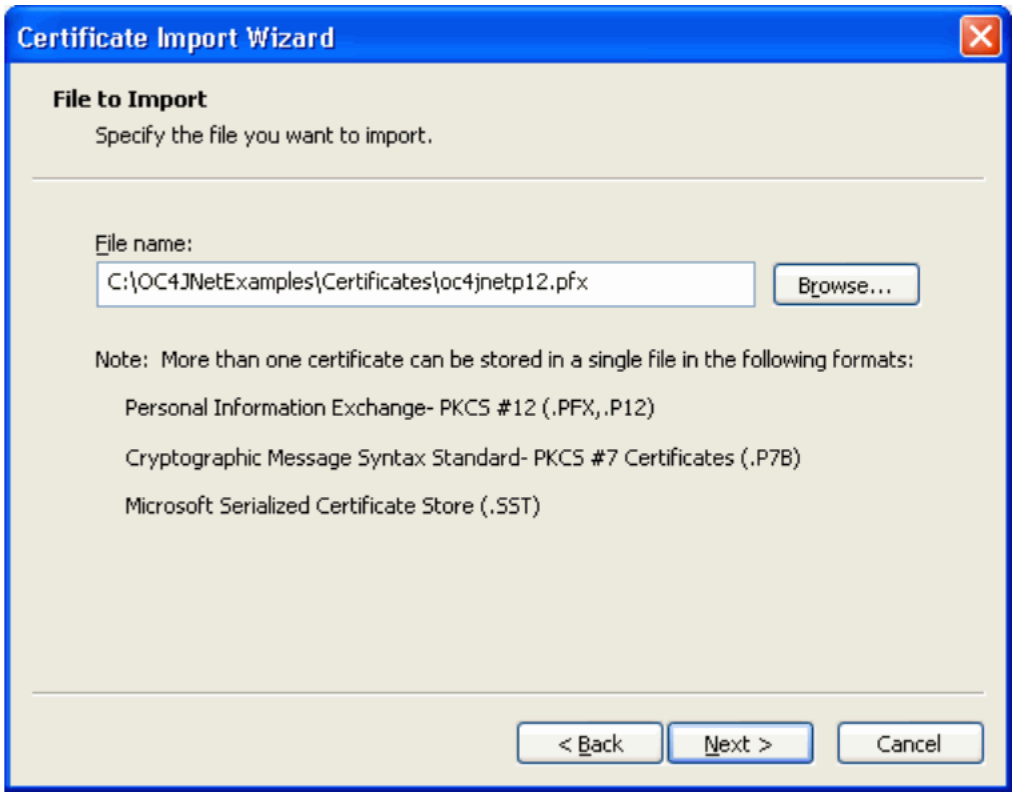
11. In the Certificate Store wizard dialog box, select **Place all certificates in the following store**. Click **Browse** and select **Personal**.
12. In the Completing the Certificate Export Wizard dialog box, click **Finish**.

Step 2: Installing the Private Key Certificate

To install the private key certificate in the Windows key store, perform the following steps:

1. Right-click `oc4jnetp12.pfx` and click **Install PFX**. This displays the Certificate Import Wizard.
2. Click **Next**. The File to Import wizard dialog box shows the `oc4jnetp12.pfx` file already selected, as shown in Figure 6–4.

Figure 6–4 File to Import



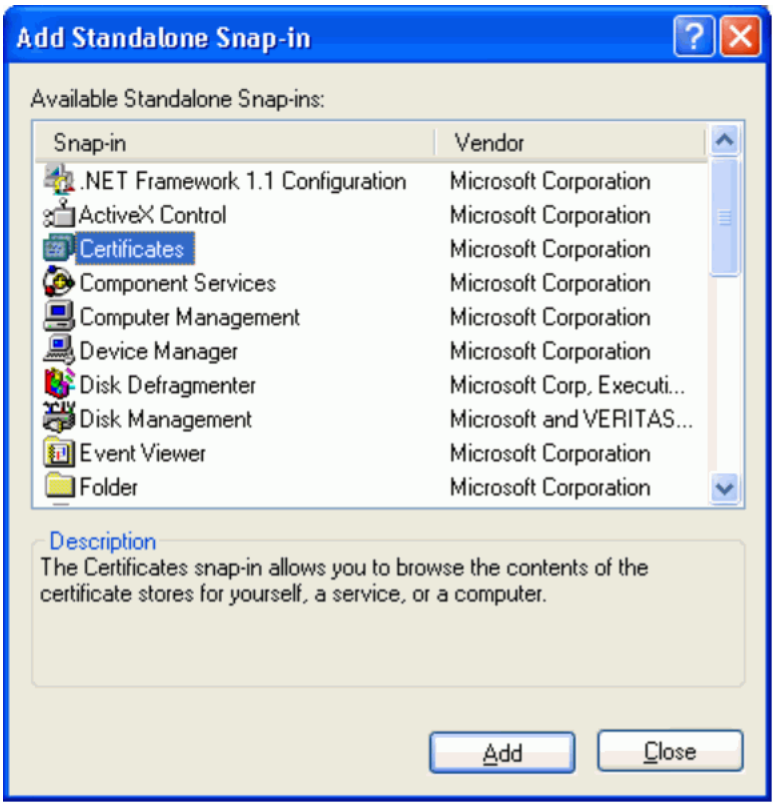
3. Click **Next**.
4. In the Password wizard dialog box, enter the password that you specified when creating the certificate (`oc4jnetsign`), and then click **Next**.
5. In the Certificate Store wizard dialog box, specify the certificate store to be **Personal**, and click **Next**.
6. In the Completing the Certificate Import Wizard dialog box, click **Finish**.

Step 3: Checking If Your Certificate Was Stored Successfully

To check if your certificate was stored successfully, perform the following steps:

1. In Windows, click **Start**, and then click **Run**.
2. Enter `mmc`, and click **OK**. This starts the Microsoft Management Console (MMC) tool.
3. Click **Console**, and then click **Add/Remove Snap-in**.
4. In the Standalone tab, click **Add**.
5. In the Add Standalone Snap-in dialog box, shown in Figure 6–5, double-click **Certificates**.

Figure 6–5 Add Certificates Snap-In



6. In the Certificates snap-in dialog box, select **My user account**, and then click **Finish**.
7. In the Add Standalone Snap-in dialog box, click **Close**.
8. In the Add/Remove Snap-in dialog box, click **OK**.
9. Click **Console Root**, **Certificates**, **Personal** and then **Certificates**. In this example, the alias name should be `jcooper`.

Step 4: Using X.509 Certificate Tool for Viewing the Certificate

To see the key identifier of a particular certificate, you must use the X.509 Certificate Tool that is installed along with Web Service Enhancements 2.0. To view the certificate, perform the following steps:

1. Start the X.509 Certificate Tool. Click **Start**, **Programs**, **Microsoft WSE 2.0**, and then **X509 Certificate Tool**.
2. Specify **Personal** as the **Store Name**.
3. Click **Open Certificate**, and select **jcooper**.

Step 5: Importing the Public Key Certificate for Verifying Signature

To import the public key certificate from `oc4jnet12.pfx` to a Java keystore for verifying the signature, use the `keytool` utility as follows:

```
keytool -import -alias jcooper -file jcooper.cer -keypass jcooper -keystore
myks.jks -storepass password
```

6.3.5.2.2 Securing the Client Side

To secure the client side, perform the following tasks:

1. Generate the private key and public key certificate for the Windows user as described in Section 6.3.3, "Creating the Smart Document DLL".
2. Integrate the smart document with the Web service proxy and pass the certificate created in Section 6.3.3, "Creating the Smart Document DLL".

   In Microsoft Visual Studio .Net 2003, add the code shown in Example 6–2 to the `onTextboxContentChange()` method in your smart document class created in Section 6.3.3, "Creating the Smart Document DLL":

Example 6–2 Code to Add to the onTextboxContentChange() Method (X.509 Token)

```
{
   if (Value.Length > 0)
   {
      String subName = Environment.UserName;

      SecureDoc.SecureWS.MyWebService1Wse proxy = new
SecureDoc.SecureWS.MyWebService1Wse();

      X509SecurityToken signtoken =
RetrieveTokenFromStoreUsingSubName(subName);

      if (signtoken == null)
      {
         throw new ApplicationException("Unable to obtain Sign security
token.");
      }

      //Add the X.509 token
      proxy.RequestSoapContext.Security.Tokens.Add(signtoken);

      //Sign the Body
      proxy.RequestSoapContext.Security.Elements.Add(new
MessageSignature(signtoken));

      String res = proxy.getAddress(uname);
      Console.WriteLine("response : " + res);
      Console.ReadLine();

      System.Windows.Forms.MessageBox.Show(res);

      String result = res;

      Microsoft.Office.Interop.Word.Range objRange =
(Microsoft.Office.Interop.Word.Range)Target;
      objRange.InsertAfter(result);
   }
}
public static X509SecurityToken RetrieveTokenFromStoreUsingSubName(string
subName)
{
   // Open the CurrentUser Certificate Store and try MyStore only
   X509CertificateStore store =
X509CertificateStore.CurrentUserStore(X509CertificateStore.MyStore);
   X509SecurityToken token = null;
   try
```

```
   {
      if (store.OpenRead())
      {
         String modSubName = "CN="+subName;
         Console.WriteLine("Find certificate with Subject Name : " +
modSubName);
         //Find certificate by Subject Name
         X509CertificateCollection certs =
store.FindCertificateBySubjectName(modSubName);
         if (certs.Count > 0)
         {
            // Get the first certificate in the collection
            token = new X509SecurityToken(((X509Certificate)certs[0]));
         }
      }
   }
   finally
   {
      if (store != null)
      {
         store.Close();
      }
   }
   return token;
}
```

3. Rebuild the project to generate an updated DLL, and copy it to the demonstration support folder, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocX509`.

6.3.5.2.3 Securing the Web Service on the Server Side

To secure the Web service on the server side, perform the following steps:

1. Open the Web service generated using Oracle JDeveloper.
2. Right-click **MyWebService1** and select **Secure Web Service**. This displays a wizard.
3. Select authentication option as **X.509 Digital Certificate**.
4. Edit the keystore options and configure it to use **myks.jks** that was created earlier.

   The keystore password is `password`.

6.3.5.3 Securing the Web Service using OWSM Gateway

The Web service can also be secured using OWSM Gateway. Refer to the Web Services Manager page on Oracle Technology Network (OTN), at http://www.oracle.com/technology/products/webservices_manager/index.html.

6.3.5.4 Integrating with Oracle Identity Management

This example illustrates integration with `system-jazn-data.xml`, which is a lightweight XML repository for storing user and role information. If you have Oracle Identity Management (OID/SSO, COREid) and you want to integrate your Web service with Oracle Identity Management, refer to the chapter describing administering Web Services security in *Oracle Application Server Web Services Security Guide*.

6.3.6 Testing the Smart Document Configuration

This section describes the steps to test the smart document security configuration.

To test the smart document with Username token:

1. Open the smart document from the demonstration support folder, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocumentUtoken\SecureSmartDocument.doc`.
2. Click **Name**, and enter your name.

   A username token dialog box is displayed.
3. Enter password and click **OK**.

   You should see the address printed on the document.

To test the smart document with X.509:

1. Open the smart document from the demonstration support folder, for example, `C:\microsoft-interoperability-guide-demo-support\securingsmartdocs\SecureSmartDocumentX509\SecureSmartDocument.doc` file.
2. Click **Name**, and enter your name.

   You should see the address printed on the document.

6.4 Related Documentation

Oracle Application Server Web Services Security Guide

7

Delivering Business Activity Monitoring Alerts and Reports to Microsoft Outlook

This chapter demonstrates how to use Oracle Business Activity Monitoring to send alerts and reports to your Microsoft Outlook e-mail client.

This chapter contains the following sections:

7.1 Overview

In today's enterprise, there is a real need to streamline and automate enterprise business processes, especially those requiring human participation. By sending workflow alerts and notifications in e-mail, you draw attention to the events that triggered them, thus increasing the likelihood that those events are addressed swiftly and accurately.

Oracle Business Activity Monitoring provides real-time visibility into enterprise operations, which enables business users to cut costs and improve processes while business events, such as a drop in inventory levels, occur. The Oracle Business Activity Monitoring architecture utilizes messaging, data integration, advanced data caching, analytics monitoring, alerting, and reporting technology to deliver critical information within seconds of an event or change in status.

You can create alerts that are triggered by specific events and conditions. These alerts can, in turn, automatically send reports to specific users. Alerts can be sent when data changes in a report, or you can use alerts to send a report to users daily or at set intervals. As a result of events and conditions, your alerts can send reports to Oracle Business Activity Monitoring users through e-mail. In this chapter, the focus is on sending alerts and reports to a Microsoft Outlook e-mail client.

Instead of having business users constantly monitor their business processes or activities for changes, you can create solutions to send alerts from Oracle Business Activity Monitoring directly into their Microsoft Outlook e-mail clients. These alerts can be regular links that open in a browser, or embedded in documents sent as e-mail attachments.

7.2 Prerequisites

To perform the steps outlined in this chapter, install the following software:

- Oracle Business Activity Monitoring 10*g* Release 2 (10.1.2)

  See *Oracle Business Activity Monitoring Installation Guide* or *Oracle Business Activity Monitoring Quick Installation Guide* for more information.

- The Media Sales sample data object

  This is installed with Oracle Business Activity Monitoring if you select the Samples option.

- Microsoft Outlook (any version)

- Microsoft Internet Explorer or Microsoft Word to open reports sent in the Multipurpose Internet Mail Extension HTML (`.MHT` format)

In addition, you must make sure that the From address is configured. To do this, perform the following steps:

1. Start Oracle Business Activity Monitoring Administrator.
2. From the drop-down list, choose **Message Center management.**
3. Click **Edit.**
4. Enter appropriate values in the Server Name, Server Port, and Email Account for Alerting fields.
5. Click **Save.**
6. Restart the Oracle Business Activity Monitoring Event Engine:
   a. In the Control Panel, click Administrative Tools, then Services.
   b. Right-click the Oracle BAM Event Engine Service, and choose Restart from the popup menu.
7. Click **Continue.**

7.3 Step-by-Step Procedures

Oracle Business Activity Monitoring lets you create the following solutions for sending alerts and reports to Microsoft Outlook e-mail clients:

- Sending E-Mail Alerts with Links
- Sending Reports as E-Mail Attachments

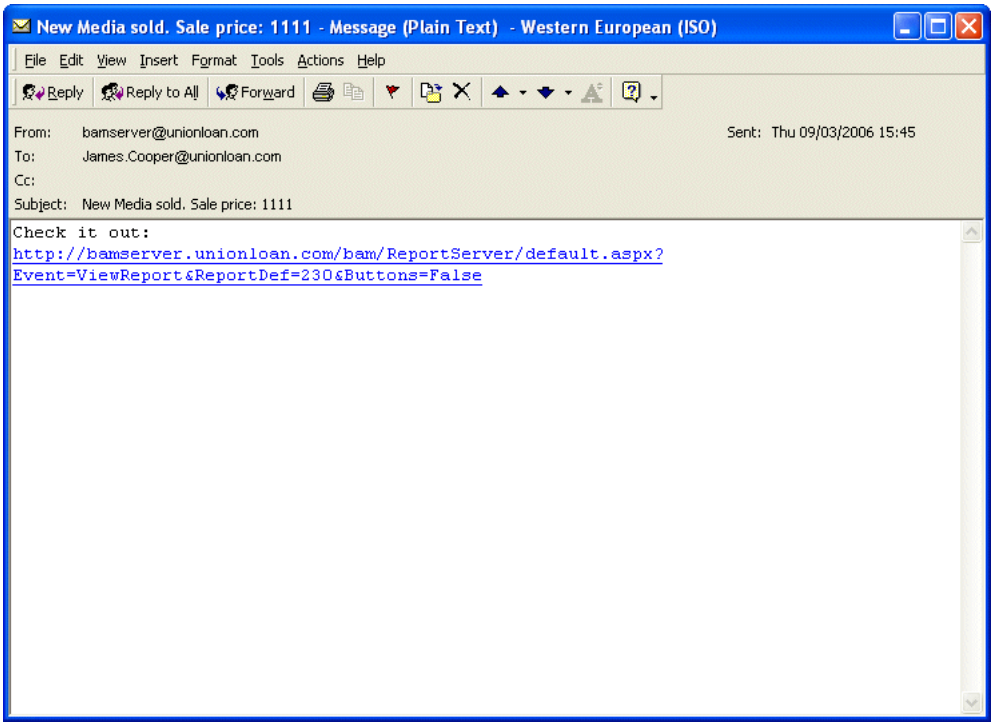
7.3.1 Sending E-Mail Alerts with Links

Oracle Business Activity Monitoring enables business executives and operations managers to improve their decision-making process by providing a real-time view of the business events occurring in their enterprise, then enabling them to use the derived intelligence to analyze and improve the efficiency of their business processes. For example, enterprises that run distributed global supply chains with just-in-time inventory practices must continually monitor their inventory levels and correlate them to the bills of material and replenishment requests they have sent to their suppliers and logistics partners. Failure to do so jeopardizes their ability to maintain a balanced flow of parts and inventory throughout their entire supply chain. With Oracle Business Activity Monitoring, as soon as the inventory level on a particular item drops below a certain level, the operations manager receives an alert. This alert can include a link to a

Web-based user interface through which the alert recipient can reorder the relevant item.

Suppose a salesperson for a music retailer wants to receive an e-mail alert (shown in Figure 7–1) when the sales of any cassette or CD reach a certain level. The e-mail must contain a link to a report that shows the breakdown of sales for that particular cassette or CD.

Figure 7–1 E-Mail Alert for Media Sales



To configure Oracle Business Activity Monitoring to send an e-mail alert with a link, perform the steps in the following sections:

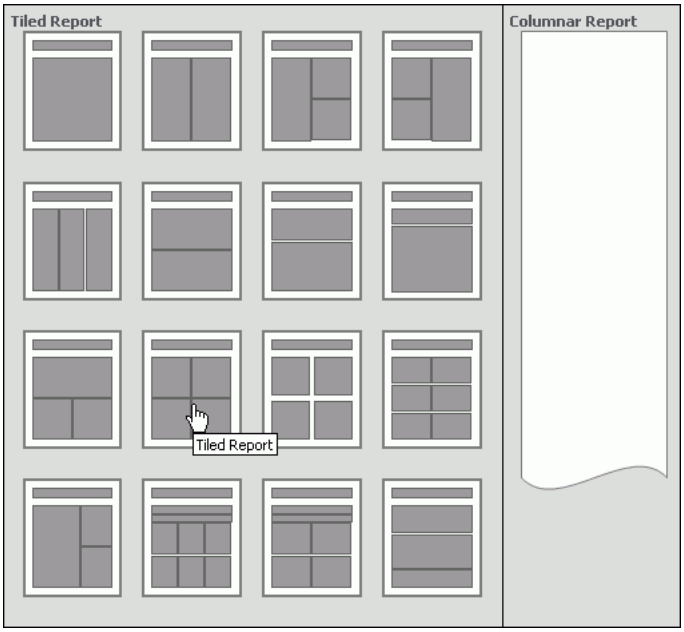
- Creating a Report
- Creating an Alert Rule
- Verifying That the Alert Is Working

7.3.1.1 Creating a Report

To create a report in Oracle Business Activity Monitoring, perform the following steps:

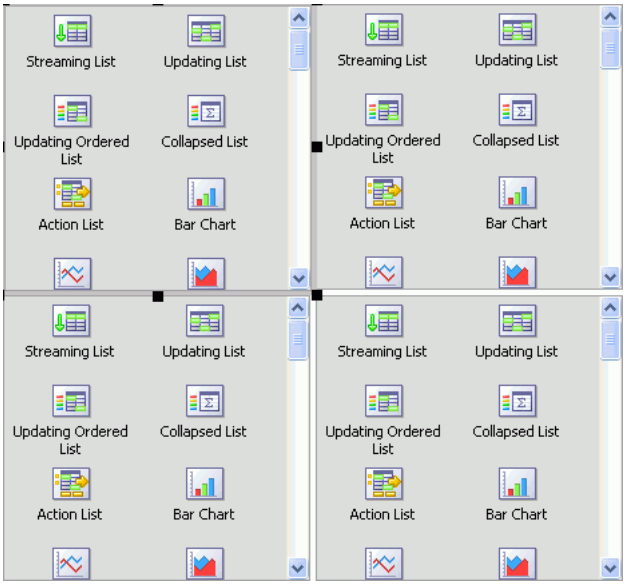
1. Start Oracle Business Activity Monitoring Active Studio.
2. Click **CREATE A NEW REPORT**.
3. Click the Tiled Report option with a title and four tiles (see Figure 7–2).

Figure 7–2 Selecting a Tiled Report



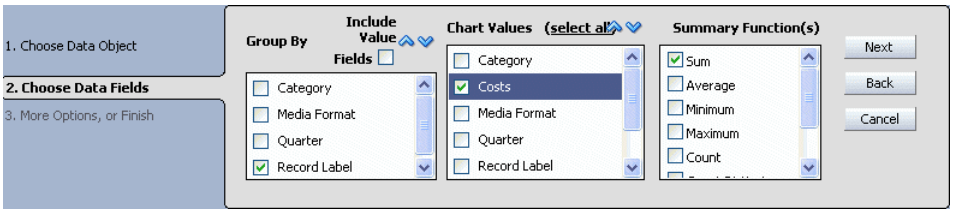
4. Click **Click to add a report title** at the top of the page, and add the report title `Media Sales Report`.
5. In the top left tile, click the **Bar Chart** icon (see Figure 7–3).

Figure 7–3 Selecting the Type of Chart for a Tile in a Report



6. In the Data Objects section, double-click **Samples.**
7. Select Media Sales, and then click **Next**.
8. In the Group By list, select the Record Label check box.
9. In the Chart Values list, select the Costs check box.
10. In the Summary Function(s) list, select the **Sum** check box (see Figure 7–4).

Figure 7–4 Choosing Data Fields

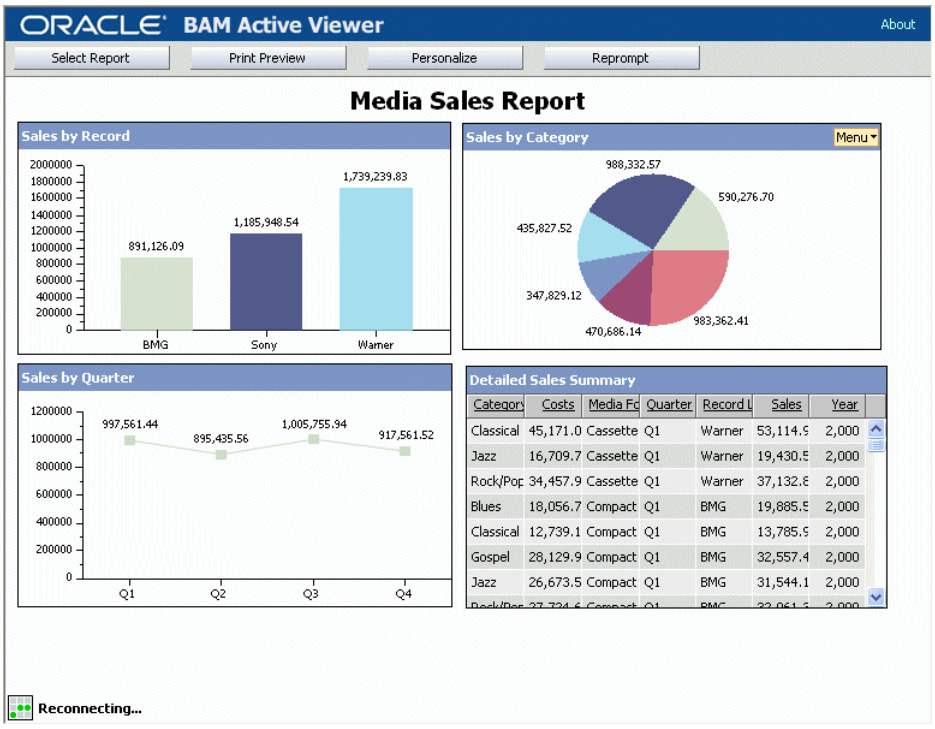


11. Click **Next**.
12. Click the **Change View Properties** icon.
13. In the View Title field, type `Sales by Record Label.`
14. Click **OK**.
15. In the top right tile, click the **Pie Chart** icon.
16. In the Data Objects section, double-click **Samples.**
17. Select Media Sales, and then click **Next**.
18. In the Group By list, select the Category check box.
19. In the Chart Values list, select the Costs check box.
20. In the Summary Function(s) list, select the Sum check box.
21. Click **Next**
22. Click the **Change View Properties** icon.
23. In the View Title field, type `Sales by Category.`
24. Click the **Data Labels** tab.
25. Select the Percent and Series Name check boxes.
26. Click **OK**.
27. In the bottom left tile, click the **Line Chart** icon.
28. In the Data Objects section, double-click **Samples.**
29. Select Media Sales, and then click **Next**.
30. In the Group By list, select the Quarter check box.
31. In the Chart Values list, select the Costs check box.
32. In the Summary Function(s) list, select the Sum check box.
33. Click **Next**.
34. Click the **Change View Properties** icon.
35. In the View Title field, type `Sales by Quarter.`
36. Click **OK**.
37. In the bottom right tile, click the **Streaming List** icon.
38. In the Data Objects section, double-click **Samples.**
39. Select Media Sales, and then click **Next**.
40. Select all the check boxes: Category, Costs, Media Format, Quarter, Record Label, Sales, and Year.

41. Click **Next.**
42. Click the **Change View Properties** icon.
43. In the View Title field, enter `Detailed Sales Summary.`
44. Click **OK**.
45. In the Actions pane, click **Save Report.**
46. Select the **My Reports** folder in Save in.
47. In the Report Name field, enter `Media Sales Report.`
48. Click **OK.**
49. Start Oracle Business Activity Monitoring Active Viewer.
50. Click **Select Report.**
51. Select Media Sales Report, and then click **OK.**

    You should see the Media Sales Report as shown in Figure 7–5.

Figure 7–5 Media Sales Report



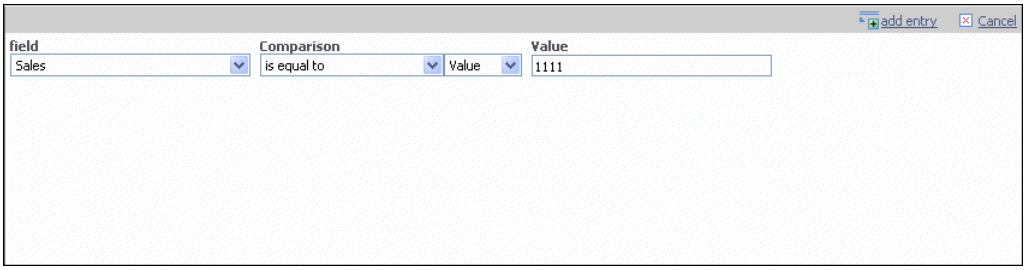
7.3.1.2 Creating an Alert Rule

To create alert rules in Oracle Business Activity Monitoring, perform the following steps:

1. Start Oracle Business Activity Monitoring Active Studio.
2. Click the **Alerts** tab.
3. Click **CREATE A NEW ALERT**.
4. Click **CREATE A RULE**.
5. In the Rule Name field, type `CD Sales.`

6. From the Select an Event list, select **When a data field in a data object meets specified conditions.**
7. Click **Next**.
8. From the Select an Action list, select **Send a message via email**.
9. In the Rule Expression section, click **this data field has a condition of x** to display the Alert Rule Editor.
10. In the Data Objects section, select **Media Sales**.
11. Click the **Row Filter** tab.
12. Click **add new entry**.
13. From the field list, choose Sales.
14. From the Comparison list, choose is equal to.
15. From the next list, select Value.
16. In the Value field, type `1111`. This reflects the number of CDs or cassettes that must be sold in order to trigger the alert.
17. Click **add entry** (see Figure 7–6).

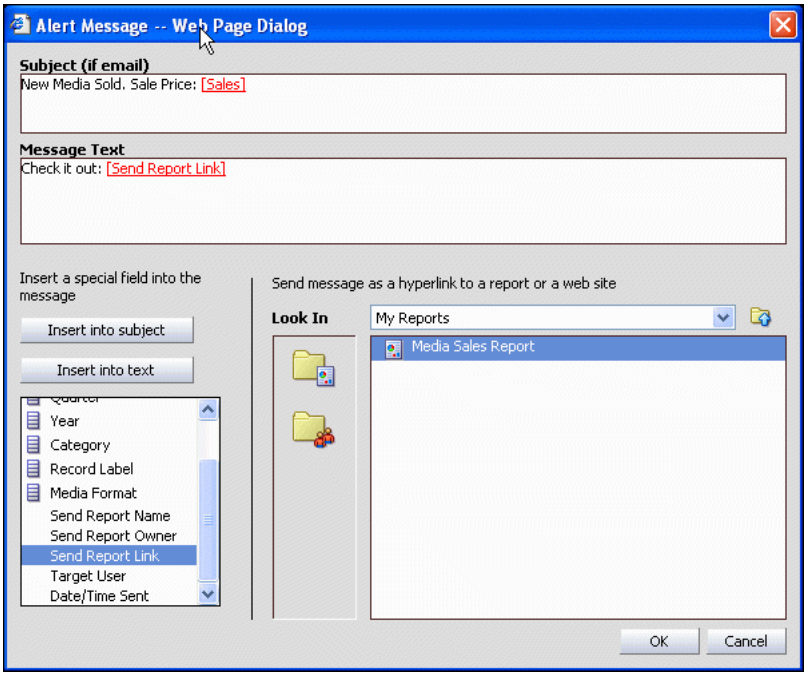
Figure 7–6 Specifying a Filter for the Alert



18. Click **OK**.
19. In the Rule Expression section, click **create message** to display the Alert Message dialog box.
20. In the Subject field, type `New media sold. Sale price:.`
21. Alerts can include the data that triggered the alert in the first place.

    From the list on the left, select Sales, then click **Insert into subject**.
22. In the Message Text field, enter `Check it out:.`
23. From the list on the right, select the Media Sales Report that you created in Section 7.3.1.1, "Creating a Report".
24. From the list on the left, select Send Report Link, then click **Insert into text** (see Figure 7–7).

Figure 7–7 Specifying the Alert Message



25. Click **OK**.
26. In the row for the action, click **select user** to display the Select Names dialog box.
27. Select the user to whom you want to send the alert (see Figure 7–8). For the purposes of this exercise, send the alert to yourself.

Note: You can use Oracle Business Activity Monitoring Administrator to manage users and specify their e-mail addresses. For information about how to do this, refer to *Oracle BAM Administrator's Guide*.

It is also possible to configure Oracle Business Activity Monitoring to be provisioned with the user information stored in your corporate directory, for example Oracle Internet Directory.

Figure 7–8 Specifying Alert Recipients

28. Click **OK**.

29. Click **OK** to close the Rule Creation and Edit dialog box.

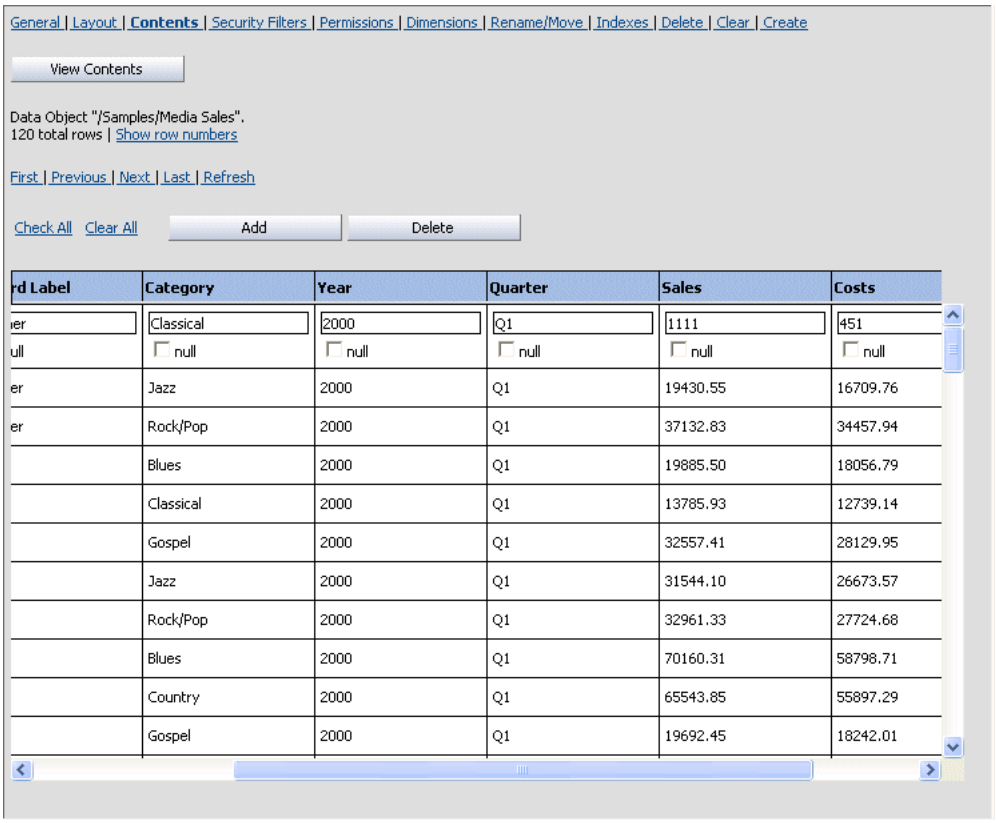
An alert will now be sent to your e-mail client when sales of any CD or cassette reach 1111.

7.3.1.3 Verifying That the Alert Is Working

As a developer, you typically want to test the alert without having to wait until the alert condition is satisfied. If you have administrative privileges, you can use Oracle Business Activity Monitoring Architect to manipulate the underlying data to trigger the alert. To do this, perform the following steps:

1. Start Oracle Business Activity Monitoring Architect.
2. In the Folders pane, click **Samples**.
3. In the Data Objects pane, click **Media Sales**.
4. Click **Contents** in the list of links at the top of the right pane.
5. In the top row, make a note of the current value for the Sales column.
6. Click **Edit Contents**.
7. In the top row, click **Edit**.
8. Change the value in the Sales column to `1111` (see Figure 7–9).

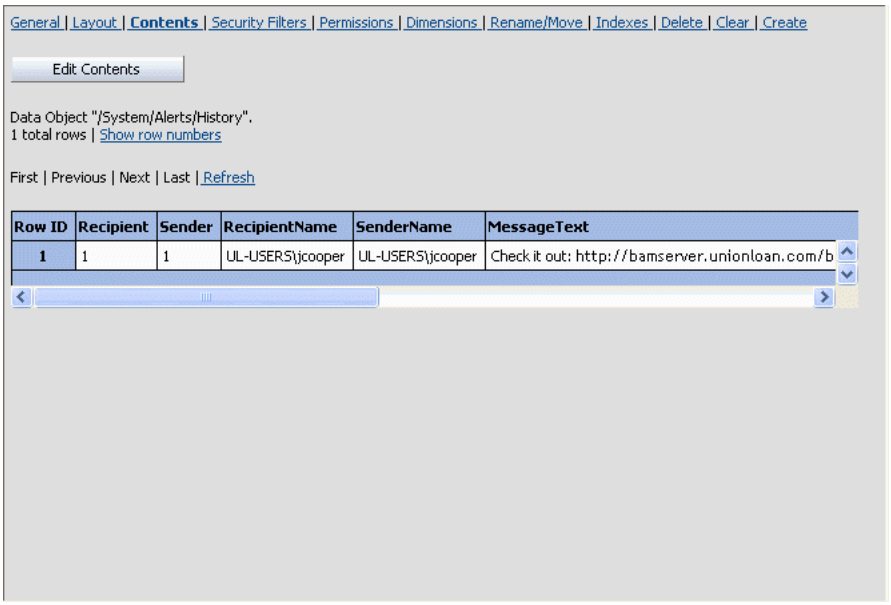
Figure 7–9 Triggering the Alert



9. Click **Update**.
10. In the Folders pane, click **System**, then click **Alerts**.
11. In the Data Objects pane, click **History**.

12. Click **Contents** in the list of links at the top of the right pane.
13. If necessary, scroll down to the end of the table. You should see that an alert has been sent to your e-mail account (see Figure 7–10).

Figure 7–10 Alert History



14. Check your e-mail client to make sure that you received the alert.
15. Open the message and click the link to view the report.
16. As soon as you have verified that the alert is working, remember to use Oracle Business Activity Monitoring Architect to reset any changes you made to the Media Sales data.

7.3.2 Sending Reports as E-Mail Attachments

Suppose that, instead of sending an alert, you want to e-mail a daily report that contains a snapshot of data from data objects in the Active Data Cache, as shown in Figure 7–5. Oracle Business Activity Monitoring enables you to e-mail such a report as an attachment in the `.MHT` format. Because `.MHT` files can be opened in either Microsoft Internet Explorer or Microsoft Word, the data can be reviewed even when the recipient is not connected to the Internet.

To configure Oracle Business Activity Monitoring to send reports to e-mail addresses, perform the steps in the following sections:

- E-Mailing the Report
- Verifying That the Report Was Sent

7.3.2.1 E-Mailing the Report

To e-mail a report to one or more Oracle Business Activity Monitoring users, perform the following steps:

1. Start Oracle Business Activity Monitoring Active Studio.
2. Click the **My Reports** tab.
3. Double-click the report that you created in Section 7.3.1.1, "Creating a Report".

4. In the Actions pane, click **Email**, and then click **Report page.**

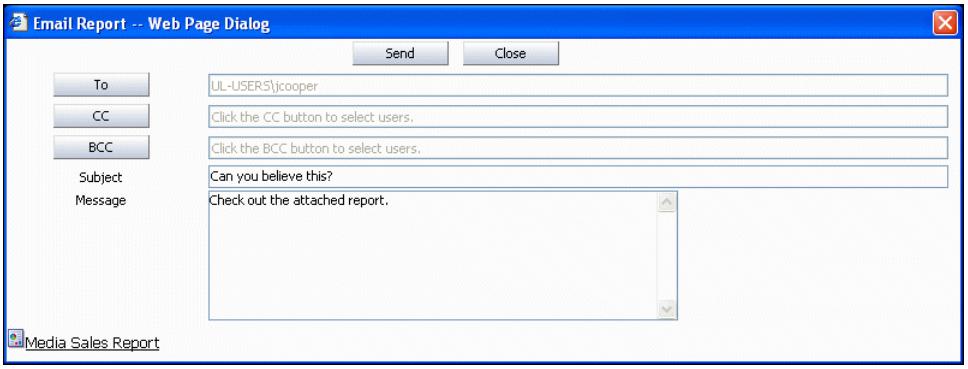
   The E-mail Report dialog box is displayed.

5. In the To, CC, and BCC fields, specify Oracle Business Activity Monitoring user names. You can send e-mail *only* to users of Oracle Business Activity Monitoring. Include yourself as one of the recipients.

6. In the Subject field, type `Can you believe this?`.

7. In the Message field, type `Check out the attached report.` (see Figure 7–11).

Figure 7–11 Specifying Details for E-Mailing a Report



8. Click **Send**.

   Tip: If you receive an error at this point, make sure that you have configured the From address as specified in Section 7.2, "Prerequisites".

7.3.2.2 Verifying That the Report Was Sent

To verify that the report was sent, perform the following steps:

1. Check your e-mail client to make sure that you received the message.
2. Open the message and download the attached `.MHT` file to your local file system.
3. Open the downloaded file in Microsoft Internet Explorer or Microsoft Word and view the report.

7.4 Related Documentation

- *Oracle Business Activity Monitoring Installation Guide*
- *Oracle BAM Active Studio User's Guide*
- *Oracle BAM Architect User's Guide*

8

Delivering Business Intelligence Information to Microsoft Excel

This chapter demonstrates how to deliver business intelligence information to Microsoft Office. It shows how you can save an Oracle Business Intelligence Discoverer worksheet as a Microsoft Excel worksheet and how you can use the Oracle Business Intelligence Spreadsheet Add-In to work with live OLAP data in Microsoft Excel.

This chapter contains the following sections:

8.1 Overview

Oracle Business Intelligence enables you to rapidly develop and deploy data warehouses and data marts with an integrated array of query, reporting, analysis, data integration and management, desktop integration, and Business Intelligence application development capabilities. Oracle Business Intelligence, available both as a standalone or as part of Oracle Application Server Enterprise Edition, includes the following components:

- OracleBI Discoverer: An intuitive ad-hoc query, reporting, analysis, and Web-publishing tool that empowers business users at all levels to gain immediate access to information from data marts, data warehouses, online transaction processing systems, and Oracle E-Business Suite.
- OracleBI Spreadsheet Add-In: Provides OLAP data access from within Microsoft Excel worksheets. You can also use the OracleBI Beans Calculation and Query Builder beans to analyze that data.
- OracleBI Warehouse Builder: Enables rapid design, deployment, and management of data and metadata.
- OracleBI Beans: Builds powerful custom business intelligence applications.
- OracleAS Reports Services: Provides enterprise reporting.

Microsoft Excel remains the most widely used worksheet tool. It is also a product with which many users are familiar. While OracleBI Discoverer provides users with a powerful and secure environment to analyze and visualize data from data marts and data warehouses, it may be required at certain times to provide this data to users who

do not have access to OracleBI Discoverer. Under this situation, it makes sense to provide users with access to the data in order to use the features of Microsoft Excel to further analyze data. OracleBI Discoverer enables users to do just this by exporting their OracleBI Discoverer worksheets as Microsoft Excel worksheets, with formatting and layout preserved.

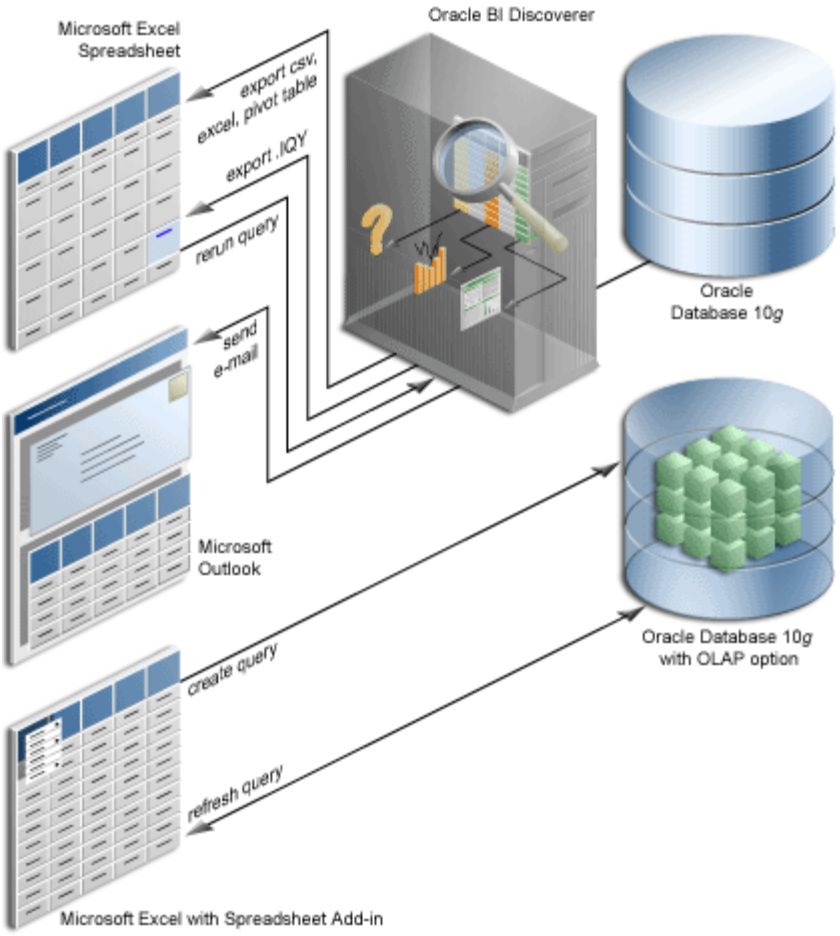
OracleBI Discoverer also enables users to export crosstabs as Microsoft Excel pivot tables. This powerful feature means that users neither need to know how to create pivot tables in Microsoft Excel (which can be a nonintuitive exercise for novice users), nor do they need to waste time in report creation; instead they can spend more time on analysis.

Users can e-mail OracleBI Discoverer reports (tables or crosstabs) as Microsoft Excel worksheets to other people. This way they can share the results of their analysis and insight with others. By preserving the layout, format, and certain calculations in the exported worksheet, users can spend more time on analysis and less on re-creating reports.

The use of the Spreadsheet Add-In combines the features of Microsoft Excel with the robustness, security, and scalability of Oracle Database. Users can use the familiar interface of Microsoft Excel, and at the same time, have their data inside a secure Oracle Database. Users can also make use of the powerful analytics built inside the OLAP option of the database.

Figure 8–1 illustrates the various ways that Oracle Business Intelligence interoperates with Microsoft Office.

Figure 8–1 Business Intelligence Interoperation with Microsoft Excel



8.2 Prerequisites

To perform the steps outlined in this chapter, first install the following software:

- Oracle Database 10*g* (10.1.0.3 or later)
- Oracle Business Intelligence 10*g* Release 2 (10.1.2)

  See *Oracle Business Intelligence Installation Guide* for your operating system for more information.

- Oracle Business Intelligence Spreadsheet Add-In. You can download this add-in from OTN at

  `http://www.oracle.com/technology/products/bi/spreadsheet_addin/download/index.html`

  For instructions on how to install the add-in, read the Installation Guide and Release Notes at

  `http://www.oracle.com/technology/products/bi/spreadsheet_addin/docs/10121/html_ssa_ig_rn/output/toc.htm`

- Oracle Business Intelligence samples. The samples are available at

  `http://www.oracle.com/technology/products/bi/samples`

Download the ZIP file and follow the instructions provided in the `samples_readme.htm` file. You must install the common schema and the OracleBI Discoverer sample workbooks.

Tip: It is advisable that this task be performed by a DBA.

- Microsoft Excel 2000 or later

8.3 Step-by-Step Procedures

This chapter shows two ways that you can use Oracle Business Intelligence to send business intelligence information to Microsoft Excel:

- Pushing Business Intelligence Information to Microsoft Excel
- Pulling Live Data into Microsoft Excel

8.3.1 Pushing Business Intelligence Information to Microsoft Excel

If you want to share your OracleBI Discoverer worksheet with others, it is helpful to do so using a familiar format that does not require them to install additional software. OracleBI Discoverer enables you to meet this requirement in the following ways:

- By viewing an OracleBI Discoverer worksheet on the Web in a browser.
- By making a worksheet available in many different formats, including HTML, PDF, CSV, text, and so on.
- By saving an worksheet as a Microsoft Excel worksheet. You can export the worksheet to the Microsoft Excel format, and also save any graphs within your worksheet as `.PNG` (or `.GIF`) files. You can then insert these graphics into the worksheet. When you export to Microsoft Excel, you can also export formats and formulas, and your worksheet fonts, colors, and styles are preserved.

Three ways you can push business intelligence information to Microsoft Excel are the following:

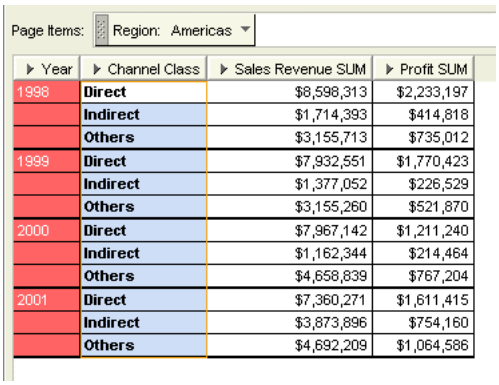
- Saving an OracleBI Discoverer Worksheet as a Microsoft Excel Worksheet
- Saving an OracleBI Discoverer Workbook as a Microsoft Excel Web Query
- Sending a Worksheet as an E-Mail Attachment

8.3.1.1 Saving an OracleBI Discoverer Worksheet as a Microsoft Excel Worksheet

To save an OracleBI Discoverer worksheet as a Microsoft Excel worksheet, perform the following steps:

1. Start OracleBI Discoverer Plus and connect to the relational data source where you installed the samples as described in Section 8.2, "Prerequisites".
2. In the Open Workbook from Database dialog box, expand the **Sales & Profits by Time, Geography, & Channel** sample workbook.
3. Select the **Annual Regional Sales & Profits by Channel** sample worksheet and click **Open.**
4. Apply some formatting to the worksheet, for example, make some text bold and add some color (see Figure 8–2).

Figure 8–2 OracleBI Discoverer Worksheet with Formatting



Page Items: Region: Americas

Year	Channel Class	Sales Revenue SUM	Profit SUM
1998	Direct	$8,598,313	$2,233,197
	Indirect	$1,714,393	$414,818
	Others	$3,155,713	$735,012
1999	Direct	$7,932,551	$1,770,423
	Indirect	$1,377,052	$226,529
	Others	$3,155,260	$521,870
2000	Direct	$7,967,142	$1,211,240
	Indirect	$1,162,344	$214,464
	Others	$4,658,839	$767,204
2001	Direct	$7,360,271	$1,611,415
	Indirect	$3,873,896	$754,160
	Others	$4,692,209	$1,064,586

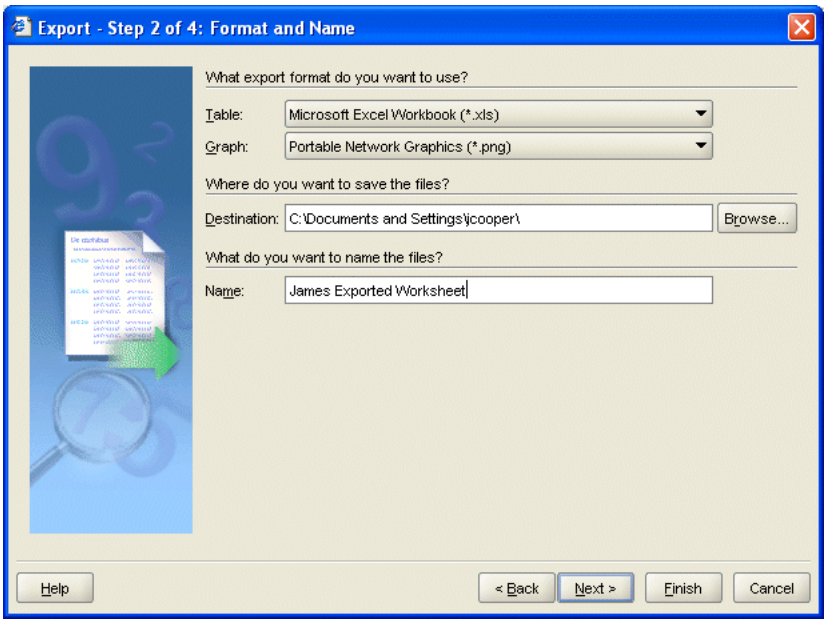
5. Select File, then Export to display the Export Wizard.

> **Note:** You can also click the Export to Excel toolbar button. This exports your worksheet using default settings. If you want to specify the export settings yourself, use the menu.

6. Select Current Worksheet.
7. Click **Next**.
8. From the Table list, select Microsoft Excel Workbook.

   If the worksheet was a crosstab, you could, optionally, select Microsoft Excel Workbook with Pivot Table instead.
9. From the Graph list, select Portable Network Graphics.
10. In the Destination field, enter the location where you want to save the exported files. Click **Browse**, if necessary.
11. In the Name field, enter `<YourName> Exported Worksheet` (see Figure 8–3).

Figure 8–3 Exporting a Worksheet in Microsoft Excel Format



12. Click **Next**.
13. Select Use Current On Screen Size for the size of the exported graph.
14. Click **Finish** to start the export operation.

    When the export operation is completed, the Export Log dialog box displays a list of the files created during the export operation.
15. Click **OK**.

    Figure 8–4 shows that the formatting you applied earlier has been preserved.

Figure 8–4 Exported Worksheet in Microsoft Excel (with Formatting Preserved)

Microsoft Excel - Sales & Profits by Time, Geography, & Channel.xls

File Edit View Insert Format Tools Data Window OracleBI Help

Arial

F22

	A	B	C	D	E
1	Sales & Profits by Time, Geography, & Channel				
2					
3	Region:Americas				
4					
5	Year	Channel Class	Sales Revenue SUM	Profit SUM	
6	1998	**Direct**	$8,598,313	$2,233,197	
7		**Indirect**	$1,714,393	$414,818	
8		**Others**	$3,155,713	$735,012	
9	1999	**Direct**	$7,932,551	$1,770,423	
10		**Indirect**	$1,377,052	$226,529	
11		**Others**	$3,155,260	$521,870	
12	2000	**Direct**	$7,967,142	$1,211,240	
13		**Indirect**	$1,162,344	$214,464	
14		**Others**	$4,658,839	$767,204	
15	2001	**Direct**	$7,360,271	$1,611,415	
16		**Indirect**	$3,873,896	$754,160	
17		**Others**	$4,692,209	$1,064,586	
18					

The graph is exported into a separate `.PNG` file. If you want to include it, you must manually insert it into the Microsoft Excel worksheet.

Note: You have the following options when exporting to Microsoft Excel:

- Microsoft Excel worksheet with formatting preserved.
- Microsoft Excel worksheet with an Microsoft Excel Pivot Table created. This option is available for Discoverer crosstabs.

In addition, you can export to a CSV (comma-delimited values) format, which is suitable when you do not need formatting and need to conserve the file size.

8.3.1.2 Saving an OracleBI Discoverer Workbook as a Microsoft Excel Web Query

You can also export OracleBI Discoverer workbooks in Microsoft Excel Web Query (`.IQY`) format. This means that the Microsoft Excel worksheet stores the query used to obtain the OracleBI Discoverer data, so that users can refresh the data within Microsoft Excel. This ensures that users can always view the most up-to-date data.

1. Start OracleBI Discoverer Plus and connect to the relational data source where you installed the samples as described in Section 8.2, "Prerequisites".
2. Expand the **Sales & Profits by Time, Geography, & Channel** sample workbook.
3. Select the **Annual Regional Sales & Profits by Channel** sample worksheet and click **Open.**
4. Select File, then Export to display the Export Wizard dialog box.
5. Select Current Worksheet.
6. Click **Next**.
7. From the Table list, select Web Query for Microsoft Excel 2000+.
8. In the Destination field, enter the location where you want to save the exported files. Click **Browse**, if necessary.
9. In the Name field, enter `<YourName> Exported Worksheet2.`
10. Click **Finish** to start the export.

    When the export is completed, the Export Log dialog displays a list of the files created during the export.
11. Click **OK**.

    You are prompted to enter the password for the current user.
12. Enter your password and click **OK.**

    Microsoft Excel connects to the database, then retrieves and displays the latest data.

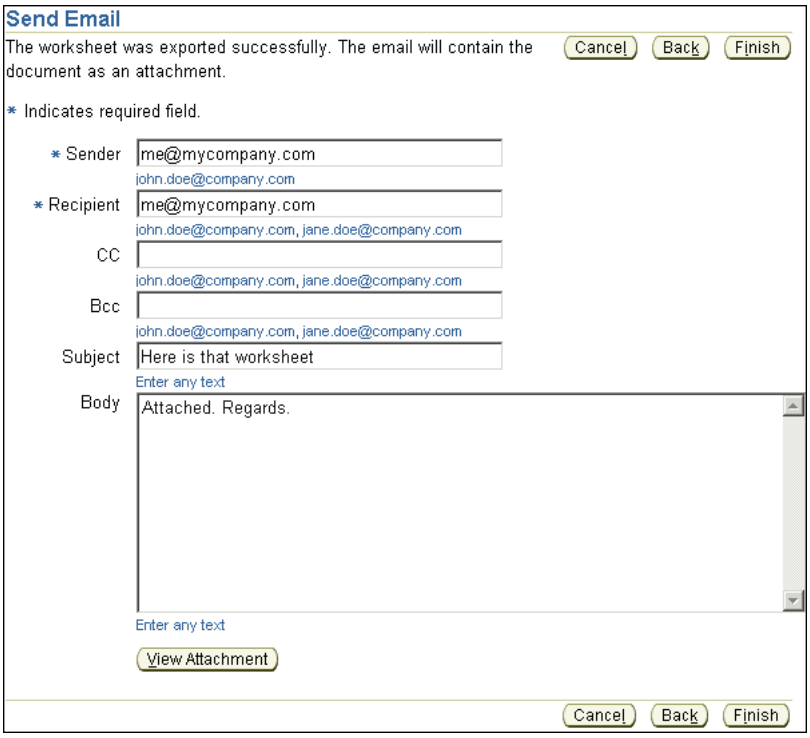
8.3.1.3 Sending a Worksheet as an E-Mail Attachment

If you want to send a worksheet directly to another user's e-mail account, you can use OracleBI Discoverer Viewer. You can e-mail the worksheet in a number of formats, such as HTML (in a ZIP file), Oracle Reports, XML, PDF, and a Microsoft Excel workbook.

To send a worksheet as an e-mail attachment, perform the following steps:

1. Start OracleBI Discoverer Viewer and connect to the relational data source where you installed the samples as described in Section 8.2, "Prerequisites".
2. Expand the **Sales & Profits by Time, Geography, & Channel** sample workbook.
3. Click the **Annual Regional Sales & Profits by Channel** sample worksheet.
4. From the Actions list, click **Send as e-mail**.
5. From the list, select **Microsoft Excel Workbook**.
6. Click **Next**.
7. In the Sender field, enter your own e-mail address.
8. In the Recipient field, enter the e-mail address of the user to whom you want to send the worksheet. For the purposes of this exercise, enter your own e-mail address.
9. In the Subject field, enter `Here is that worksheet.`
10. In the Body field, enter `Attached. Regards` (see Figure 8–5).

Figure 8–5 Sending a Worksheet as a Microsoft Excel Attachment



11. To preview what the attached file will look like, click **View Attachment**.
12. Click **Finish**.
13. Check your inbox to make sure that you received the worksheet.
14. Open the message and open the attached Microsoft Excel worksheet.

8.3.2 Pulling Live Data into Microsoft Excel

Many users within an enterprise are more familiar and prefer working with desktop applications, such as Microsoft Excel. Therefore, most enterprises have seen a

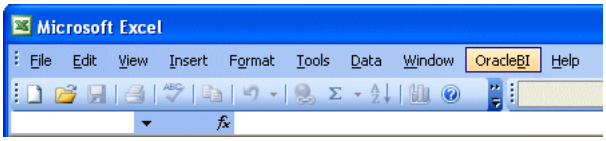
proliferation of data-extraction programs and downloading of static data into worksheets within their data warehouse environment. This causes enormous problems, both technical and business-related. Continually downloading and manipulating data in this manner causes an ever-expanding range of disconnected worksheets. It is never clear from the worksheet analysis how historical data based on numerous sources is managed. For example, what happens when the source systems are updated or restarted? Because there is no connection to the source data, the user is not alerted to the availability of refreshed data. As a result, it is never clear which worksheet is the latest version. Worksheets also have scalability limitations in terms of the volume of data that can be processed by a single sheet. Again, users resolve this by creating multiple worksheets and attaching them together with formulas.

The OracleBI Spreadsheet Add-In resolves all of these issues and many more. It combines the analytic power and scalability of Oracle OLAP with the familiarity of Microsoft Excel by embedding OLAP capabilities directly within Microsoft Excel. Users report against common business definitions that are stored centrally in Oracle Database. This provides a consistent and high-quality view of their corporate information. In addition, users can perform ad hoc analysis on this data using traditional OLAP exploration techniques such as drilling, pivoting, and paging the view of the data.

To pull data into Microsoft Excel, perform the following steps:

1. Start Microsoft Excel.
2. You should see a menu option for OracleBI, as shown in Figure 8–6.

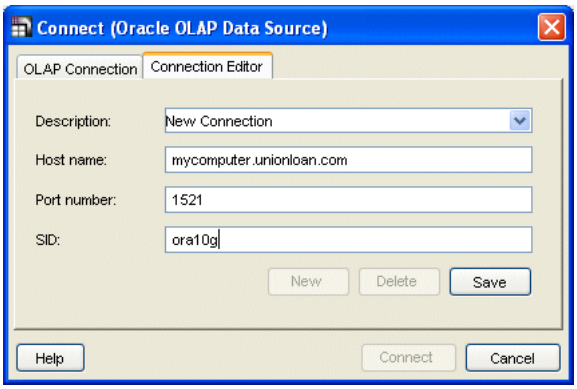
Figure 8–6 OracleBI Spreadsheet Add-In Menu Option



> **Note:** If you do not see this option, make sure that you have downloaded and installed the OracleBI Spreadsheet Add-In as described in Section 8.2, "Prerequisites".

3. Select OracleBI, then New Query to create a new OLAP query in your worksheet.
4. Click the **Connection Editor** tab.
5. Click **New.**
6. Enter an appropriate Description (see Figure 8–7), and then enter the Host Name, Port Number, and SID for the OLAP data source where you installed the samples as described in Section 8.2, "Prerequisites".

Figure 8–7 Connecting to the Database

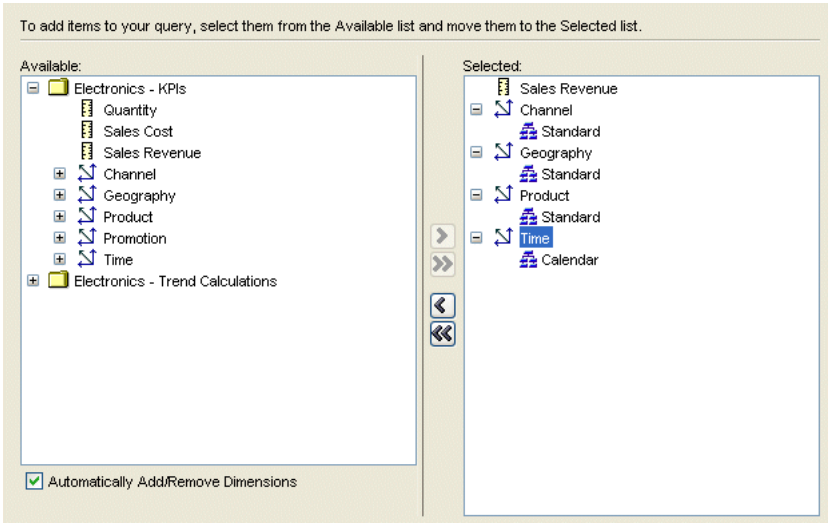


7. Click **Save.**
8. Click the **OLAP Connection** tab.
9. Enter your User name and Password.
10. Click **Connect.**

    The Oracle OLAP Query Wizard appears. This is the same as the Query Wizard in OracleBI Discoverer, so if you are already familiar with that, you do not need to learn how to use a new tool.
11. Click **Next** to continue from the Welcome page.
12. The Available list contains all the OLAP measures and dimensions that can be displayed in your worksheet.

    Expand the **Electronics - KPIs** folder, select Sales Revenue, and click the right-angle bracket (**>**).
13. In this report, Promotions will not be part of the analysis, so select Promotion and click the left-angle bracket (**<**) to remove it from the list (see Figure 8–8).

Figure 8–8 Items for the Query



14. Click **Next.**

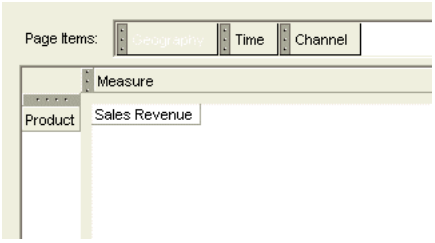
15. In the Layout step of the wizard, you can change the layout of the data by dragging and dropping the appropriate dimension or measure tiles.

    Move Channel to the Page Items region.

16. Move Product and Geography so that Product is displayed on the Row edge and Geography is displayed first in the Page Items region.

    The layout should look like Figure 8–9.

Figure 8–9 Layout of OLAP Query



17. Click **Next.**

18. Select the members for the Channel dimension.

    Click Channel total, then click the right-angle bracket (**>**).

19. Click **Next.**

20. Select the members for the Geography dimension.

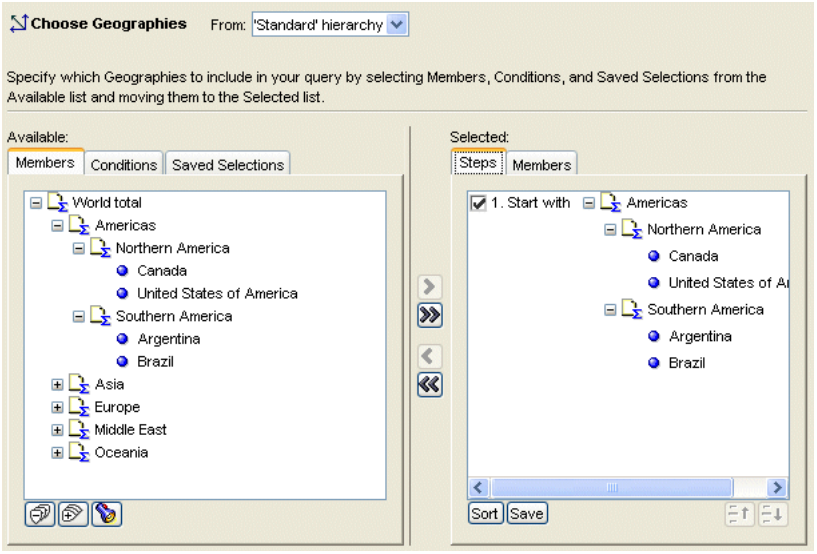
    a. Expand the **World total** node.

    b. Expand the **Americas** node.

    c. Expand the **Northern America** and **Southern America** nodes.

    d. Select the following: Americas; Northern America; Canada; United States of America; Southern America; Argentina; and Brazil, then click the right-angle bracket (>) (see Figure 8–10).

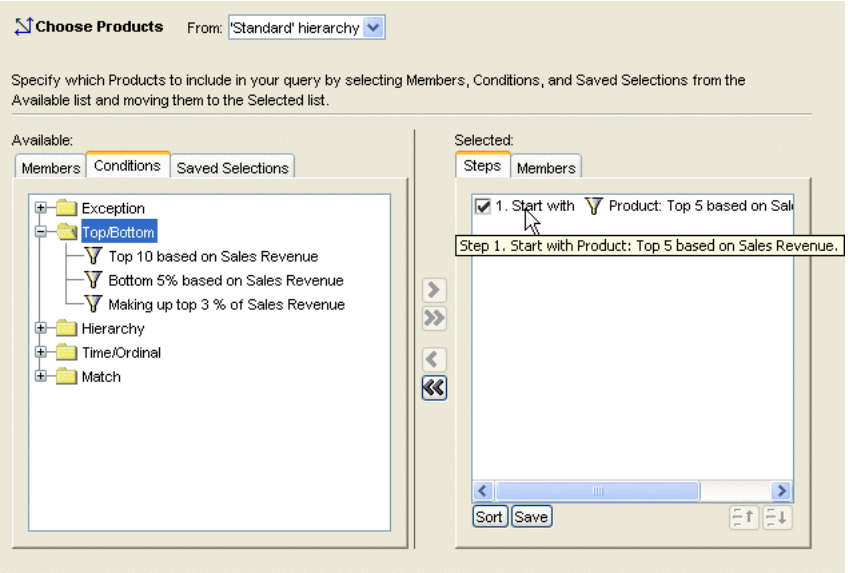
Figure 8–10 Members for the Geography Dimension



21. Click **Next.**
22. Select the members for the Products dimension.
    a. Click the **Conditions** tab.
    b. Expand the **Top/Bottom** folder.
    c. Select Top 10 based on Sales Revenue, and click the right-angle bracket (**>).**
    d. In the Selected list, select the Top 10 condition.
    e. Click the **10** hyperlink, then change the value to `5` .

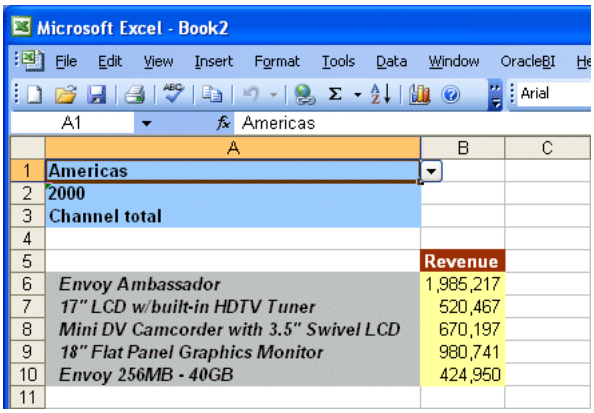
       The condition now reads Start with Product: Top 5 based on Sales Revenue (see Figure 8–11).

Figure 8–11 Members for the Products Dimension



23. Click **Next.**
24. For the Times dimension, select 2000 and 2001, then click the right-angle bracket (>).
25. Click **Finish** to execute the query. The results should look something like Figure 8–12.

Figure 8–12 OLAP Query Results in Microsoft Excel



The OLAP query returns the data for the top 5 products based on the sales revenue for the Americas, Channel total, and the year 2000.

Whenever the underlying data changes, you can refresh your worksheet so that it reflects those changes simply by selecting the data and choosing Refresh Query from the OracleBI menu.

26. Save your worksheet.

The data in the worksheet is preserved. This means that when you close the worksheet or disconnect from the database, you simply have to reconnect to the OLAP data source to retrieve the most recent data.

27. The results of the query use default formatting. You can use Microsoft Excel's formatting features to change the formatting as required.

a. In the worksheet, select the cells containing the revenue values.

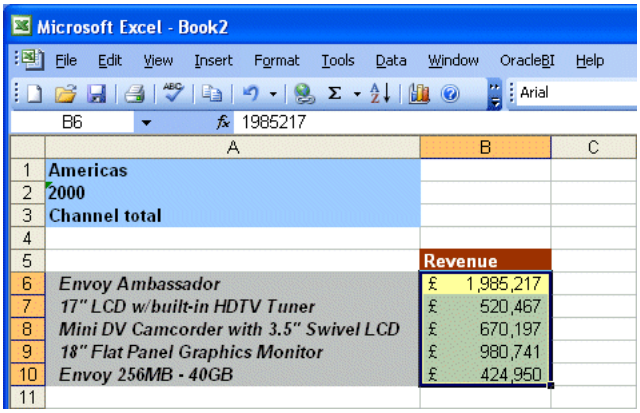
b. Click the Currency Style tool.

Currency formatting is applied to the data.

c. Remove the decimal places by clicking the Decrease Decimal tool twice.

The OLAP data should now look something like Figure 8–13.

Figure 8–13 OLAP Data with Microsoft Excel Formatting



28. Save your worksheet.

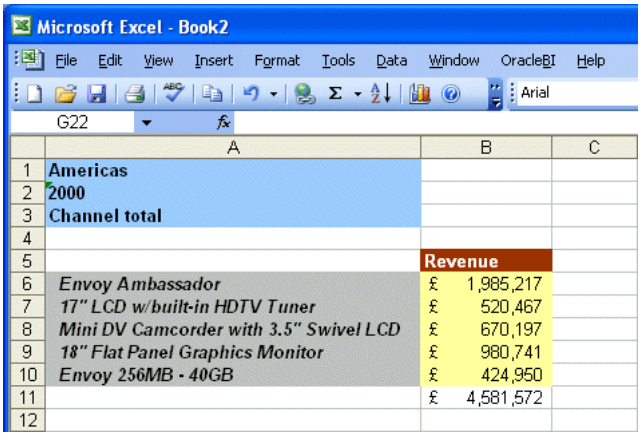
29. You can add Microsoft Excel calculations, working with the OLAP data just as you would any other data in a Microsoft Excel worksheet.

    a. To add a subtotal formula for the Revenue values, first select the cell below the last Revenue value, then click the Auto Sum tool.

       The subtotal formula is created.

    b. Press Enter to accept the formula. The results should look something like Figure 8–14.

Figure 8–14 Microsoft Excel Subtotal of OLAP Data



30. Save your worksheet.

31. You can also use the Microsoft Excel Chart Wizard to create charts based on the OLAP data, just as you do with regular Microsoft Excel data.

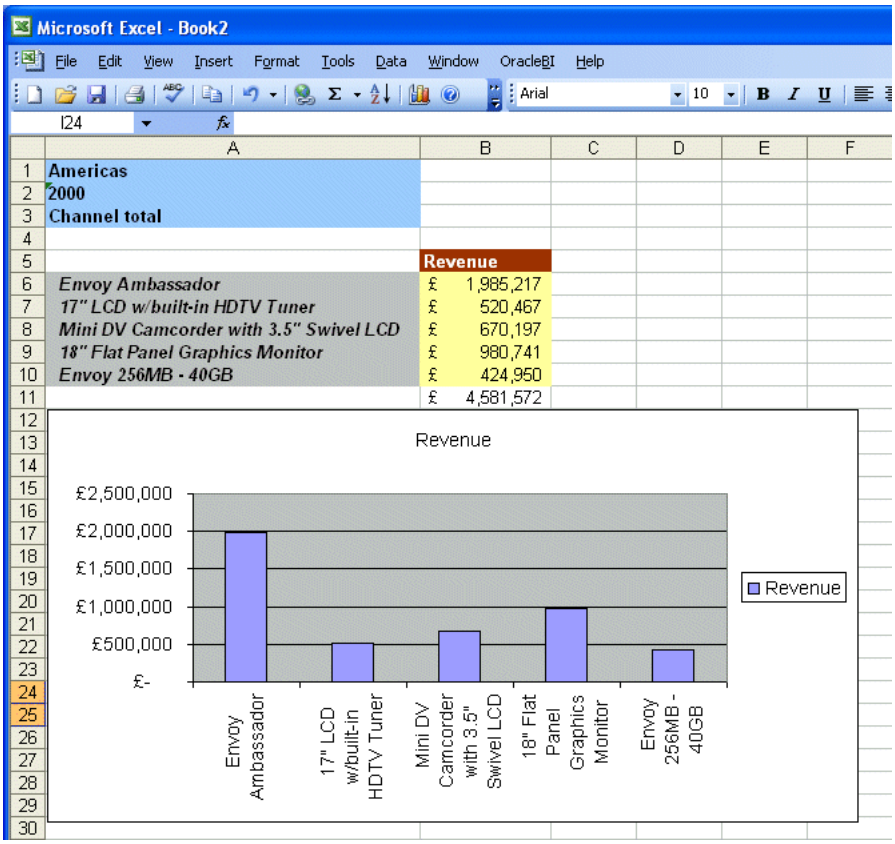
    a. Select the product and revenue values, then click the Chart Wizard tool.

    b. From the Chart Type list, select Column.

    c. Click **Finish.**

    d. Drag the chart underneath the data and resize it appropriately so that it looks like Figure 8–15.

Figure 8–15 Microsoft Excel Chart Based on OLAP Data



32. Save your worksheet.

8.4 Related Documentation

- *Oracle Business Intelligence Discoverer Plus User's Guide*
- *Oracle Business Intelligence Discoverer Viewer User's Guide*

9

Managing Tasks and Collaborating in Microsoft Outlook

This chapter demonstrates how you can set up and use Oracle Connector for Outlook to access a broad array of collaborative functions through the familiar Microsoft Outlook interface. It also provides information on using Oracle Real-Time Collaboration tools that let you communicate and collaborate in real time with coworkers and business contacts.

This chapter contains the following sections:

9.1 Overview

Until recently, business managers implementing systems for departmental communication and collaboration have been forced to choose between using fragmented applications from separate vendors, or collaborative products that did not meet all of their scalability, reliability, manageability, and security concerns. Oracle meets these concerns with Oracle Collaboration Suite 10*g*, a secure and reliable content management and collaboration solution built on Oracle Database 10*g* and Oracle Application Server 10*g* infrastructure.

With effortless integration with Microsoft Office and Microsoft Active Directory, Oracle Collaboration Suite processes unstructured content, documents, e-mail, scheduling, workspaces, discussion boards, and real-time collaboration as enterprise business processes, for which Microsoft Office can be configured to be the interface.

This section covers the following Oracle Collaboration Suite components:

9.1.1 Oracle Collaboration Suite 10*g* Calendar

Oracle Calendar is the time management component of Oracle Collaboration Suite, combining group and resource scheduling functions with a variety of access methods to give you up-to-date task management information.

Microsoft Outlook is one of the most heavily used desktop applications in the world and can be deployed and managed in ways that will suit a wide range of enterprise needs. By using Oracle Connector for Outlook, you can integrate Oracle Calendar with Microsoft Outlook, and as a result, you can manage all your Oracle Calendar-related tasks from the Microsoft Outlook client.

The subsequent topics describe the features of Oracle Calendar that can be supported from Microsoft Outlook, as well as a few related concepts.

Meetings, Tasks, Daily Notes, and Contacts

The building blocks of your agenda are meetings, which are blocks of reserved time in your schedule for any type of activity with a start and an end time. You can use meetings to block off time in your agenda for any amount of time, or even an entire day.

Oracle Calendar enables you to create tasks to keep track of ongoing projects and work that must be completed within a specific time frame. You can set reminders, add details, and attach documents to those tasks. Daily notes and day events can be created to keep track of who is out of the office, statutory holidays, a coworker's birthday, and more. You can make sure that no event is forgotten, by setting reminders and notifications for your agenda entries. You can keep track of your business and personal contacts using the Oracle Calendar desktop client or Oracle Connector for Outlook address book. Add notes to your contacts if you want to be reminded of deadlines or other important events.

Real-Time Conflict Checking and Resolution Capabilities

Oracle Calendar offers real-time conflict checking and resolution capabilities to help ease the process of scheduling meetings and decrease the likelihood of absent invitees. When scheduling a meeting using the Oracle Calendar Web client or desktop client, click the **Check Conflicts** button to view scheduling conflicts with users or resources. If a conflict is found, then you can use the Suggest Date and Time feature to have Oracle Calendar suggest a series of available times for all invitees. The AutoPick feature in Oracle Connector for Outlook offers similar functions. Before you schedule a meeting, use the Group View feature to quickly check what date and time best suits the schedules of the invitees, including meeting resources, such as conference rooms, video equipment and so on. The Group View feature displays the agendas of the included users and resources, with unavailable time marked in red and mutually free time clearly indicated.

Resource Coordination

Administrators can designate a shared property, such as a conference room or projector, as a resource, available for all connected users to reserve. You can perform a search for a resource based on a set of parameters (location, size, resource type), and *invite* the resource as you would any other user, thereby booking the resource and making it unavailable for other users to book during that time. Resources can be set up to be reserved on a first-come, first-served basis. Oracle Calendar also supports the booking of resources that require approval from an administrator. If you book a resource that requires approval, then an e-mail is sent to the resource's administrator, who then approves or rejects your request.

Access Rights

You can control how much of your calendar can be accessed by other users, through the use of access rights. For example, you can grant one user access rights to view all your agenda entries marked as Normal, while you grant another user access rights to view all your agenda entries marked as Normal and Personal. Granting rights to other users allows them to create, modify, and reply to calendar events on your behalf. When granting rights, you can choose which type of calendar entries a particular person has permission to create and modify. For example, you can grant one person the right to modify your meetings, notes, day events, and tasks, while you grant another person the right to modify only your tasks.

Standalone Deployment Capabilities

When the Oracle Calendar server is installed, by default it is integrated with the Oracle Internet Directory server and other components of Oracle Collaboration Suite. However, the Oracle Calendar server can also be deployed as a *standalone* application.

In such installations, the Oracle Calendar server can be configured to use either an external or an internal directory. With an external directory, all user information is stored in a third-party LDAP directory server. With an internal directory, all user information is stored in the Oracle Calendar server database.

For more information on deploying the Oracle Calendar as a standalone product, see chapter 5, "Deploying Oracle Calendar" in *Oracle Collaboration Suite Deployment Guide*.

Alerts, Notifications, and Web Conferencing Integration

As part of a suite of collaborative applications, Oracle Calendar enables users to schedule and join Web conferences directly from their calendars as well as send notifications. You can also set alert reminders for an alternate e-mail address, or even a wireless account.

Data Synchronization with Oracle Calendar Sync

Oracle Calendar Sync synchronizes your Oracle Calendar data with your Personal Digital Assistant (PDA) using Palm Desktop for Windows or Macintosh (Palm devices), or Microsoft ActiveSync (Pocket PC devices). This enables you to download meetings, contacts, daily notes, day events, holidays, and tasks to your PDA. You can make updates and then synchronize them back to Oracle Calendar through your device's synchronization process.

9.1.2 Oracle Connector for Outlook

Oracle Connector for Outlook presents the enterprise collaboration market with a unique proposition, by enabling enterprises to offer their users access to a broad array of collaborative functionality, including e-mail, voice mail, calendar, directory, Web conferencing, and wireless services, through the convenience of the familiar Microsoft Outlook interface.

9.1.3 Oracle Collaboration Suite 10*g* Real-Time Collaboration

Oracle Collaboration Suite 10*g* Real-Time Collaboration (Oracle Real-Time Collaboration) tools let you communicate and collaborate in real time with coworkers and business contacts. This section covers the following Oracle Real-Time Collaboration products:

- **Oracle Web Conferencing:** Lets customers, employees, teams, and partners meet online and collaborate in real time, from one-to-one instant conferences to large,

scheduled Web seminars. Oracle Web Conferencing is reliable, secure, flexible, and scalable, and can be customized to support various lines of business within your company.

- **Oracle Messenger:** A full-featured Instant Messaging system that lets you exchange chat messages with another user and participate in chat conferences with multiple users.
- **Oracle Real-Time Collaboration Add-in for Microsoft Office:** Provides a convenient way to schedule Web conferences, start instant conferences, share Microsoft Office documents in a Web conference, and chat with other Oracle Messenger users from within Microsoft Office applications.

9.1.4 Oracle Drive

Oracle Drive is the desktop client for Oracle Content Services. Oracle Drive enables you to access content (files) and file properties through a mapped drive in Windows Explorer, from any Windows applications, and Microsoft Office applications. Content is also accessible through a Web browser. The Oracle Drive client uses the WebDAV protocol to access Oracle Content Services.

See Also: Section 12.3.3, "Using Oracle Drive as a WebDAV Client"

9.2 Prerequisites

To manage Oracle Calendar tasks in Microsoft Outlook, you need the following prerequisite software:

- **Oracle Collaboration Suite** or **Oracle Calendar Standalone Installation**

  Refer to *Oracle Collaboration Suite Installation Guide for Microsoft Windows* for details.

- **Oracle Connector for Outlook**

  Refer to Section F.2, "Installing Oracle Connector for Outlook" in *Oracle Collaboration Suite Installation Guide for Microsoft Windows* for details.

  After Oracle Connector for Outlook is installed, you will be able to see the corporate address book, corporate shared and public folders, and the corporate calendar in the left navigation pane of your Microsoft Outlook window.

  Caution: Installing Oracle Connector for Outlook removes all customizations you may have made earlier in Microsoft Outlook.

- **Oracle Real-Time Collaboration**

  Refer to *Oracle Collaboration Suite Installation Guide for Microsoft Windows* for details.

- **Oracle Real-Time Collaboration Add-in for Microsoft Office**

  Refer to *Oracle Collaboration Suite Installation Guide for Microsoft Windows* for details.

- **Microsoft Outlook** (version 2000 or later)

9.3 Step-by-Step Procedures

In this section, you will learn how to manage Oracle Calendar information and tasks, and Oracle Real-Time Collaboration tools, from within Microsoft Outlook. It contains the following subsections:

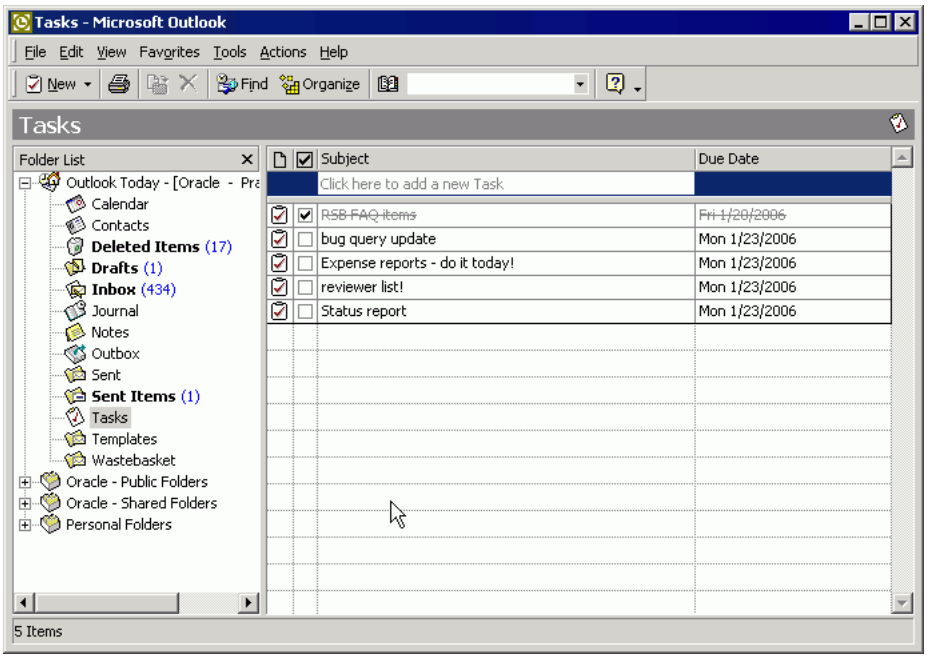
- Creating Tasks
- Scheduling Meetings
- Viewing Contact Information
- Chatting with Other Users
- Starting an Instant Conference
- Viewing Conference Archives

9.3.1 Creating Tasks

An employee at Star Loan Company, John Steinbeck, is responsible for handling loan requests and approving loans. He has multiple tasks to perform in a typical workday. To help keep track of important tasks, he uses Microsoft Outlook with Oracle Calendar and creates tasks that also serve as reminders. To create tasks, perform the following steps:

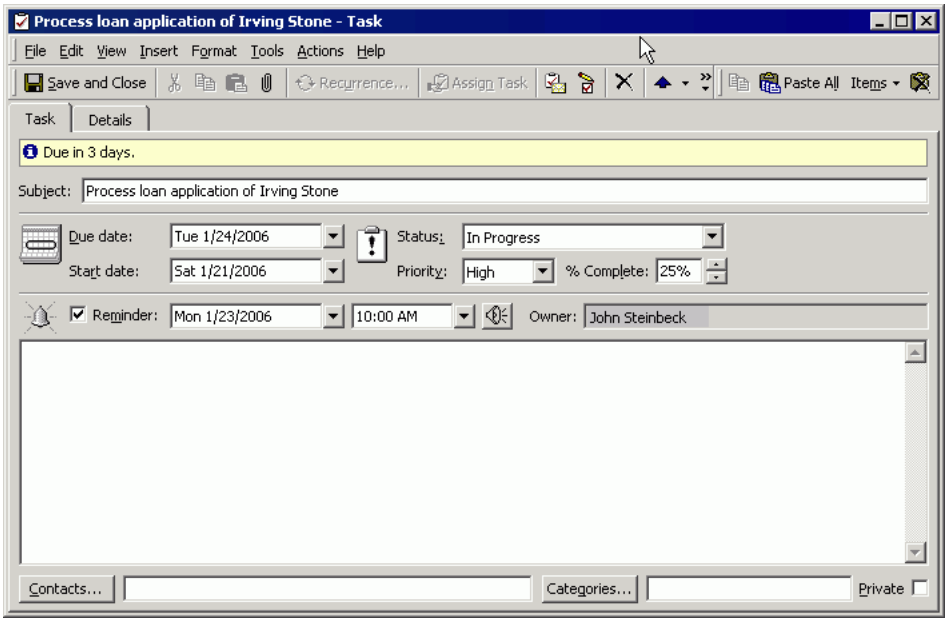
1. In Microsoft Outlook, click **Tasks** in the left navigation pane. This displays the tasks created earlier, as shown in Figure 9–1.

Figure 9–1 Viewing Oracle Calendar Tasks Using Microsoft Outlook



2. Click **New** from the Microsoft Outlook toolbar, or right-click anywhere in the tasks list pane, and select **New Task**. This displays the Task window, as shown in Figure 9–2.

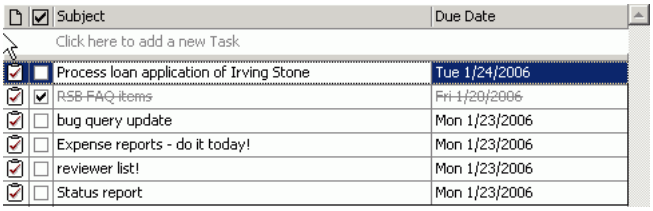
Figure 9–2 New Outlook Task



3. After entering the required details, click **Save and Close**.

   A new task is created in Oracle Calendar and appears in your Tasks list, as shown in Figure 9–3. You can keep track of the progress of this task and updated it if necessary. You can mark a task as completed when done.

Figure 9–3 New Calendar Task Entry Viewed in Microsoft Outlook



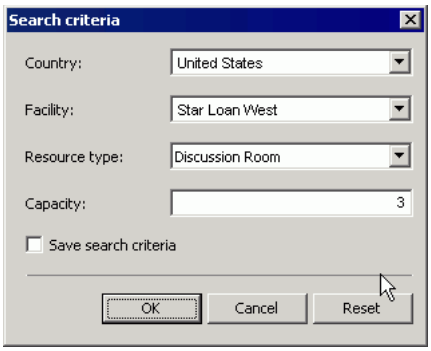
9.3.2 Scheduling Meetings

Let us assume a sample scenario where a manager at the Star Loan Company, David Cook, wants to communicate a new cutting edge loan program to the loan approvers that report to him, James Cooper and John Steinbeck. He wants to schedule a meeting on Wednesday with James and John in a conference room on the third floor. He uses the Oracle Calendar feature in Microsoft Outlook to organize this meeting.

There are multiple ways to schedule meetings. You can do this from Oracle Calendar, from the Microsoft Outlook File menu, or by clicking **Resources** in your Microsoft Outlook toolbar. The third option is outlined in the following steps:

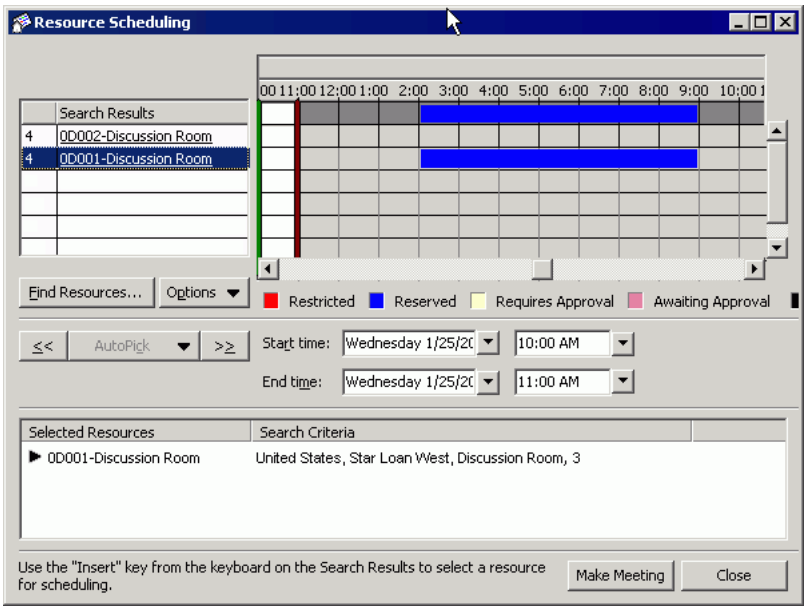
1. In Microsoft Outlook, click **Resources** from the toolbar. This displays the Search criteria dialog box, as shown in Figure 9–4.

Figure 9–4 Search Resources



2. Specify location details and resource type, and click **OK**. This displays the Resource Scheduling dialog box, as shown in Figure 9–5.

Figure 9–5 Resource Scheduling in Microsoft Outlook Using Oracle Calendar



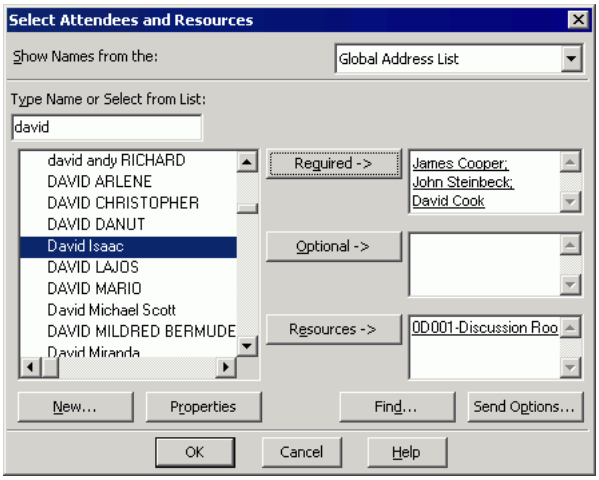
(Optional) You can click **Find Resources** to change location and find the relevant resources for that location.

3. From the Search Results pane, right-click the resource you want to reserve for your meeting, and choose **Select resource**.

4. Select the start and end times for your meeting, and click **Make Meeting**. This displays a window, which you can use to create the meeting, and add attendees.

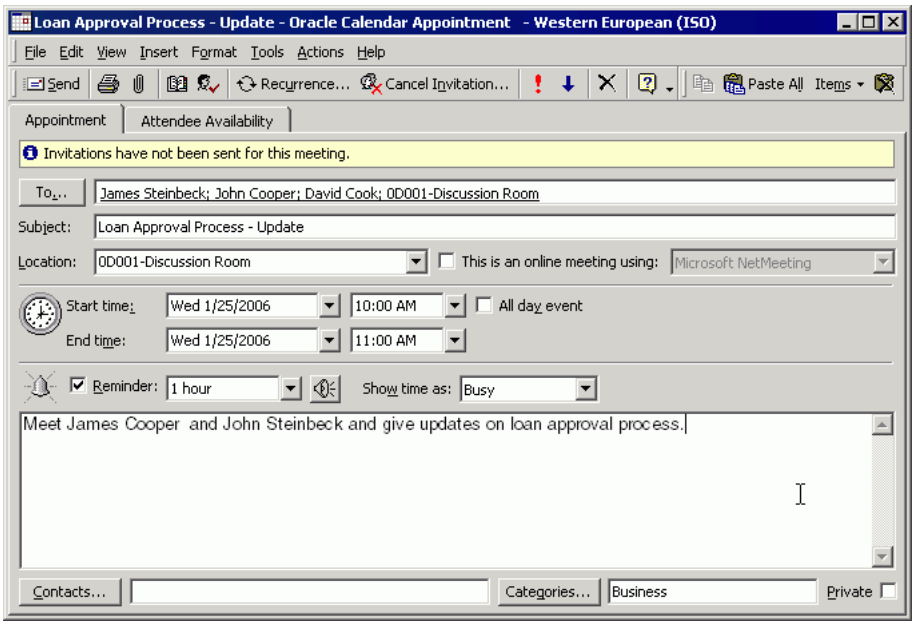
5. Click **To**. This displays the Select Attendees and Resources dialog box, as shown in Figure 9–6. Select the meeting attendees and resources. You can add users either from your local Contacts list or from the corporate address book.

Figure 9–6 Selecting Meeting Attendees



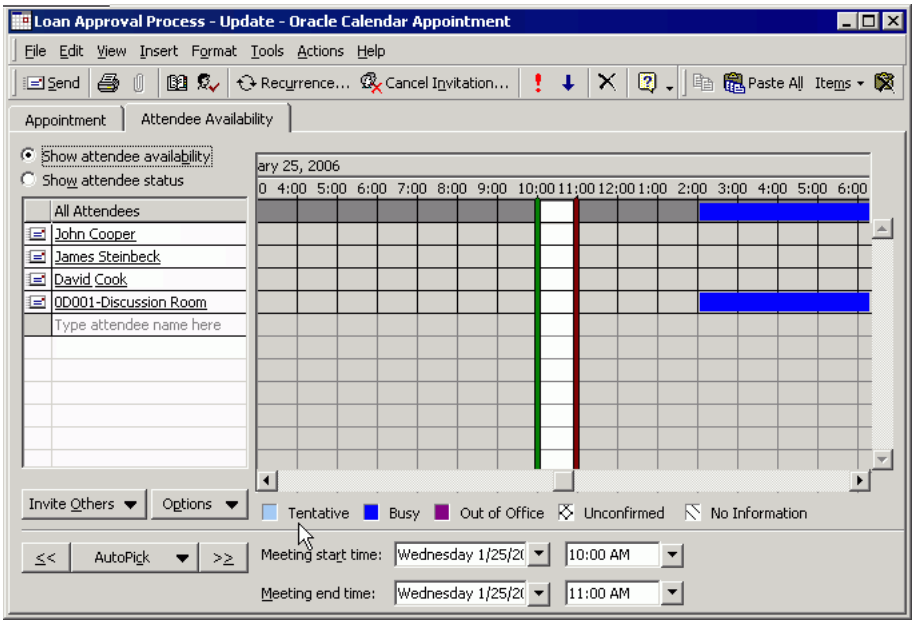
6. Click **OK**. This displays the Calendar Appointment dialog box, as shown in Figure 9–7.

Figure 9–7 Creating a Meeting in Microsoft Outlook Using Oracle Calendar



7. You can view the availability of attendees at the scheduled meeting time, by clicking the **Attendee Availability** tab, as shown in Figure 9–8.

Figure 9–8 Viewing Attendee Availability in Microsoft Outlook Using Oracle Calendar



8. After you have verified that all attendees are available at the scheduled time, click the **Appointment** tab, and then click **Send**.

   An e-mail message is sent to all attendees informing them about the meeting. Oracle Calendar also updates the schedules of the attendees to show the meeting time as busy.

9.3.3 Viewing Contact Information

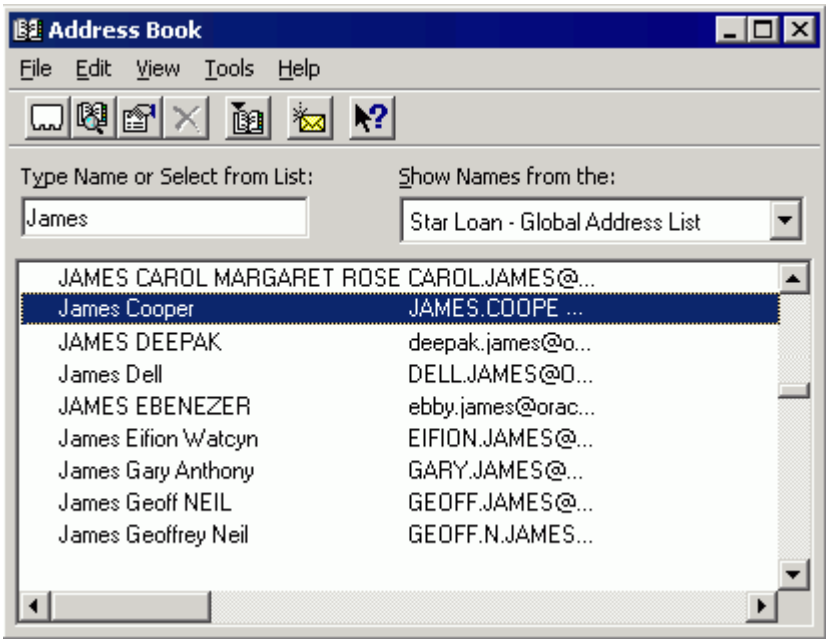
You can view up-to-date contact information stored either in Oracle Internet Directory or Microsoft Exchange, using Microsoft Outlook.

> **Note:** If you are using Microsoft Exchange with Microsoft Active Directory, then you must synchronize Oracle Internet Directory and Microsoft Active Directory. For details, refer to Chapter 10, "Provisioning User Identity Information and Alerting Microsoft Outlook Contacts".

Assume that the address of James Cooper has changed. This change, when made in the HR application, is synchronized with Oracle Internet Directory, which in turn, synchronizes with, for instance, Microsoft Exchange. To look up this updated information, perform the following steps:

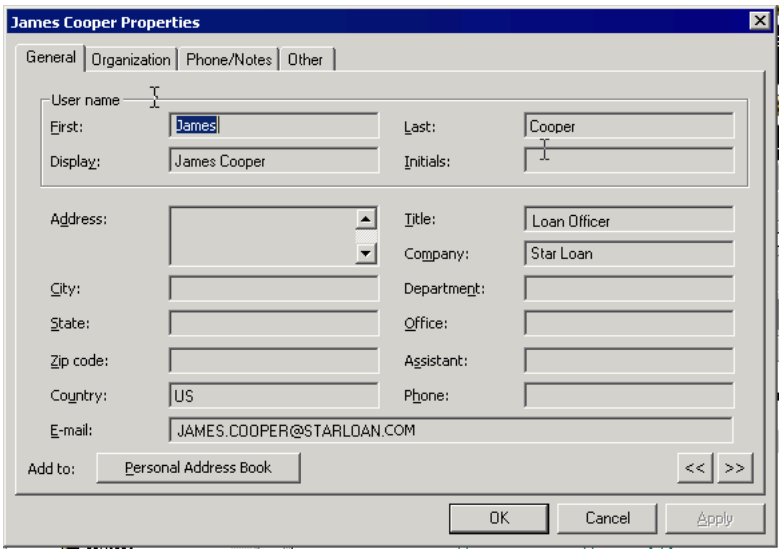
1. In Microsoft Outlook, press Ctrl+Shift+B, or from the menu bar, click **Tools**, and then **Address Book**. This displays the Address Book dialog box, as shown in Figure 9–9.

Figure 9–9 Address Book



2. From Show Names from the list, as shown in Figure 9–9, select your corporate address book.
3. In the Type Name or Select from List field, type the name of the contact for which you are searching. The list of names is automatically filtered as you type.
4. To view the properties of a contact, select the contact entry in the list. Click **File**, and then **Properties**. Alternatively, you can also right-click the contact and select **Properties**. The Contact Properties dialog box is displayed, as shown in Figure 9–10.

Figure 9–10 Contact Properties



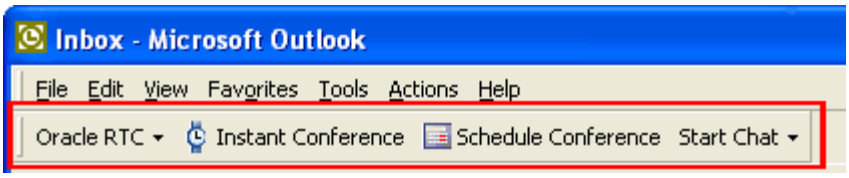
5. Click **OK** to exit the dialog box.

9.3.4 Chatting with Other Users

Oracle Real-Time Collaboration Add-in for Microsoft Office lets you chat with other Oracle Messenger users from within Microsoft Office applications, such as Microsoft Excel, Microsoft Outlook, Microsoft PowerPoint, or Microsoft Word.

Once you are logged in to Oracle Real-Time Collaboration, you can download and install the Oracle Real-Time Collaboration Add-in for Microsoft Office. Installing the Oracle Real-Time Collaboration Add-in for Microsoft Office adds the Oracle Real-Time Collaboration toolbar to all Microsoft Office applications, as shown in Figure 9–11.

Figure 9–11 Oracle Real-Time Collaboration Toolbar in Microsoft Outlook



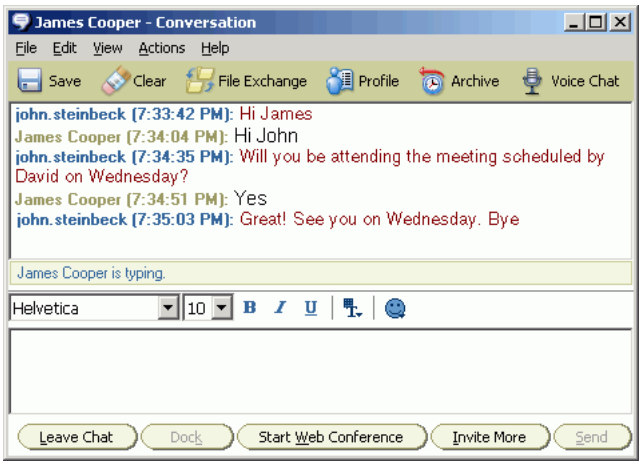
Notes:

- You must install and be signed in to Oracle Messenger if you want to chat with other Oracle Messenger users from Microsoft Office applications.
- If you have already installed Oracle Connector for Outlook, and then you install the Oracle Real-Time Collaboration Add-in for Microsoft Office, then you won't be able to use this Add-in to schedule Web conferences. You can continue to schedule Web conferences using Oracle Connector for Outlook.

To chat with other Oracle Messenger users from within Microsoft Outlook, perform the following steps:

1. From the Oracle Real-Time Collaboration toolbar in Microsoft Outlook, click the arrow next to **Start Chat**. This displays a list of other Oracle Messenger users who are online.
2. Select any user. This displays an instant message window, as shown in Figure 9–12.

Figure 9–12 Oracle Messenger Instant Messaging Window



3. Close the window when finished, and return to Microsoft Outlook.

9.3.5 Starting an Instant Conference

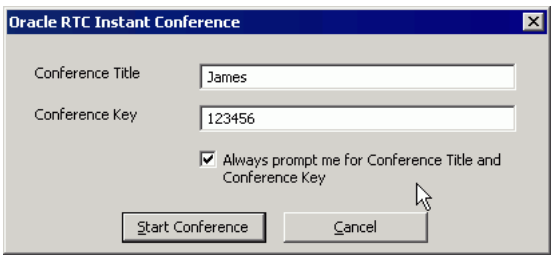
With Oracle Real-Time Collaboration Add-in for Microsoft Office, you can schedule Web conferences and start instant conferences from within Microsoft Office applications such as Microsoft Excel, Microsoft Outlook, Microsoft PowerPoint, or Microsoft Word.

James Cooper is processing Irving Stone's loan application. While verifying the application, he comes across an invalid entry in the form. He is not sure how to proceed. He wants to consult John Steinbeck, his colleague, and also show him some more documents on his computer. With the Oracle Real-Time Collaboration Add-in for Microsoft Office, he can start an instant conference and share documents that John can view.

To start an instant conference from within Microsoft Outlook, perform the following steps:

1. From the Oracle Real-Time Collaboration toolbar in Microsoft Outlook, click **Instant Conference**. This displays the Oracle RTC Instant Conference dialog box, as shown in Figure 9–13.

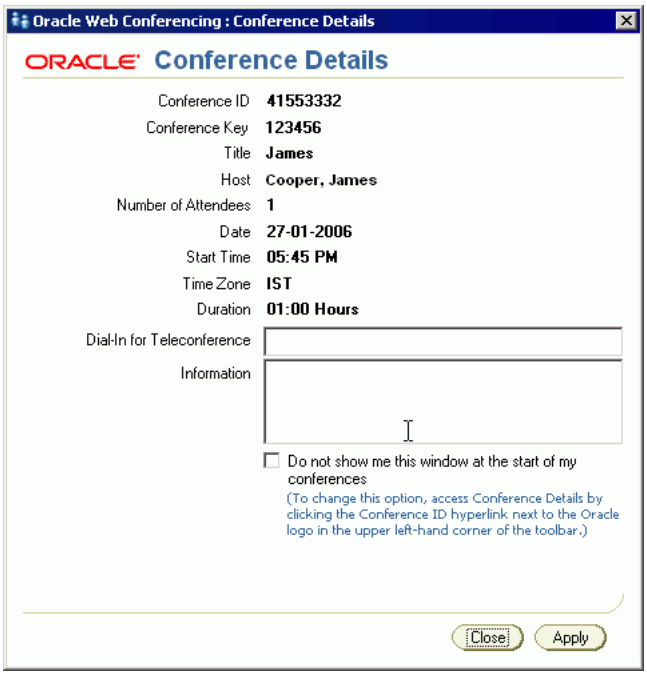
Figure 9–13 Instant Conference Details



2. Change the conference details from the default values if you desire, and then click **Start Conference**.
3. A new browser window opens displaying the Oracle Web Conferencing Console initialization details. If a permission dialog box appears, click **Yes** to install Oracle Web Conferencing.

4. After the console is initialized, a dialog box with the Web Conference details is displayed, as shown in Figure 9–14.

Figure 9–14 Oracle Web Conference Details



5. You must then provide the details of the conference to anyone who wants to join your conference, for example, John Steinbeck. Refer to the steps in Section 9.3.4, "Chatting with Other Users" for details on how you can use Oracle Messenger to send these details to other users.
6. Click **Apply** to exit the dialog box. This displays the Oracle Web Conferencing console at the top of your window.
7. By default, nothing is shared to others. From the Share list, select the applications to share, or even your entire desktop.
8. When other users join your conference, you see a notification at the bottom of your window.
9. When finished, close the console, and return to your Microsoft Office application.

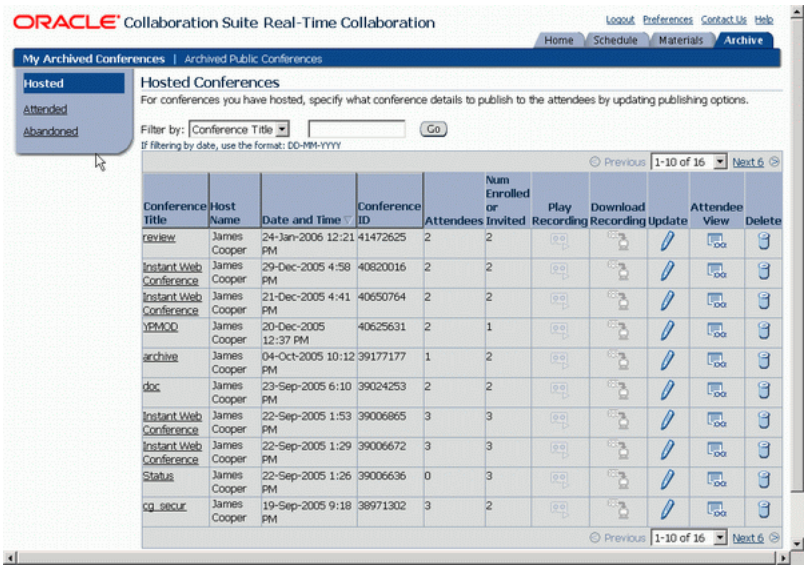
9.3.6 Viewing Conference Archives

With Oracle Real-Time Collaboration Add-in for Microsoft Office, you can easily view details about the conferences you have hosted in the past. Not only can you view details about your own past conferences, but also other public conferences.

To go to the archives, and view details about your past conferences, perform the following steps:

1. From the Oracle Real-Time Collaboration toolbar in Microsoft Outlook, click **Oracle RTC**, and then select **My Archives**. A new browser window opens with details of your past Web conferences, as shown in Figure 9–15.

Figure 9–15 Archived Conferences



2. If you are looking for a specific conference, you can use the Filter by list. Select an item from the Filter by list, type the relevant text in the text field, and then click **Go**.
3. To view previously hosted public conferences, click **Archived Public Conferences**.

9.4 Related Documentation

Refer to the following documents at the Oracle Collaboration Suite 10*g* Release 1 (10.1.2) home page at `http://www.oracle.com/pls/cs101/homepage`:

- *Oracle Collaboration Suite Concepts Guide*
- *Oracle Real-Time Collaboration Administrator's Guide*
- *Oracle Calendar Administrator's Guide*

10

Provisioning User Identity Information and Alerting Microsoft Outlook Contacts

This chapter describes how you can do the following:

- Get up-to-date information about employees whose identity information is integrated in Microsoft Outlook contacts.
- Create and receive organization alerts that are sent when any user identity information is changed.

It contains the following sections:

10.1 Overview

Whether user information is stored in Microsoft Active Directory, or in Oracle Internet Directory, it is expected that information in Microsoft Outlook Contacts is always up-to-date. When changes are made in an enterprise application, such as Sales Force Automation, or a Human Resources Management Systems, updates to the directory should be reflected immediately in Microsoft Outlook Contacts. In addition, alerts about organizational changes may need to be sent as soon as they happen.

Oracle Identity Management enables you to reduce administrative time and costs by integrating your applications and directories, including third-party LDAP directories, with Oracle Internet Directory. It does this by using Oracle Directory Integration and Provisioning.

Throughout the integration process, Oracle Directory Integration and Provisioning ensures that the applications and other directories receive and provide the necessary information in a reliable way. Oracle provides centralized security administration by integrating components with Oracle Identity Management. Similarly, Microsoft provides centralized security administration in Microsoft Windows by integrating Microsoft applications with Microsoft Active Directory. If your environment uses both Oracle Identity Management and Microsoft Active Directory, then, to enable interoperability between the two systems, you must synchronize their data. You can do this by using Oracle's Active Directory Connector.

This chapter describes what must be done to ensure contact data in Microsoft Outlook Contacts is always up-to-date in an environment where both Microsoft Exchange and

Microsoft Active Directory are used together with Oracle Internet Directory. Microsoft Active Directory and Oracle Internet Directory must be synchronized (integrated) in order to ensure that both have the same up-to-date contact data. This chapter describes how to achieve accurate and timely directory synchronization.

In addition, this chapter also covers the steps that must be performed if an Oracle solution is used to send alerts about organizational changes. These steps include configuring Oracle Directory Integration and Provisioning and Oracle BPEL Process Manager to generate organization alerts whenever user identity information changes in Oracle Internet Directory.

When user identity information changes in an enterprise application, Oracle Internet Directory is updated with this information. Using Active Directory Connector to synchronize Microsoft Active Directory and Oracle Internet Directory ensures that Microsoft Outlook Contacts data is always up-to-date.

Note: If you are using Oracle Collaboration Suite applications with Microsoft Outlook, then you must use Oracle Connector for Outlook, to enable interoperability of management tasks in Oracle Collaboration Suite applications with Microsoft Outlook. Refer to Chapter 9, "Managing Tasks and Collaborating in Microsoft Outlook" for more information.

The following topics describe the components used in enabling interoperability between Oracle Identity Management and Microsoft Active Directory, and a few related concepts.

Oracle Internet Directory

Oracle Internet Directory is a critical component of Oracle Application Server management and security infrastructure. It ensures that user accounts and groups are managed centrally through the LDAP Version 3 standard. Oracle Application Server enables user accounts and groups to be created centrally in Oracle Internet Directory and shared across all components in Oracle Application Server. When users log in, they are authenticated once by Oracle Application Server Single Sign-On against their Oracle Internet Directory credentials, and can thereby access multiple applications seamlessly.

Oracle Directory Integration and Provisioning

Oracle Directory Integration and Provisioning enables users to synchronize data between various directories and Oracle Internet Directory. Oracle Directory Integration and Provisioning is a set of services and interfaces that makes it possible to develop synchronization solutions with third-party directories and other enterprise repositories. Oracle Directory Integration and Provisioning includes a connector, called Active Directory Connector, for out-of-the-box synchronization with Microsoft Active Directory.

Oracle Application Server Single Sign-On

OracleAS Single Sign-On enables users to access Oracle Web-based components by logging in only once. Oracle components delegate the login function to the OracleAS Single Sign-On server. When a user first logs in to an Oracle component, the component directs the login to the OracleAS Single Sign-On server. The OracleAS Single Sign-On server compares the credentials entered by the user to those stored in Oracle Internet Directory. After verifying the credentials, the OracleAS Single Sign-On

server grants the user access to all components the user is authorized to use throughout the current session.

OracleAS Single Sign-On enables native authentication in a Microsoft Windows environment, using the user's Kerberos credentials.

> **See Also:** *Oracle Identity Management Integration Guide* for details about configuring Windows native authentication.

Directory Synchronization

Synchronization, which is a service of Oracle Directory Integration and Provisioning, enables you to make changes persist between Oracle Internet Directory and connected directories, like Microsoft Active Directory. For all directories to both use and provide only the latest data, each directory must be informed of change made in the other connected directories. Synchronization ensures that any change to directory information is kept consistent.

Connectors for Directory Synchronization

To synchronize between Oracle Internet Directory and a connected directory, Oracle Directory Integration and Provisioning relies on a prepackaged connectivity solution called a connector. Minimally, this connector consists of a directory integration profile containing all the configuration information required for synchronization, including the following:

- Direction of synchronization
- Type of interface
- Mapping rules and formats
- Connection details of the connected directory
- Other information

Active Directory Connector

Oracle Directory Integration and Provisioning includes connectors to synchronize Oracle Internet Directory with other LDAP directories or identity stores. One of its connectors, Active Directory Connector, is designed to synchronize Oracle Internet Directory with Microsoft Active Directory.

Active Directory Connector enables any the following:

- Establishing one-way or two-way synchronization with Microsoft Active Directory.
- Synchronizing a specific subset of attributes. Configure appropriate mapping rules in the connector profile to do this.
- Synchronizing with multiple Microsoft Active Directory domains. Synchronize changes with an individual domain or an entire Microsoft Active Directory environment by using Microsoft Global Catalog Server.

Using Active Directory Connector for Microsoft Exchange Provisioning

Active Directory Connector, available as part of Oracle Identity Management release 10.1.2, can be used for provisioning users to Microsoft Exchange. This is applicable in deployments having Microsoft Active Directory Server 2000 or later as their identity store. Provisioning users to Microsoft Exchange Server involves creating a user account in the corresponding Microsoft Active Directory with Microsoft Exchange-specific user attributes. These attributes contain details about the Microsoft

Exchange server, mail transfer agent, proxy address, and so on. To configure provisioning to Microsoft Exchange, the default mapping rules for Active Directory Connector are enhanced to include Microsoft Exchange-specific mapping rules.

10.2 Prerequisites

In the scenario described in this chapter, it is assumed that you have deployed Microsoft Exchange 2000 or later with Microsoft Active Directory as its back-end repository, and Microsoft Office, specifically Microsoft Outlook as the e-mail client. The scenario expects the following Oracle software components to be installed:

- Oracle Application Server Infrastructure 10*g* Release 2 (10.1.2.0.2)
- An Oracle Directory Integration and Provisioning patch

  This patch is required to perform Microsoft Exchange specific mappings in express configuration. Perform the following steps to install the patch:

  1. Download an Automated Release Update (ARU) for bug 5066404.

     You can download ARUs from Oracle*Metalink* at `http://metalink.oracle.com`.

  2. Follow the instructions provided in the Readme for the patch.

- (Optional) Oracle BPEL Process Manager 10*g* Release 2 (10.1.2.0.2)

  To configure organization alerts using Oracle BPEL Process Manager, as shown in Section 10.3.2, "Procedure 2: Configuring BPEL-Based Organization Alerts", the following software must be installed:

  – To configure the BPEL process, you must use the Oracle JDeveloper BPEL Designer that is part of the standalone version of Oracle BPEL Process Manager. You can install this by selecting the BPEL Process Manager for Developers option during installation.

  – To deploy the sample BPEL process described in Section 10.3.2, "Procedure 2: Configuring BPEL-Based Organization Alerts", you must install BPEL Process Manager for Oracle Application Server middle tier. This requires a J2EE and Web Cache installation type of Oracle Application Server 10*g* Release 2 (10.1.2.0.2) in the same Oracle home directory. Additionally, you must have Oracle Application Server Metadata Repository installed.

    See Also: *Oracle Application Server BPEL Process Manager Installation Guide* on the Oracle BPEL Process Manager page on OTN at

    `http://www.oracle.com/technology/bpel`

- The support files and folders in the `identitymanagement` demonstration folder. Refer to Accessing the Demonstration Support Files in the Preface for details about the demonstration support files. These files and folders are listed and described in Table 10–1.

Table 10–1 Identity Management Files and Folders

File or Folder	Description
`identitymanagement/IdentityNotification`	Folder that contains the BPEL process called `IdentityNotification`. Copy the `IdentityNotification` folder, located in the `identitymanagement` folder, to the *`BPEL_ORACLE_HOME`*`/integration/orabpel/samples/demos` directory.

Table 10–1 (Cont.) Identity Management Files and Folders

File or Folder	Description
`identitymanagement/sql`	Folder that contains the SQL script to create the schema, and corresponding packages to propagate organization alerts.
`identitymanagement/sql/notificationsetup.sql`	SQL script file used to create a schema for organization alerts.

Note: Download the contents of the `identitymanagement` folder into a new folder named `identitymanagement` under the *`BPEL_ORACLE_HOME`*`\integration\orabpel\samples\demos` folder, for example, `C:\OraBPELPM_1\integration\orabpel\samples\demos\identitymanagement\`.

10.3 Step-by-Step Procedures

This section will provide the following procedures, based on example data:

- Procedure 1: Synchronizing Enterprise Identity Information

  Configure an enterprise environment where the user identity information is always consistent and up-to-date between Oracle Internet Directory and Microsoft Exchange.

- Procedure 2: Configuring BPEL-Based Organization Alerts

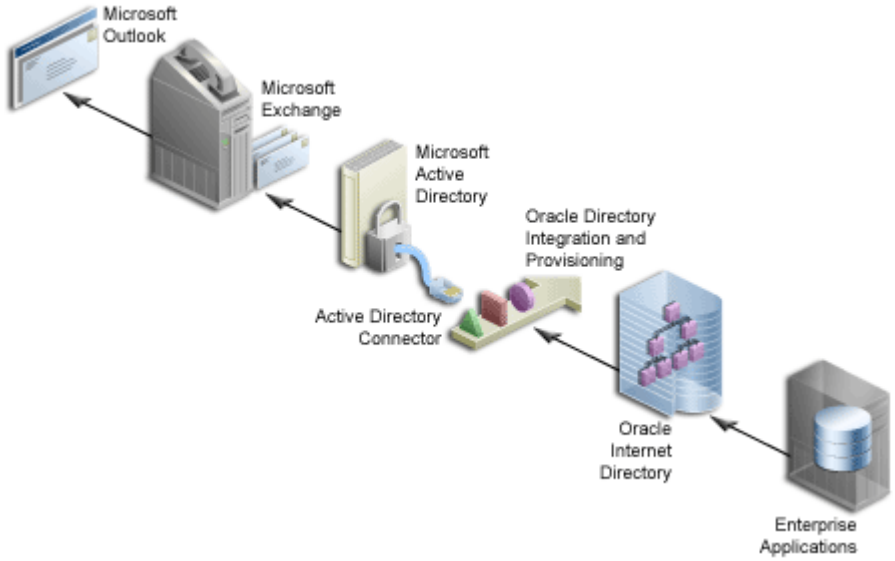
  Use Oracle Directory Integration and Provisioning and Oracle BPEL Process Manager to create and receive organization alerts that are sent when user identity information changes.

10.3.1 Procedure 1: Synchronizing Enterprise Identity Information

User identity information can be stored in many places, but some common directories are Oracle Internet Directory and Microsoft Active Directory. Microsoft Exchange version 2000 or later uses Microsoft Active Directory as its identity store. This procedure shows you how to synchronize Oracle Internet Directory with Microsoft Active Directory.

A very common enterprise scenario is illustrated in Figure 10–1. Enterprise applications, for example, the Human Resources Management System, update user identity information in Oracle Internet Directory. Oracle Directory Integration and Provisioning synchronizes user identity information between Oracle Internet Directory and Microsoft Active Directory by using Active Directory Connector. Microsoft Active Directory is used by Microsoft Exchange as the identity store. Users in the enterprise use Microsoft Outlook to read their e-mail and to get up-to-date contact information. Besides setting up the Microsoft Active Directory synchronization, you must perform some additional configuration to provision Microsoft Exchange-specific attributes.

Figure 10–1 Oracle Internet Directory Interoperability with Microsoft Active Directory and Microsoft Exchange



The subsequent example describes how Microsoft Exchange and Microsoft Active Directory may need to be synchronized with Oracle Internet Directory.

The context of this example is the Union Loan Company, which has hundreds of employees. Union Loan Company uses Oracle Application Server for its enterprise applications, Oracle Internet Directory for central identity management, and Microsoft Exchange server with Microsoft Active Directory for e-mail and contact information. Employees use Microsoft Outlook.

Assume that Union Loan Company hires John Steinbeck as a new loan approver. John's personal information is entered into the Human Resources Management System, and provisioned to Oracle Internet Directory. Directory synchronization is set up between Oracle Internet Directory and Microsoft Active Directory. Additional configuration modifications ensure that all the Microsoft Exchange-specific attributes were modified appropriately. As a result, John's profile now shows up in Microsoft Outlook and John's manager and co-workers can now quickly look up his telephone number. Directory integration ensures that change in user identity information, such as a change of John's telephone number, will almost immediately be visible in everyone's Microsoft Outlook Contacts.

To provision users in Microsoft Exchange, that is, to *push* data from Oracle Internet Directory to Microsoft Active Directory, you must use Active Directory Connector to synchronize all the attributes of the data that is exchanged. Figure 10–1 illustrates how this interoperability works. To ensure that this data is ready to be used by Microsoft Exchange, however, you must make some changes to the mapping rules. To synchronize between Oracle Internet Directory and Microsoft Active Directory, a directory integration profile for synchronization must be created that contains all the configuration information required for synchronization to Microsoft Exchange. To do this, perform the steps in the following sections:

- Configuring Microsoft Active Directory Synchronization Profiles for Microsoft Exchange
- Enabling the Profiles for Synchronization
- Verifying the Synchronization

10.3.1.1 Configuring Microsoft Active Directory Synchronization Profiles for Microsoft Exchange

The Oracle Directory Integration and Provisioning server includes an express configuration option that you can run with Directory Integration and Provisioning Assistant. This type of configuration uses certain predefined values and creates two synchronization connector profiles, one for import and one for export, pointing to Microsoft Active Directory.

> **See Also:** *Oracle Identity Management Integration Guide* for more information about express configuration.

Run express configuration by using the Directory Integration and Provisioning Assistant tool as follows:

1. Start the Oracle Directory Integration and Provisioning Server Administration tool by entering the following command:

   ```
   INFRA_ORACLE_HOME/bin/dipassistant expressconfig -h <oid_host> -p <oid_non-SSL_
   port> -configset <DIP_configuration_set> -3rdpartyds adforexchange
   ```

   where, `DIP_configuration_set` refers to the configuration set for Oracle Directory Integration and Provisioning. The default value is `1`. For example:

   ```
   INFRA_ORACLE_HOME/bin/dipassistant expressconfig -h m1.abc.com -p 389
   -configset 1 -3rdpartyds adforexchange
   ```

2. You are prompted for information about the setup. Enter appropriate values as described in Table 10–2.

Table 10–2 Parameters for Running the dipassistant Tool

Parameter	Description
OID user name	Oracle Internet Directory user name. Specify the super user, that is, `cn=orcladmin`, or any user that is a member of the Directory Integration and Provisioning Administrators group (`cn=dipadmingrp, cn=odi, cn=oracle internet directory`).
OID password	Password for the Oracle Internet Directory user.
Active Directory Host	Host name of the Microsoft Active Directory.
Active Directory Port	Microsoft Active Directory port number.
Account Name	User name of a privileged Microsoft Active Directory user. **Note**: To synchronize deletions, you must have the necessary administrative privileges in Microsoft Active Directory, for example, `administrator@MyCompany.com`.
Account Password	Microsoft Active Directory password.
Connector name	Name for the connector. Depending on the name specified here, two profiles, an import and an export profile, are created with names as <*connector name*>`Import` and <*connector name*>`Export` respectively. For example, if you specify the name `test`, then the tool creates two profiles: `testImport` and `testExport`.
Options to configure ACLs	ACL configuration options. Enter `y` to update the access control policies for the default realm user search base, to give Oracle components the required access. Default value is `n`.

Running express configuration creates two synchronization connector profiles pointing to Microsoft Active Directory. The <*Connectorname*>IMPORT profile maintains the configuration information for importing user identity information from Microsoft Active Directory to Oracle Internet Directory and the <*Connectorname*>EXPORT profile maintains configuration details for exporting changes from Oracle Internet Directory to Microsoft Active Directory. By default, both profiles are configured to synchronize the data between the connected directories in one minute intervals. At the end of the express configuration, the necessary mapping rules are configured to handle Microsoft Active Directory and Microsoft Exchange-specific attributes in case of export of users from Oracle Internet Directory to the Microsoft Active Directory.

10.3.1.2 Enabling the Profiles for Synchronization

You now must enable the export and import profiles by performing the following steps:

1. If user identity information changes are made only in Human Resources Management System, from which the identity details are synchronized to Oracle Internet Directory, then users are synchronized from Oracle Internet Directory to Microsoft Active Directory. In this case, enable the export synchronization profile by using the Oracle Directory Integration and Provisioning Server Administration tool with the `modifyprofile` option. The following Oracle Directory Integration and Provisioning assistant command enables an export profile:

   ```
   INFRA_ORACLE_HOME/bin/dipassistant modifyprofile -profile <profile_name> [-host
   <oid_host>] [-port <oid_port>] [-dn "<oid_bindDN>"] [-passwd <oid_bindDN_
   password>] odip.profile.status=ENABLE
   ```

   The following is an example of a command used to enable an export profile named `testExport`:

   ```
   INFRA_ORACLE_HOME/bin/dipassistant modifyprofile -profile testExport -host
   m1.abc.com -port 389
   -dn "cn=orcladmin" -passwd welcome1 odip.profile.status=ENABLE
   ```

   In addition, if user entries created from a Human Resources Management System can be modified from Microsoft Active Directory and Microsoft Exchange server, then you must synchronize the data from Microsoft Active Directory to Oracle Internet Directory. In this case, you must enable the import profile. The following Oracle Directory Integration and Provisioning assistant command enables an import profile:

   ```
   INFRA_ORACLE_HOME/bin/dipassistant modifyprofile -profile <profile_name> [-host
   <oid_host>] [-port <oid_port>] [-dn "<oid_bindDN>"] [-passwd <oid_bindDN_
   password>] odip.profile.status=ENABLE
   ```

   The following is an example of a command used to enable an import profile named `testImport`:

   ```
   INFRA_ORACLE_HOME/bin/dipassistant modifyprofile -profile testImport -host
   m1.abc.com -port 389
   -dn "cn=orcladmin" -passwd welcome1 odip.profile.status=ENABLE
   ```

2. Start the Oracle Directory Integration and Provisioning server as follows if this is not already running with the configuration set that contains the Microsoft Exchange profiles:

   ```
   oidctl connect=<oid_metadatarep_connect_string> server=odisrv
   instance=<instance_number> configset=<configuration_set_number>
   ```

```
[flags="flagname=<value> ..."]  {start | stop | restart}
```

For example:

```
oidctl connect=dbs1 server=odisrv instance=1 configset=1
flags="host=ldaphost.company.com port=389" start
```

10.3.1.3 Verifying the Synchronization

To verify that the synchronization between Oracle Internet Directory and Microsoft Active Directory is working properly, perform the following steps:

1. After you have enabled the profiles, you can verify the status of synchronization by running the following command (the default interval for change synchronization is 1 minute):

   ```
   INFRA_ORACLE_HOME/bin/ldapsearch -h <oid_host> -p <oid_port> -D "<DN of
   privileged oid user>" -w "<password of privileged oid user>"
   -b "orclodipagentname=testExport,cn=subscriber profile,cn=changelog
   subscriber,cn=oracle internet directory" -s base "objectclass=*"
   orclodipsynchronizationstatus orclodiplastsuccessfulexecutiontime
   ```

   For example:

   ```
   INFRA_ORACLE_HOME/bin/ldapsearch -h m1.abc.com -p 389 -D "cn=orcladmin" -w
   "welcome1"
   -b "orclodipagentname=testExport,cn=subscriber profile,cn=changelog
   subscriber,cn=oracle internet directory" -s base "objectclass=*"
   orclodipsynchronizationstatus orclodiplastsuccessfulexecutiontime
   ```

   When synchronization is successfully started:

   - The value of the `orclodipsynchronizationstatus` attribute is `Synchronization Successful`.
   - The value of the `orclodiplastsuccessfulexecutiontime` attribute is the specific date and time of that execution. Note that this must be close to the current date and time.

   The following is an example of a result indicating successful synchronization:

   ```
   orclodipsynchronizationstatus=Synchronization Successful
   orclodiplastsuccessfulexecutiontime=20060302170214
   ```

2. After verifying that synchronization has started, check if the entries in Oracle Internet Directory are actually synchronized to Microsoft Active Directory by performing the following steps:

   a. Click **Start**, **Programs**, **Microsoft Exchange**, and then **Active Directory Users and Computers**.

   b. Look for entries under the `Users` container under the Active Directory domain.

10.3.2 Procedure 2: Configuring BPEL-Based Organization Alerts

Although it is convenient to always have up-to-date information available in Microsoft Outlook, there are some changes for which you may want to be notified by e-mail. For example, if a change is made in the Human Resources Management System to John Steinbeck's telephone number, John might want to be notified about this.

If you are using Oracle Internet Directory and Microsoft Active Directory in an enterprise environment such as the one described in Figure 10–1, you can optimize the

power of Oracle Directory Integration and Provisioning, combined with Oracle BPEL Process Manager, to create a robust alerting system that generates organization alerts based on changes that happen to user identity information in the directory.

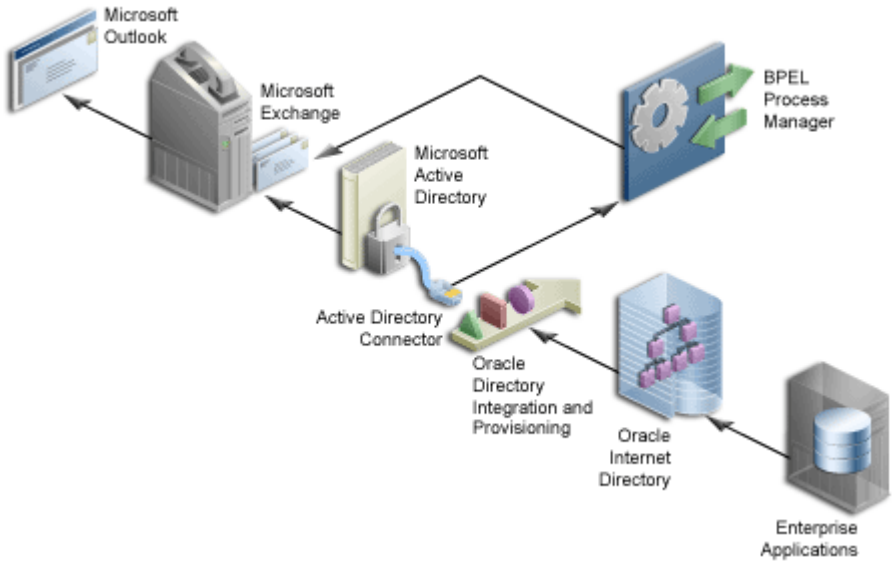
User identity information in Oracle Internet Directory can change for the following reasons:

1. Changes occur in Human Resources Management System and are synchronized to Oracle Internet Directory.
2. Modifications are directly applied to Oracle Internet Directory from other directory administration components in the deployment such as Oracle Delegated Administration Services or Oracle COREid administration service.
3. Modifications are made in Microsoft Active Directory or Microsoft Exchange in deployments where Microsoft Active Directory is the enterprise directory.

Extending our first example, Enterprise applications, for example, the Human Resources Management System, update user identity information in Oracle Internet Directory. Oracle Directory Integration and Provisioning synchronizes user identity information between Oracle Internet Directory and Microsoft Active Directory by using Active Directory Connector. A BPEL process can be deployed on the enterprise's Oracle Application Server middle tier to send organization alerts when specific user identity information attributes change in a certain domain.

Figure 10–2 shows how a BPEL process is integrated into the directory synchronization process.

Figure 10–2 Generating Organization Alerts Using Oracle BPEL Process Manager



This example is an extension of Section 10.3.1, "Procedure 1: Synchronizing Enterprise Identity Information". When John's telephone number changes, it is updated in Human Resources Management System and synchronized with Oracle Internet Directory. John would like to receive an e-mail notifying him of the recent change.

Oracle Directory Integration and Provisioning and Oracle BPEL Process Manager enable enterprise applications to create these kinds of organization alerts based on changes in Oracle Internet Directory.

The high-level steps in this scenario are the following:

1. A change is made in Union Loan's Human Resources Management System. For example, John Steinbeck's telephone number is changed.
2. This change in John's profile is recorded in Oracle Internet Directory. The `telephonenumber` attribute in Oracle Internet Directory is updated with John's new telephone number.
3. Oracle Directory Integration and Provisioning detects the change and writes this change to a database table.
4. A BPEL process, which is configured to check for changes in the same database table, picks up this change and sends an organization alert to John.

In this example, Oracle Directory Integration and Provisioning must be configured to check Oracle Internet Directory for user identity information changes. It must provision these changes into the `EXCHGSYNC` schema in OracleAS Metadata Repository. Specifically, the changes should be written to the `ORG_ALERTS` table.

> **Note:** Alternatively, the `EXCHGSYNC` schema can be present in any Oracle Database.

A BPEL process must be configured to check for changes in the `ORG_ALERTS` table, and send an organization alert to the interested parties depending on the attributes that are changed.

The following sections show how to create and send organization alerts that trigger when user information changes:

- Configuring the BPEL Process
- Configuring Oracle Directory Integration and Provisioning Profile
- Testing the Identity Alerting Configuration

10.3.2.1 Configuring the BPEL Process

A sample BPEL process, `IdentityNotification`, SQL scripts to create `EXCHGSYNC` schema with the `ORG_ALERTS` table, and corresponding packages for attribute change propagation to the table are available in the demonstration folder described in Section 10.2, "Prerequisites". The BPEL process checks for user identity information changes in the table, `EXCHGSYNC.ORG_ALERTS`, and sends organization alerts to the user if specific attributes like telephone number have changed.

To set up the schema, and to configure and test the BPEL process, `IdentityNotification`, you must perform the steps in the following sections:

- Installing the EXCHGSYNC Schema
- Configuring the Database Adapter in the BPEL Process
- Configuring E-Mail Server Settings to Enable Organization Alerts
- Compiling and Deploying the IdentityNotification BPEL Process

Installing the EXCHGSYNC Schema

To install the `EXCHGSYNC` schema, you must perform the following steps:

1. Locate the `notificationsetup.sql` script. This is available in the `identitymanagement/sql` folder in the examples ZIP file, described in Section 10.2, "Prerequisites".
2. Run the `notificationsetup.sql` script as follows:

```
sqlplus "sys/<sys pwd>@<DB connect string> as SYSDBA" @notificationsetup.sql
```

3. Set the password for the database user, EXCHGSYNC, by performing the following steps:
   a. Connect to the database as the SYS user.
   b. Run the alter command as follows:

```
alter user EXCHGSYNC identified by <password>
```

Configuring the Database Adapter in the BPEL Process

After installing Oracle BPEL Process Manager, you must edit the database adapter's oc4j-ra.xml file to enter connection details for the databases to which you will be connecting. You must create a new connection called eis/DB/IdNotifySample, which lets you run the samples against the EXCHGSYNC schema in OracleAS Metadata Repository or your custom database.

To connect to the EXCHGSYNC schema in the OracleAS Metadata Repository, perform the following steps:

1. Open the oc4j-ra.xml file, which is available in the *BPEL_ORACLE_HOME*/j2ee/OC4J_BPEL/application-deployments/default/DbAdapter directory.

   Here, *BPEL_ORACLE_HOME* is Oracle home on the Oracle Application Server middle tier that contains the BPEL Process Manager.

2. Add a new connector factory element called eis/DB/IdNotifySample, as shown in Example 10–1.

Example 10–1 New Connection String Details in the oc4j-ra.xml File

```
<connector-factory location="eis/DB/IdNotifySample" connector-name="Database
Adapter">
      <config-property name="driverClassName"
value="oracle.jdbc.driver.OracleDriver"/>
      <config-property name="connectionString" value="
jdbc:oracle:thin:@abc.unionloan.com:1521:iasdb "/>
      <config-property name="userName" value=" EXCHGSYNC "/>
      <config-property name="password" value="secret"/>
      <config-property name="minConnections" value="1"/>
      <config-property name="maxConnections" value="5"/>
      <config-property name="minReadConnections" value="1"/>
      <config-property name="maxReadConnections" value="5"/>
      <config-property name="usesExternalConnectionPooling" value="false"/>
      <config-property name="dataSourceName" value=""/>
      <config-property name="usesExternalTransactionController" value="false"/>
      <config-property name="platformClassName"
value="oracle.toplink.internal.databaseaccess.OraclePlatform"/>
      <config-property name="usesNativeSequencing" value="true"/>
      <config-property name="sequencePreallocationSize" value="50"/>
   </connector-factory>
```

   Replace the values for connectionString and password (in **bold**) with the connection string for the EXCHGSYNC schema in OracleAS Metadata Repository or your custom database and EXCHGSYNC password respectively.

3. After editing the file, stop and restart the Oracle BPEL Process Manager server. To do this, click **Start**, **Programs**, **Oracle-*ORACLE_BPEL_HOME***, **Oracle BPEL**

Process Manager 10.1.2, **Stop BPEL PM Server**, and then click **Start BPEL PM Server**.

Configuring E-Mail Server Settings to Enable Organization Alerts

The default e-mail account is used to send organization alert e-mail messages. Therefore, you must configure the e-mail server settings for the account, `Default`. To do this, perform the following steps:

1. Edit the e-mail server configuration file, *ORACLE_HOME*`/integration/orabpel/system/services/config/ns_emails.xml`, and change the parameters that are indicated in **bold** in Example 10–2.

Example 10–2 Parameters in the E-Mail Configuration File

```
<EmailAccount>
      <Name>Default</Name>
      <GeneralSettings>
         <FromName>Oracle BPM</FromName>
         <FromAddress>bpm1@m1.abc.com</FromAddress>
      </GeneralSettings>
      <OutgoingServerSettings>
         <SMTPHost>m1.abc.com</SMTPHost>
         <SMTPPort>225</SMTPPort>
      </OutgoingServerSettings>
      <IncomingServerSettings>
         <Server>m1.abc.com</Server>
         <Port>2110</Port>
         <Protocol>pop3</Protocol>
         <UserName>bpm1</UserName>
         <Password ns0:encrypted="false"
xmlns:ns0="http://xmlns.oracle.com/ias/pcbpel/NotificationService">welcome</Passwo
rd>
         <UseSSL>false</UseSSL>
         <Folder>Inbox</Folder>
         <PollingFrequency>1</PollingFrequency>
         <PostReadOperation>
            <MarkAsRead/>
         </PostReadOperation>
      </IncomingServerSettings>
   </EmailAccount>
```

Compiling and Deploying the IdentityNotification BPEL Process

To compile and deploy the BPEL process to Oracle BPEL Process Manager on the Oracle Application Server middle tier, perform the following steps:

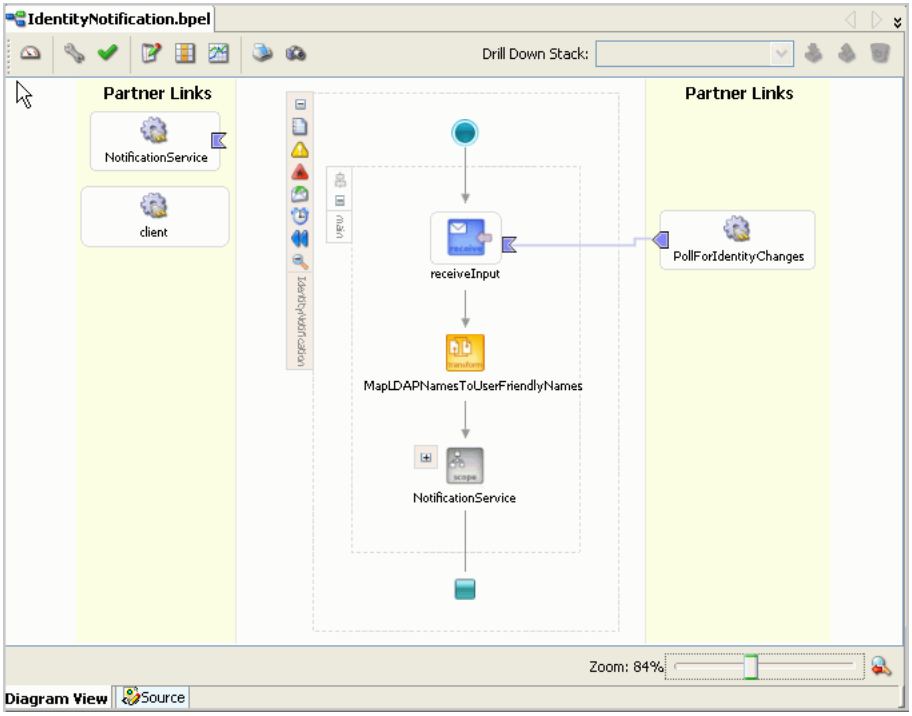
> **Note:** The database connector feature in Oracle BPEL Process Manager requires the Oracle Application Server version of Oracle BPEL Process Manager for connectivity with Oracle Internet Directory. You cannot use the standalone version of Oracle BPEL Process Manager for this.

1. Open JDeveloper BPEL Designer. Click **Start**, **All Programs**, **Oracle - *ORACLE_HOME***, **Oracle BPEL Process Manager 10.1.2**, and then **JDeveloper BPEL Designer**.

2. Open the `IdentityNotification.jpr` file in JDeveloper BPEL Designer. This contains the `IdentityNotification` BPEL process shown in Figure 10–3.

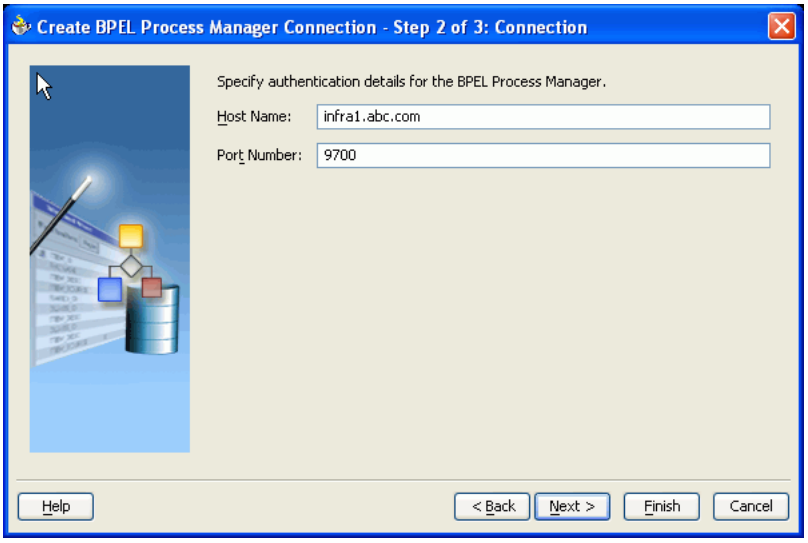
   The `IdentityNotification.jpr` file is located in the `identitymanagement/IdentityNotification` folder in the examples ZIP file, described in Section 10.2, "Prerequisites".

Figure 10–3 IdentityNotification BPEL Process



3. On the Navigator pane in JDeveloper BPEL Designer, right-click the Project node, and click **Deploy**.
4. Click **Invoke Deployment Tool**.
5. In the Deploy Properties dialog box, click **New**. The BPEL Process Manager Connection wizard is displayed.
6. Specify your connection name, and click **Next**.
7. Specify authentication details using the Oracle Application Server middle tier host name and port number, as shown in Figure 10–4.

Figure 10–4 Entering Authentication Details in the BPEL Process Manager Connection Wizard



8. Click **Next**.
9. Test your connection, and click **Finish**.
10. Enter your password, and click **OK**.

    You are now ready to deploy the BPEL process to the remote Oracle Application Server middle tier.

10.3.2.2 Configuring Oracle Directory Integration and Provisioning Profile

To enable this capability, you must configure Oracle Directory Integration and Provisioning by running the following command:

```
dipassistant exchgalertcfg -h <oid_host> -p <oid_non-ssl_port> -profile <profile_name>
```

For example:

```
dipassistant exchgalertcfg -h stadd63 -profile testExport
```

The script prompts you for the following information. Default answers to the questions are given in brackets. Press Enter to accept the default values.

- `Account DN:(default: cn=orclAdmin) >`

  Specify the super user, that is, `cn=orcladmin`, or any user that is a member of the Directory Integration and Provisioning Administrators group (`cn=dipadmingrp, cn=odi, cn=oracle internet directory`).

- `Account Password >`

  Enter the account password.

- `User login attribute name:(default: uid) >`

  If the login ID is different from the user ID, then specify the login ID.

- `Oracle Database URL for Alert Notification: >`

  Specify the URL for accessing the database. This must be in the format *`localhost`*`:`*`port`*`:`*`iasdb`*`.us.oracle.com`.

- `Oracle Database User for Alert Notification:(default: EXCHGSYNC) >`

  Specify the database user who must receive the organization alert. Accept the default value, `EXCHGSYNC`, because that is the user you created in the section Installing the EXCHGSYNC Schema.

- `Oracle Database Password for Alert Notification: >`

  Specify the `EXCHGSYNC` user password. This is the password you set in the section Installing the EXCHGSYNC Schema.

- `Microsoft Exchange Attributes to be propagated for Alert Notification:(default: mail, telephonenumber) >`

  Specify a comma-delimited list of attributes, for which organization alerts must be generated. For this example, accept the default values of mail and telephone number.

Oracle Directory Integration and Provisioning will now check Oracle Internet Directory for changes to the user's mail information and telephone number. To enable the BPEL process to use this information, you must configure it as described in the next section.

10.3.2.3 Testing the Identity Alerting Configuration

After you configure, compile, and deploy the `IdentityNotification` BPEL process, you can now test if all these steps were correct, and if the sample BPEL process is working. To do this, perform the following steps:

1. Log in to Oracle Directory Manager as the Oracle Internet Directory administrator by running the following command:

   On UNIX:

   ORACLE_HOME`/bin/oidadmin`

   On Windows:

   From the Start menu, choose **Programs**, then `ORACLE_HOME`, then **Integrated Management**, then **Oracle Directory Manager**

2. Update the `telephonenumber` attribute with a new telephone number in the container that is set up to be synchronized. To do this, perform the following steps:

   a. Click **Entry Management** and locate the entry you want to modify. You can also perform a search for the entry.

      See Also: *Oracle Internet Directory Administrator's Guide*

   b. At the top of the tab page, select **View Properties: All**.

   c. Find the `telephonenumber` property and modify it.

   d. Click **Apply**.

3. Check if the appropriate users receive an organization alert e-mail message about this change.

4. Access the BPEL Console. The URL has the following format:

   `http://<`*BPEL_Host*`>:<`*BPEL_Port*`>/BPELConsole`

   Look for the BPEL process that was invoked when the value for the `telephonenumber` attribute was changed.

10.4 Troubleshooting

Refer to the appendix titled "Troubleshooting Oracle Directory Integration and Provisioning" in *Oracle Identity Management Integration Guide*.

10.5 Related Documentation

The following is a list of references to documents that provide more information about synchronizing and provisioning Oracle Identity Management with Microsoft Active Directory:

- Chapter titled "Oracle Directory Synchronization Service" in *Oracle Identity Management Integration Guide* for more information about synchronization between Oracle Internet Directory and Microsoft Active Directory.
- Section titled "Model of Integrating Oracle Identity Management in a Windows Environment" in *Oracle Identity Management Administrator's Guide*.
- Section titled "Configuring Mapping Rules" in *Oracle Identity Management Integration Guide*.
- Some Microsoft Exchange Scenarios explained in the chapter titled "Deployment Options for Integrating with Microsoft Active Directory" in *Oracle Identity Management Integration Guide*.
- *Oracle Application Server BPEL Process Manager Installation Guide* on the Oracle BPEL Process Manager page on OTN at

  `http://www.oracle.com/technology/bpel`

11

Accessing in-Context Web Information and Invoking an Enterprise Portal

This chapter shows how to invoke an enterprise application directly from a Microsoft Office document.

This chapter contains the following sections:

11.1 Overview

Microsoft Office 2003 Professional provides facilities for invoking enterprise Web applications. These facilities enable enterprises to combine the familiar user interface of Microsoft Office with the ability to invoke enterprise applications in context when the user needs access to more detailed information.

Users can use Microsoft Office as a familiar user interface to access enterprise information. Sometimes a casual user may discover that he or she needs access to more detailed information, typically accessed through a Web application hosted on an enterprise portal. Microsoft Office 2003 Professional offers a number of options such as the following for using an Internet browser in order to invoke a Web application from the context of a Microsoft Office document:

- Static hyperlinks; for more information, refer to Section 11.3.1, "Embedding a Static Hyperlink to Invoke an Enterprise Portal".
- Microsoft Visual Basic for Application (VBA) code; for more information, refer to Section 11.3.2, "Using VBA Code to Invoke an Enterprise Portal".
- Smart tags; for more information, refer to Section 11.3.3, "Using Smart Tags to Invoke an Enterprise Portal".

11.2 Prerequisites

To perform the steps outlined in this chapter, you must install the following software:

- Oracle JDeveloper 10*g* Release 3 (10.1.3)
- Microsoft Office 2003 Professional, specifically Microsoft Word 2003

- Microsoft Internet Explorer 6.0, Mozilla Firefox 1.0, or equivalent browser
- The smart document template created in Chapter 5, "Completing Forms and Entering Data Using Microsoft Office"
- The support files in the `smarttags` demonstration folder. Refer to Accessing the Demonstration Support Files in the Preface for details about the demonstration support files. The support files in the `smarttags` demonstration folder are listed and described in Table 11–1.

Table 11–1 Smarttags Files

File or Folder	Description
unionloan_home.htm	A dummy home page for the Union Loan Portal. Download this file to your local file system. We recommend that you download it to: `C:\OfficeInt\samples\smarttags` If you download this to another location, you must edit the code in Section 11.3.3, "Using Smart Tags to Invoke an Enterprise Portal" to point to the appropriate location.
unionloan_menu.gif	One of the graphics that makes up the Union Loan Portal home page. Download this file to the same location as `unionloan_home.htm`.
unionloan_dashboard.gif	One of the graphics that makes up the Union Loan Portal home page. Download this file to the same location as `unionloan_home.htm`.
Employee Information.dot	The template document created in Chapter 5, "Completing Forms and Entering Data Using Microsoft Office". Download this file to your local file system. We recommend that you download it to: `C:\OfficeInt\samples\smarttags`

11.3 Step-by-Step Procedures

This chapter provides the following examples of how to enable a Microsoft Office smart document to invoke an enterprise portal:

- Embedding a Static Hyperlink to Invoke an Enterprise Portal
- Using VBA Code to Invoke an Enterprise Portal
- Using Smart Tags to Invoke an Enterprise Portal

These examples are based on the custom Human Resources enterprise application used by Union Loan Company. This application is used by HR and administrative personnel. You retrieve a subset of the enterprise data through a Microsoft Word 2003 Professional application (such as the smart document described in Chapter 5, "Completing Forms and Entering Data Using Microsoft Office"). Sometimes you need more data to perform their business task. For this reason, the ability to invoke the HR enterprise application hosted in the Union Loan Company portal must be added to the Microsoft Word smart document.

11.3.1 Embedding a Static Hyperlink to Invoke an Enterprise Portal

The most simple and direct way of adding browser invocation to a Microsoft Office template is by providing a static hyperlink (URL). When you move a mouse pointer over the hyperlink in the document, Microsoft Office displays a pop-up window telling you to press the CTRL key and click the link to open a browser and go to the specified address. The browser that is opened will be the default system browser, as defined by the operating system. One limitation of this approach of invoking a portal by means of static hyperlinks is that the URL can contain only static parameters as part of the string, that is, the destination of the hyperlink is fixed. However, for many enterprisewide portals this restriction will not be an issue.

To create a hyperlink in a Microsoft Office smart document, perform the following steps:

1. Start Microsoft Word.
2. Open the template you created in Chapter 5, "Completing Forms and Entering Data Using Microsoft Office" (`Employee Information.dot`).

   Tip: If you have not created this template, use the template you downloaded in Section 11.2, "Prerequisites".

3. If necessary, from the Tools menu, select **Unprotect Document** and enter the appropriate password when prompted.

   Note: If a document is protected, you must remove the protection before attempting to edit the document, otherwise edits may have unexpected results.

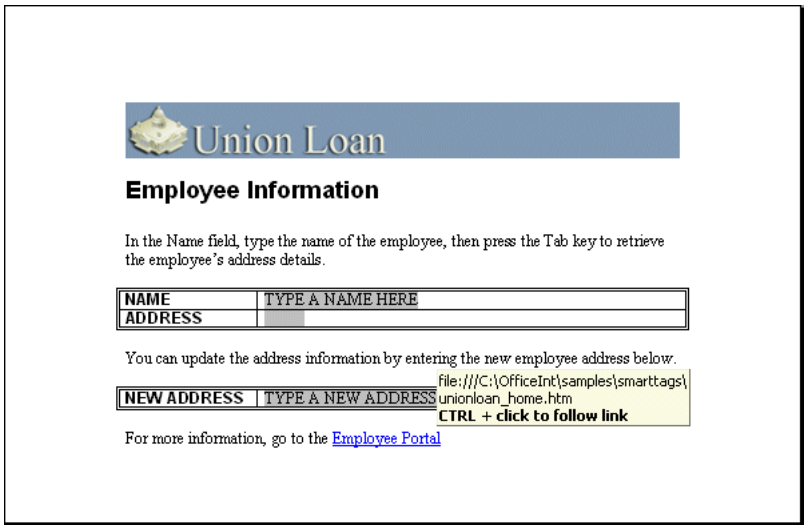
4. Place the cursor at the bottom of the template.
5. Enter the following text to introduce the hyperlink:

   ```
   For more details, go to the
   ```

6. From the Insert menu, select **Hyperlink**.
7. In the Text to display field, enter `Employee Portal.` This text is displayed in underlined blue font in the template, and is the text that users will click to invoke the portal.
8. In the Address field, enter the URL of the portal that you want to invoke. For our example, you must select the page you installed in Section 11.2, "Prerequisites", but this could be any URL.
9. Click **OK.**
10. Save and close the template.
11. In Windows Explorer, double-click the template to create a document based on the template.
12. Move your mouse pointer over the hyperlink.

    A pop-up appears indicating that you can press the CTRL key and click the URL, as shown in Figure 11–1.

Figure 11–1 Smart Document with Hyperlink that Invokes the Employee Portal



11.3.2 Using VBA Code to Invoke an Enterprise Portal

Another way of invoking an enterprise portal is to write VBA code. This code can be initiated by a button in the template document or in its task pane. One advantage of this approach is that it lets you programmatically structure the URL, enabling you to add parameters to the URL, which can be derived from data in the document, for example, from form fields.

To create a button in a Microsoft Office smart document that invokes a portal, perform the following steps:

1. Start Microsoft Word.
2. Open the template you created in Chapter 5, "Completing Forms and Entering Data Using Microsoft Office" (`Employee Information.dot`)

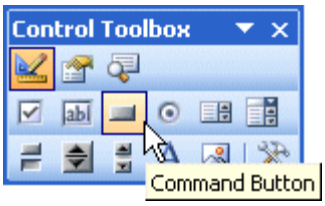
   Tip: If you have not created this template, use the template you downloaded in Section 11.2, "Prerequisites".

3. If necessary, from the Tools menu, select **Unprotect Document** and enter the appropriate password when prompted.

   Note: If a document is protected, you must remove the protection before attempting to edit the document, otherwise edits may have unexpected results.

4. Place the cursor at the bottom of the template.
5. From the View menu, select **Toolbars**, then select **Control Toolbox**.
6. In the Control Toolbox, click the **Command Button** icon (see Figure 11–2).

Figure 11–2 The Command Button Icon in the Control Toolbox



7. Right-click the new button and select **Properties** from the shortcut menu.

   Tip: If the shortcut menu does not appear, make sure that you are in Design Mode, by clicking the **Design Mode** icon in the Control Toolbox.

8. In Caption field, enter `Employee Portal.`
9. Close the Properties window.
10. Resize the button to show the caption.
11. Double-click the button. This opens the Microsoft Visual Basic Editor.
12. Replace the generated subroutine `CommandButton1_Click()` with the following code:

Example 11–1 VBA Code to Invoke Employee Portal

```
Private Declare Function ShellExecute Lib _
              "shell32.dll" Alias "ShellExecuteA" _
              (ByVal hwnd As Long, _
               ByVal lpOperation As String, _
               ByVal lpFile As String, _
               ByVal lpParameters As String, _
               ByVal lpDirectory As String, _
               ByVal nShowCmd As Long) As Long

Private Const SW_SHOW = 1

Public Sub Navigate(ByVal NavTo As String)

  Dim hBrowse As Long
  hBrowse = ShellExecute(0&, "open", NavTo, "", "", SW_SHOW)

End Sub

Private Sub CommandButton1_Click()

Navigate "<URL>"

End Sub
```

13. Replace <URL> in the body of the code with the URL of the portal you want to invoke. For our example, use the URL of the page you downloaded in Section 11.2, "Prerequisites", but this could be any URL:

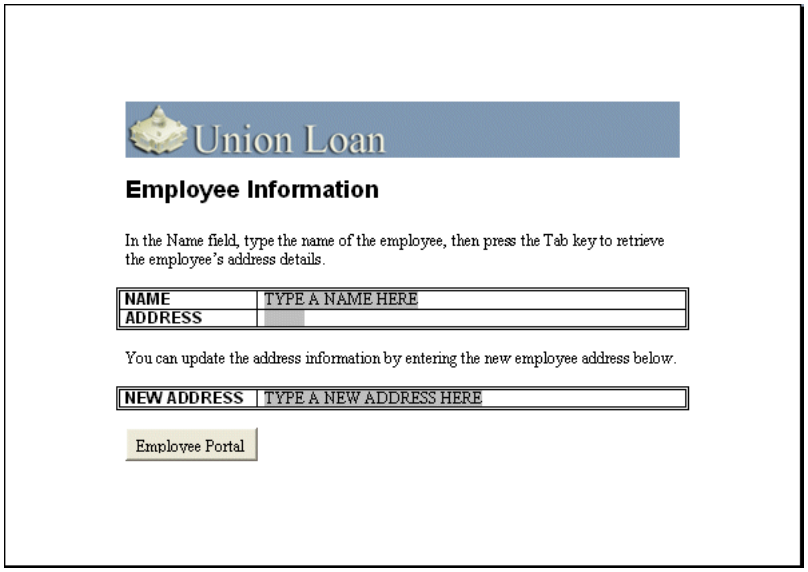
    `C:\OfficeInt\samples\smarttags\unionloan_home.htm`

14. Save the project.
15. Exit the Microsoft Visual Basic Editor and return to Microsoft Word.

16. Save and close the template.
17. In Windows Explorer, double-click the template to create a document based on the template.

    The document should look something like Figure 11–3.

Figure 11–3 Smart Document with Button that Invokes the Employee Portal



18. Click **Employee Portal**.

    A new browser window opens at the specified URL address.

If you wanted to programmatically construct a URL in Visual Basic, you could write something like the code shown in Example 11–2:

Example 11–2 Programmatically Constructing a URL

```
Private Sub CommandButton1_Click()
ActiveDocument.Fields(1).Result.Text = "PersonalLoans.htm"
Navigate "file:///C:/OfficeInt/samples/smarttags/" &
 ActiveDocument.Fields(1).Result.Text
End Sub
```

This code constructs the URL from a static string to which the value of the first form field is appended. For the purposes of this example, the value of the field is set to `PersonalLoans.htm`, but it could be a user-entered value, or one retrieved from a Web service (see Chapter 5, "Completing Forms and Entering Data Using Microsoft Office".)

11.3.3 Using Smart Tags to Invoke an Enterprise Portal

Another way of invoking an enterprise portal is to use the *smart tag* mechanism in Microsoft Office 2003 Professional. Smart tags are *recognizers* that scan the document as the user is typing. Whenever a specified phrase (that is, a literal text string, or a regular expression) for a smart tag is recognized, an associated action class is invoked. In principle, a newly developed smart tag is available to all document templates, but it must be enabled explicitly for each template to which it should apply.

Developing new smart tags typically involves Visual Studio to create the action class that must be packaged with an XML deployment descriptor that defines the tag. However, Microsoft Office Professional also comes with a set of standard tags and associated default actions, the Microsoft Office Smart Tag List (MOSTL). Using the MOSTL tags and actions simplifies the development process to authoring an XML document that conforms to the MOSTL schema and placing it in the Microsoft Office installation directory. This declarative approach does have some limitations, notably in terms of parameters that can be accessed dynamically. Standard MOSTL tags for URL invocation can access only the recognized strings, not other values in the document (such as field values).

For information on where to find additional documentation about smart tags, refer to Section 11.5, "Related Documentation".

In this section, you will create a simple MOSTL tag that uses the predefined action to invoke a Web browser.

To create a MOSTL tag:

1. Start a standard XML editor, for example, Oracle JDeveloper or Macromedia Dreamweaver.
2. Create the following sample XML document, which adheres to the standard MOSTL schema:

Example 11–3 XML Code for MOSTL Smart Tag

```
<FL:smarttaglist xmlns:FL="urn:schemas-microsoft-com:smarttags:list">
  <FL:name>Employee Portal Invocation</FL:name>
  <FL:lcid>1033,0</FL:lcid>
  <FL:description>Invoke the employee portal</FL:description>
  <FL:moreinfourl>http://msdn.microsoft.com/msdnmag</FL:moreinfourl>
  <FL:updateable>false</FL:updateable>
  <FL:autoupdate>false</FL:autoupdate>
  <FL:lastcheckpoint>0</FL:lastcheckpoint>
  <FL:lastupdate>0</FL:lastupdate>
  <FL:updateurl></FL:updateurl>
  <FL:updatefrequency>0</FL:updatefrequency>
  <FL:smarttag type="urn:schemas-microsoft-com:office:smarttags#launch">
    <FL:caption>Launch Employee Portal</FL:caption>
      <FL:re>
        <FL:exp>Employee Portal</FL:exp>
      </FL:re>
    <FL:actions>
      <FL:action id="launch">
        <FL:caption>Launch Employee Portal</FL:caption>
        <FL:url>
          file:///C:/OfficeInt/samples/smarttags/unionloan_home.htm
        </FL:url>
      </FL:action>
    </FL:actions>
  </FL:smarttag>
```

The first section of this XML file defines a set of general properties for the tag, for example, its location and if it is centrally updatable. The section `<FL:exp>Employee Portal</FL:exp>` defines the recognizer string, and the section `<FL:url>file:///C:/OfficeInt/samples/smarttags/unionloan_home.htm</FL:url>` defines that the standard URL action is to be invoked.

3. Save the file under the name `loan app smart tag.xml` in the smart tag installation directory, for example:

   ```
   C:\Program Files\Common Files\Microsoft Shared\Smart Tag\LISTS
   ```

   or in a local subdirectory of that folder (for example 1033).

4. Start Microsoft Word.

5. Open the template you created in Chapter 5, "Completing Forms and Entering Data Using Microsoft Office" (`Employee Information.dot`).

   Tip: If you have not created this template, use the template you downloaded in Section 11.2, "Prerequisites".

6. If necessary, from the Tools menu, select **Unprotect Document** and enter the appropriate password when prompted.

   Note: If a document is protected, you must remove the protection before attempting to edit the document, otherwise edits may have unexpected results.

7. Place the cursor at the bottom of the template.

8. Enter the following text:

   ```
   For more information, go to the Employee Portal.
   ```

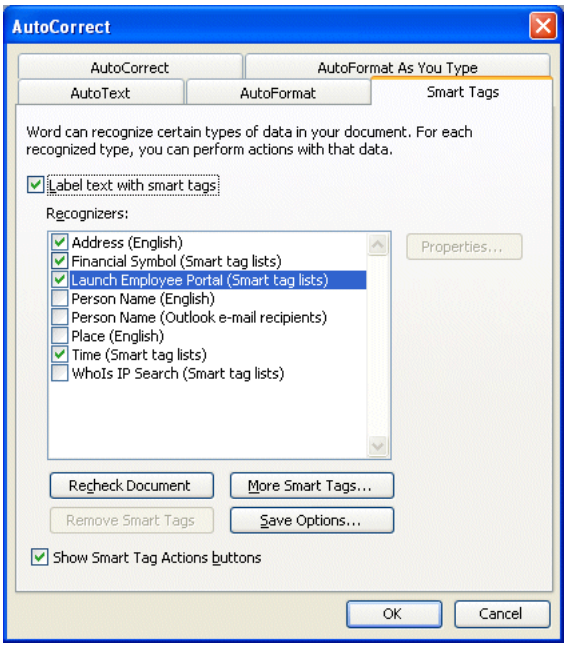
9. To enable the smart tag for the template:

   a. From the Tools menu, select **AutoCorrect Options**. The AutoCorrect dialog box is displayed as shown in Figure 11–4.

   b. Click the **Smart Tags** tab.

   c. Select the **Launch Employee Portal (Smart tag lists)** option.

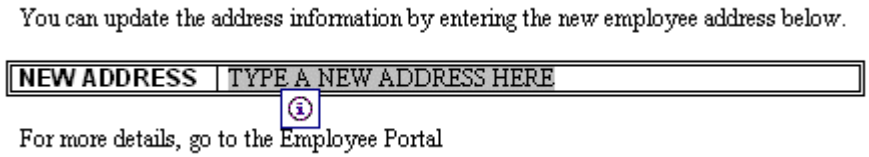
Figure 11–4 Enabling Smart Tags



10. Click **OK.**
11. Save and close the template.
12. In Windows Explorer, double-click the template to create a document based on the template.
13. Move your mouse pointer over the text Employee Portal.

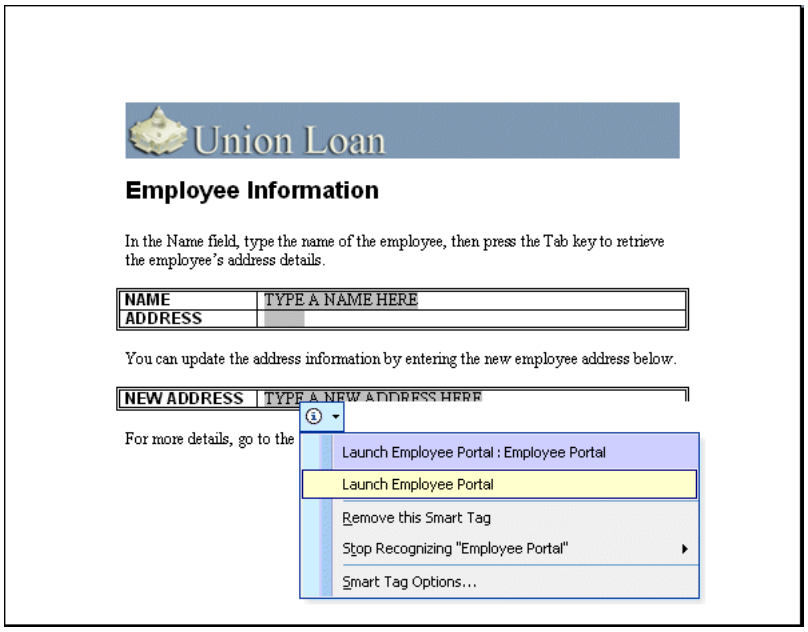
    An icon is displayed, as shown in Figure 11–5.

Figure 11–5 Smart Tag Icon



14. Click the icon to display a menu from which the browser can be invoked, as shown in Figure 11–6.

Figure 11–6 Smart Document with Smart Tag that Invokes the Employee Portal



11.4 Troubleshooting

The following information is a set of hints and tips that address potential issues:

Limitations on Using Smart Tags in Form Completing Documents

It does not appear to be possible to use smart tags in smart document templates that are restricted in editing actions to form filling mode only. Remove such restrictions in order to enable running smart tags with form fields (from the Tools menu, select **Unprotect Document**, and supply the appropriate password). Note that this does impact the usability of form completing for the end user (for example, be careful that tabbing does not add empty lines to a grid).

Editing an existing template in Microsoft Word

In order to *run* a form template, it must first be protected with a password (from the Tools menu, select **Protect Document**, then provide a password). When the document is run, remember to remove the document protection before attempting to edit the definition (from the Tools menu, select **Unprotect Document**, and supply the appropriate password), otherwise edits may have unexpected results.

Smart tags need security permission to run

Smart tags are considered a form of macros in terms of security. If the security level is set to High, then smart tags may not be able to run. Run the smart tags in Medium security level instead (From the Tools menu, select **Macro**, then select **Security**).

11.5 Related Documentation

You can find further information on smart tags on the Microsoft MSDN site:

- Introduction:

http://msdn.microsoft.com/msdnmag/issues/05/02/ManagedSmartTags/default.aspx

- Smart tag end user guide:

  http://office.microsoft.com/en-us/assistance/HP030833041033.aspx

- Sample application:

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc_2003_ta/html/odc_landoffice03_ta.asp

- MOSTL standard schema:

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/stagsdk/html/stconMOSTLNamespaces_HV01083883.asp

- MOSTL sample:

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sdsdk/html/sdconMOSTLSample_HV01083371.asp

12

Saving Microsoft Office Documents to the OracleAS Portal Content Repository

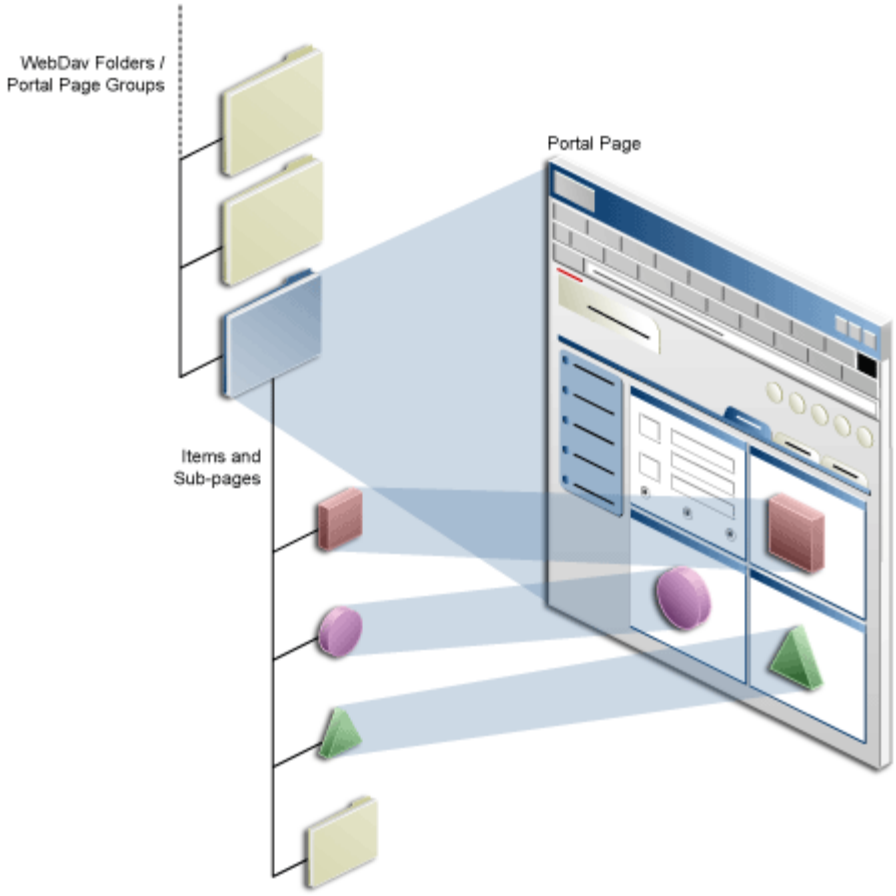
This chapter shows how you can use various tools to save Microsoft Office documents to the Oracle Application Server Metadata Repository.

This chapter contains the following sections:

12.1 Overview

In any enterprise, documents are frequently spread across many different data sources. To make the information accessible and easy to find, the ideal solution is to save the data in one central content repository, such as the one provided by OracleAS Portal. But how do you move and publish distributed content into your portal? For simple, distributed, low-volume file transfer, you can map the Portal schema in the Oracle Application Server Metadata Repository as a Web Folder (Figure 12–1).

Figure 12–1 Pages and Content in WebDAV and OracleAS Portal



OracleAS Portal supports the use of a Web-based Distributed Authoring and Versioning (WebDAV) protocol. Using a WebDAV client, such as Web Folders, you can manage your portal content re-creating your computer's file system. You can also drag and drop files between your desktop and your portal page groups, move content between page groups, and move content between source repositories and the Portal schema in the OracleAS Metadata Repository. For example, you can mount both the Oracle Content Services and the Portal schema in the OracleAS Metadata Repository through a Web Folder, and exchange files. You can perform in-place opening, editing, and saving of portal content using WebDAV-compliant desktop applications, such as Microsoft Office 2003.

Using WebDAV clients with OracleAS Portal takes full advantage of the portal's content management capabilities. It enables users to manage content through their file systems, while adhering to the content structure and access rules specified for the portal.

For example, you can access the Portal schema in the OracleAS Metadata Repository through a WebDAV client, such as Oracle Drive. Connect as a particular user, and create a ZIP archive starting from the page group's root page. This operation adheres to the access rules established for the portal—only the logged-in user's content is copied to the archive. And it respects the portal structure by maintaining the WebDAV folder hierarchy of the targeted page group.

12.2 Prerequisites

To perform the steps outlined in this chapter, first install the following software:

- Oracle Application Server Portal 9.0.4.2 or later

  Oracle Application Server Portal is part of Oracle Application Server.

- Oracle Drive. You can download the latest version of Oracle Drive from the Oracle Collaboration Suite Downloads page at

  `http://www.oracle.com/technology/software/products/cs`

  Note: If you encounter a problem with Oracle Drive, report it to the Portal Content Management Forum on the Oracle Technology Network at

  `http://www.oracle.com/technology/products/ias/portal/discussion_forums.html`

- Microsoft Office (version 2000 or later)

12.3 Step-by-Step Procedures

This chapter shows you how to use several tools to save Microsoft Office documents to the OracleAS Portal content repository:

- Setting Up OracleAS Portal for WebDAV
- Setting Up Your WebDAV Client
- Using Oracle Drive as a WebDAV Client
- Using Web Folders as a WebDAV Client
- Using Microsoft Office as a WebDAV Client

12.3.1 Setting Up OracleAS Portal for WebDAV

WebDAV is configured on both the server side (in OracleAS Portal) and the client side (your personal computer).

OracleAS Portal has a DAV configuration file (`oradav.conf`) that contains OraDAV parameters. When Oracle Application Server is installed, all required OraDAV parameters are set with values that enable access to Oracle Database content through a Web browser or a WebDAV client. If necessary, the portal administrator can modify parameter values if the default values do not meet your portal's needs.

If you are a portal administrator and would like detailed information about the `oradav.conf` file and how to modify OraDAV parameters, refer to *Oracle Application Server Portal Configuration Guide*.

Additionally, there are options within the OracleAS Portal user interface that you can set in advance to prepare the way for uploading content through a WebDAV client. For information about these options, refer to *Oracle Application Server Portal User's Guide*.

12.3.2 Setting Up Your WebDAV Client

The steps required to set up a WebDAV client to connect to your portal vary depending on the client. But all clients will eventually request a URL. The WebDAV

URL is very similar to the URL you use to access the portal in your Web browser. It uses the following format:

```
http://<hostname>:<port>/<dav_location>
```

Where `dav_location` is the location as specified in the `oradav.conf` file.

The default portal DAV URL is:

```
http://<hostname>:<port>/dav_portal/portal
```

- The `dav_portal` part of the URL is the default name of a virtual directory that is used to differentiate between portal access through a WebDAV client and portal access through the portal user interface.
- The `portal` part of the URL is the name of the database access descriptor (DAD) of the portal installation. Administrators can also configure virtual hosts to provide a different, simpler, or easier to remember URL for WebDAV access, if need be.

Directly access a particular page group or page by adding its name to the WebDAV URL, for example:

```
http://mymachine.mycompany.com:5000/dav_portal/portal/mypagegroup/mypage
```

You connect to a portal through WebDAV clients using the same user name and password that you use to log in to the portal itself. If the portal is in a hosted environment, then you also must add your company information to your user name, as follows:

```
<username>@<company>
```

If you are using Web Folders on Windows 2000, then you may be prompted for your user name and password twice: once when you click **Next** after specifying the WebDAV URL, and again when you click **Finish**.

Some WebDAV clients (such as, Windows 2000 or NT) do not support multiple simultaneous logins. If you want to log in as a new user, then you must clear your cookies, restart your computer to clear out the current login session, and then log in as the new user.

You may need to delete the file `C:\Documents and Settings\<user>\Cookies` from the command prompt window. You can do this only when no other processes are using the file.

If your WebDAV client has no explicit logout feature, you must log out of the operating system (such as Windows 2000 or NT) to log out of the portal.

12.3.3 Using Oracle Drive as a WebDAV Client

Oracle Drive is a powerful WebDAV client that you can use to map the Portal schema in the OracleAS Metadata Repository as a drive, and perform desktop authoring and publishing, and portal-specific metadata attribution directly from the Windows desktop.

Key highlights include the following:

- Mount OracleAS Portal Repositories as Microsoft Windows Drives
- Edit and view content with *any* Windows application
- Work with offline content and synchronize when online

- Extra capabilities available in the right-click menus
- Set properties, grant access, and preview content and pages
- Access the repositories with a command line (DOS) utility
- Search from Windows Explorer

In this example, a user (James Cooper) uses Oracle Drive to map the Portal schema as a drive and then work with Microsoft Office files within that portal.

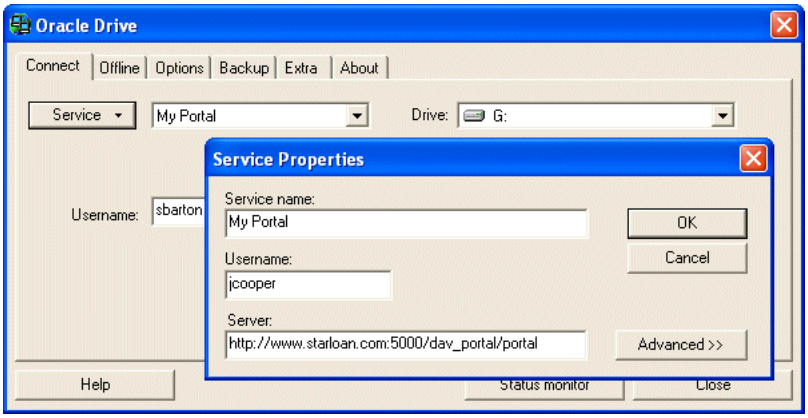
To use Oracle Drive to add content to a page, perform the following steps:

1. Click the **Oracle Drive** icon in the system tray.

   This displays the Oracle Drive dialog box, as shown in Figure 12–2.
2. If necessary, click the **Connect** tab to make it the active tab.
3. From the Service menu, select **New**.
4. In the Service name field, enter My Portal.
5. In the Username field, enter your portal user name.
6. In the Server field, enter the WebDAV URL for your portal. You can also specify a particular page within a portal. In our example, James enters the WebDAV URL of the Star Loan portal (see Figure 12–2)

   ```
   http://www.starloan.com:5000/dav_portal/portal
   ```

Figure 12–2 Oracle Drive Service Properties



Note: The **Server** field has a limit of 64 characters, therefore if you have a long WebDAV URL or you are specifying a particular page, the whole URL may not fit. In this case, you can click the **Advanced** button and specify the **Server, Port,** and **Server directory** separately, for example:

- **Server:** http://www.starloan.com
- **Port:** 5000
- **Server directory:** dav_portal/portal

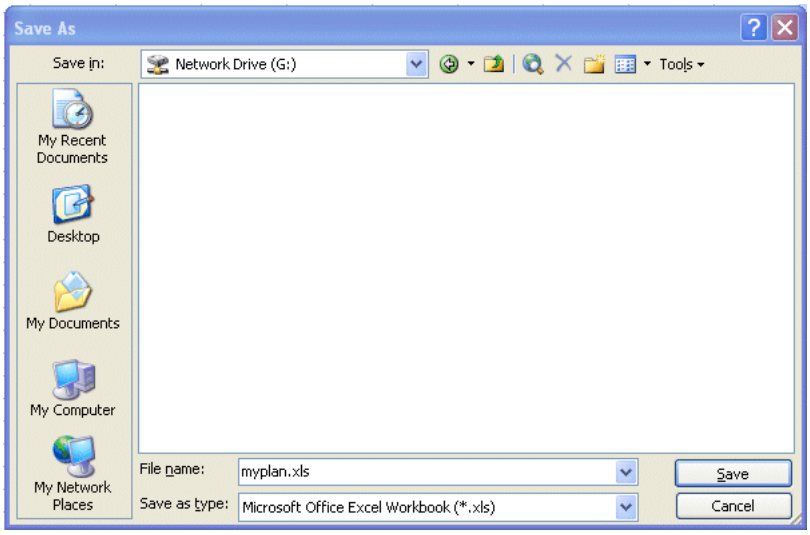
---

7. Click **OK**.

8. From the Drive list, select a drive letter to map to the new Oracle Drive connection.
9. Click **Connect**.
10. Enter your portal password to authenticate and establish the connection.
11. The Oracle Drive WebDAV connection is assigned to the drive letter selected in Step 8 and is available in Windows Explorer. You can now use this drive in the same way as any other drive.

    For example, James can create a new Microsoft Excel spreadsheet (`myplan.xls`) and save it directly to his portal, as shown in Figure 12–3.

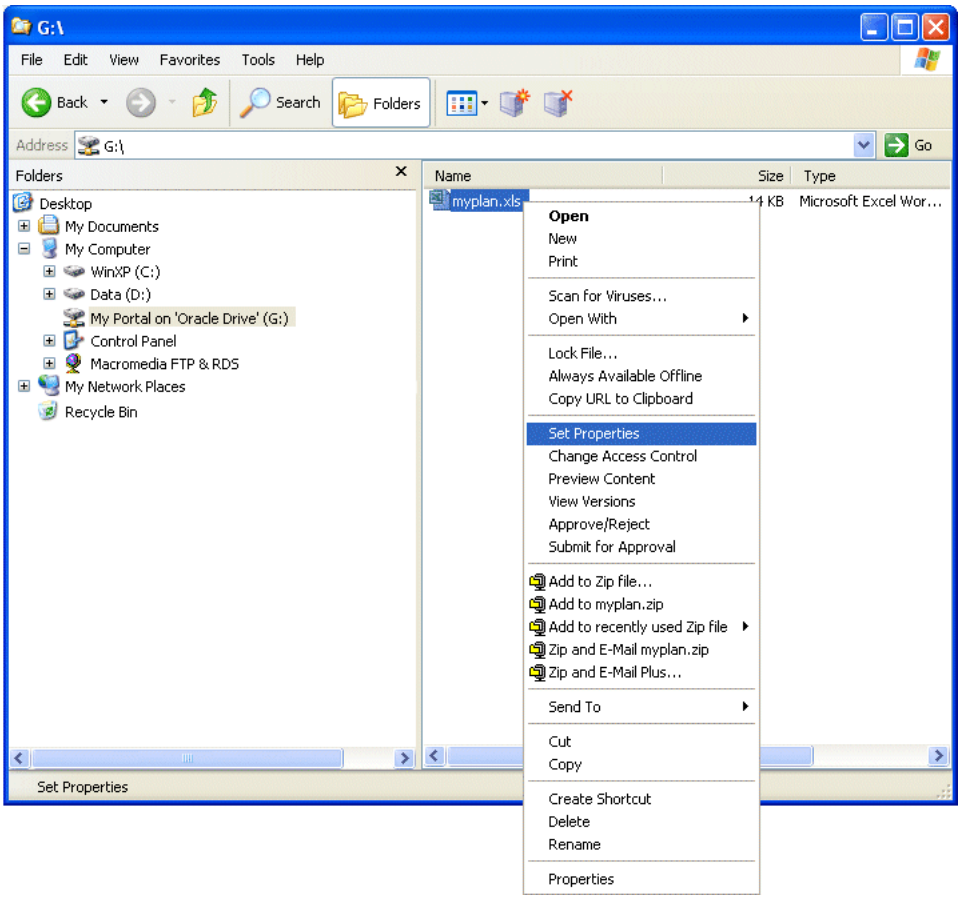
Figure 12–3 Saving Files Directly to a Network Drive



12. In addition, when you right-click any file or folder under the mapped Oracle Drive, file and folder options specific to OracleAS Portal are displayed in the resulting menu.

    For example, James wants to assign his spreadsheet to the Plan category, so he right-clicks the spreadsheet file, and then selects **Set Properties** from the shortcut menu, as shown in Figure 12–4.

Figure 12–4 Setting Properties of a File on the Network Drive



After providing his user name and password (this happens the first time he opens a file or folder through Oracle Drive; he will not need to reauthenticate for any other files or folders during this session), James sees the Edit Item window where he can assign a category for the file.

File Menu Options include the following:

- **Set Properties**—Displays the Edit Item window for the current *Active* version of the item. If approvals are enabled, and a user is accessing a *Draft*, then Set Properties displays the user's own draft version of the item, if a draft exists, and the current user created it. If the current user is the approver of the item, then Set Properties displays the Edit Item window for the *Pending* version of the item, if one exists.
- **Change Access Control**—When item-level security is enabled, Change Access Control displays the Item Access window for the current *Active* version of the item.
- **Preview Content**—Previews item content of the current *Active* version of the item. If approvals are enabled, and a user is accessing a *Draft*, then Preview Content previews the user's own draft version of the item, if a draft exists, and the current user created it. If the current user is the approver of the item, then Preview Content previews the *Pending* version of the item, if one exists.
- **View Versions**—When version creation is enabled, View Versions displays the version history of the item.

- **Approve/Reject**—This menu option is usable only when the current item is *Pending* and the current user is an approver; otherwise, when a user selects Approve/Reject, an error message displays.
- **Submit for Approval**—This menu option is usable only when the current item is a *Draft* item and the current user is the creator of the draft; otherwise, when a user selects Submit for Approval, an error message is displayed.

Folder Menu Options include the following:

- **Set Properties**—Displays the Page Properties window.
- **Change Access Control**—Displays the Page Access tab in the Page Properties window.
- **View Page**—Displays the page.

For information on which items are accessible in what states, refer to *Oracle Application Server Portal User's Guide*.

12.3.4 Using Web Folders as a WebDAV Client

Web Folders is a Microsoft operating system extension that supports the WebDAV protocol. If you access Web Folders on your computer, then you can browse the content of your portal through Windows Explorer and drag and drop Microsoft Office documents into the pages of your portal.

In this example, a user (James Cooper) uses Web Folders to add a Microsoft Word document to the Star Loan portal.

Note: This example shows you how to use Web Folders in Windows XP where the Web Folders feature is built into the operating system as part of My Network Places. You should be able to access the Add Network Place Wizard by clicking My Network Places then double-clicking Add Network Place. To use Web Folders in Windows 9*n*/NT, you must install Internet Explorer 5.5 (rather than 6.0) and the Web Folders component. You can then upgrade to 6.0, and Web Folders will remain.

If you have Internet Explorer 5.5 installed, but cannot find a Web Folders node under My Computer, you must explicitly install the Web Folders component of Internet Explorer through Add/Remove Programs in the Control Panel.

To use Web Folders to add content to a page (Windows XP), perform the following steps:

1. Open Windows Explorer and click **My Network Places**.
2. Double-click **Add Network Place** to display the Add Network Place Wizard.
3. Make sure **Choose another network location** is selected, then click **Next**.
4. Enter the WebDAV URL for your portal. You can also specify a particular page within a portal. In our example, James enters the WebDAV URL of the Star Loan portal:

   ```
   http://www.starloan.com:5000/dav_portal/portal
   ```

   If you are not sure of the WebDAV URL for your portal, contact your portal administrator.

5. Click **Next**.
6. Enter a name for the network place. James enters My Portal.
7. Click **Next**.
8. Make sure that the **Open this network place when I click Finish** check box is selected, then click **Finish**.
9. When prompted, enter your portal user name and password.

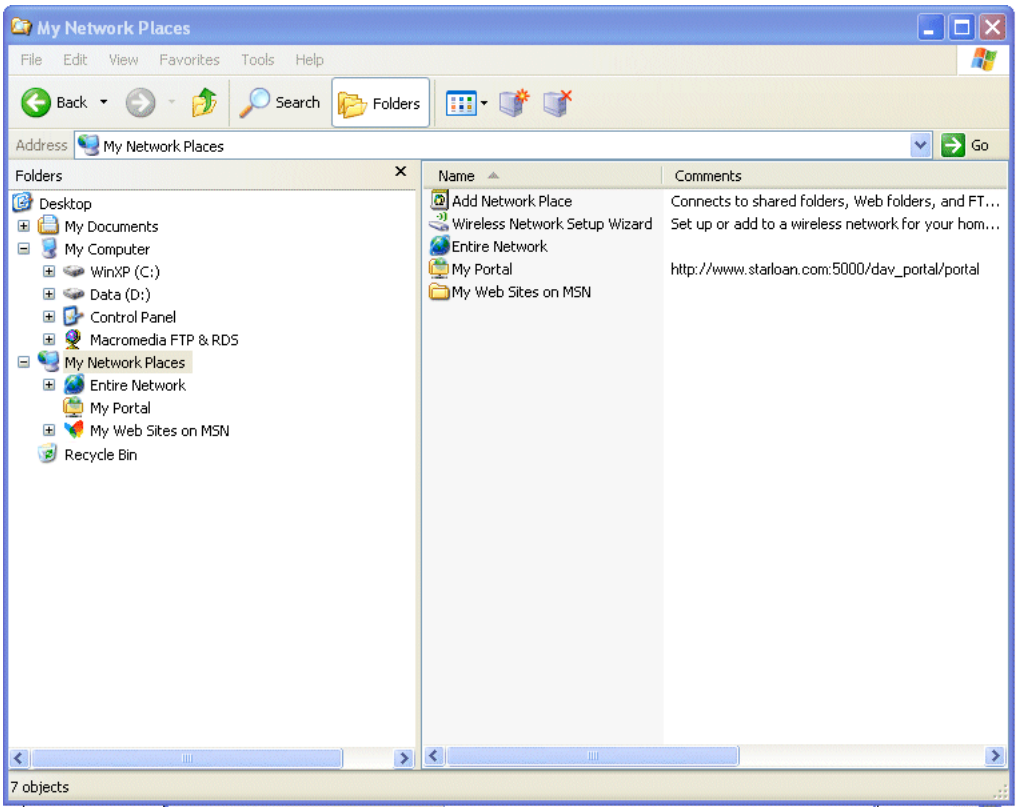
Note: You can log in to Web Folders as *one* user at any given time. That is, multiple simultaneous logins are not allowed. If you want to log in to Web Folders as a new user, then clear your cookies, restart your computer to clear out the current login session, and then log in as the new user.

Windows Web Folders sometimes stores your user name and password, or your portal session cookie, or both, and uses these details when creating subsequent Web Folders or re-creating a Web Folder that has previously existed. Consequently, it may not be possible to create multiple Web Folders for portals on the same host using different authentication information. Unless you understand exactly how your Web Folder implementation acts, on a given machine, do not create more than one Web Folder referencing any single host. Also, if you attempt to create a Web Folder that references the same portal as an existing (or previous) Web Folder but with different user log in details, then be vigilant—Windows may perform the login step without prompting you.

---

10. Windows Explorer opens a window displaying the contents of the portal or page you specified. The portal is now listed as one of your network places, as shown in Figure 12–5.

Figure 12–5 Portal Displayed As a Network Drive in Windows Explorer

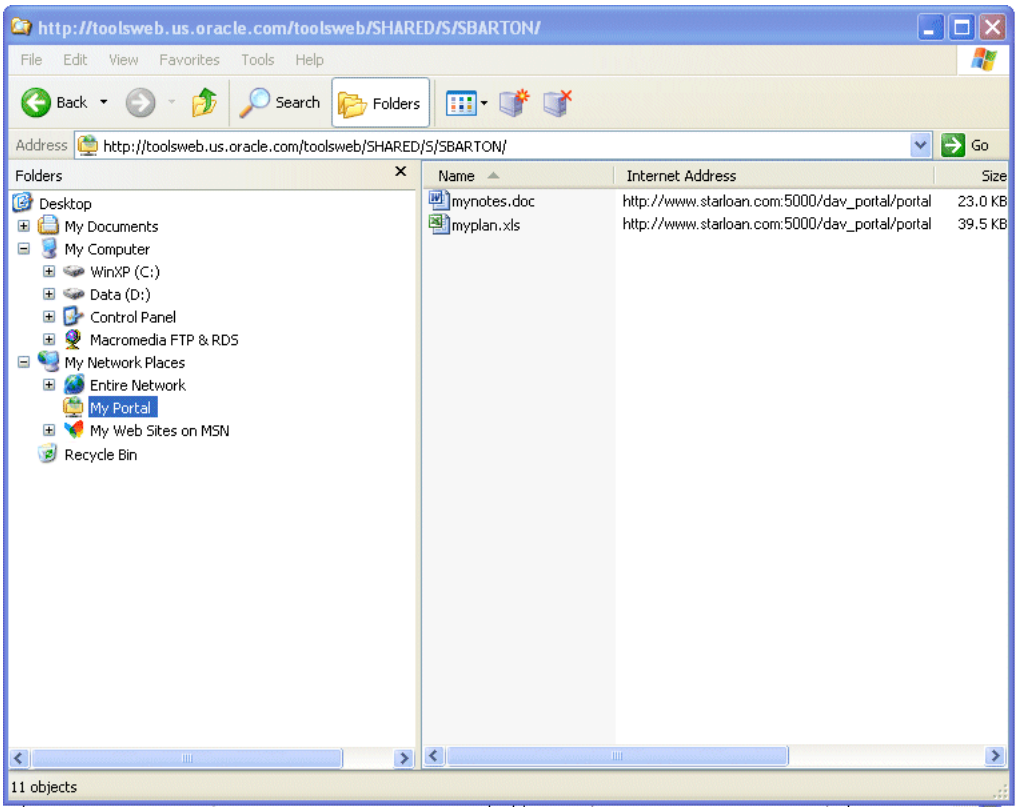


11. Because you can access your portal's Web Folder just like any other folder in Windows Explorer, you can drag and drop files from any other folder into your portal.

    You can use the drag and drop feature to move or copy files within the same page group. Copying files across page groups is not supported; though you can overcome this limitation by copying a file to your desktop, then copying it into a different page group.

    For example, James drags a Microsoft Word document called `mynotes.doc` from his local drive into `My Portal`, as shown in Figure 12–6.

Figure 12–6 Dragging and Dropping Files into the Web Folder



12. When you next access the page in the portal, it includes the new file. You may need to refresh the page to see your changes.

    The display name of the new item is the same as the file name (with the extension removed). You can edit the item later to change this.

> **Note:** The item is added to the default item region as one of the default WebDAV item types (depending on whether the file is a ZIP file, an image file, or a regular file).

12.3.5 Using Microsoft Office as a WebDAV Client

If you are using Microsoft Office 2003, a WebDAV-enabled desktop application, you can start the application, open a portal file that is exposed in a Web Folder, edit the content, and save it back to the portal. There is no need to download the document from the portal and then upload it again after you have edited it.

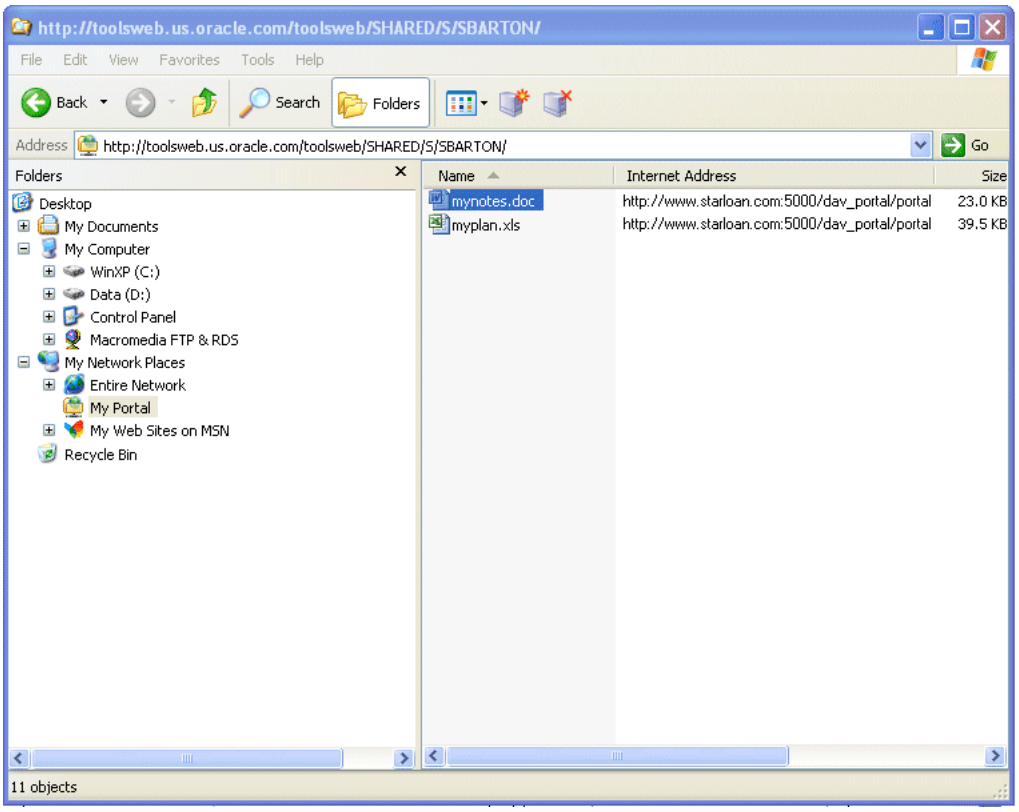
To use Microsoft Office 2003 to edit the content of a page, perform the following steps:

1. Set up Web Folders to access the page through Windows Explorer.

   See Section 12.3.4, "Using Web Folders as a WebDAV Client".

2. Double-click the file, as shown in Figure 12–7.

Figure 12–7 Working Directly with Files on the Web Folder



3. If prompted, enter your portal user name and password.

   The selected file is opened by the relevant Microsoft Office 2003 application. For example, when James double-clicks `mynotes.doc`, it opens in Microsoft Word.

   When opening a file, Microsoft Office 2003 locks the file and the related item is checked out in the portal. If the Microsoft Office application cannot lock the file (that is, it is locked by another WebDAV client, or the item was checked out in the portal), then it is opened as read-only. If the Microsoft Office application locked the document when it was opened, then it unlocks it when the document is closed.

4. Edit the document, and select **File,** then **Save**.

5. Exit Microsoft Word.

   When you next access the file item in the portal, note that it includes the changes you made.

> **Note:** You can also use Microsoft Office 2003 to create new items in the portal. Simply create the file and then save it directly to the Web Folder for your portal. The item is created as the default WebDAV item type for regular files, and is added to the default item region.

12.4 Troubleshooting

For troubleshooting information about using WebDAV clients with OracleAS Portal, refer to *Oracle Application Server Portal User's Guide.*

12.5 Related Documentation

For more detailed information about using WebDAV clients with OracleAS Portal, including general tips and information about other WebDAV clients, refer to *Oracle Application Server Portal User's Guide*.

13

Delivering Enterprise Reports to Microsoft Office with Oracle Reports

This chapter shows how to deliver reports built with Oracle Reports to Microsoft Office. It shows how you can save report output as a Microsoft Excel spreadsheet, as a Microsoft Word document, or send as an e-mail attachment.

This chapter contains the following sections:

13.1 Overview

Oracle Reports is a powerful enterprise reporting tool that enables you to rapidly develop and deploy sophisticated Web and paper reports against any data source (including an Oracle database, JDBC, XML, text files, and Oracle OLAP). Leveraging the latest J2EE technologies such as JSP and XML, you can publish your reports in a variety of formats (including HTML, XML, PDF, spreadsheet, delimited text, PostScript, and RTF) to any destination in a scalable, efficient manner. In addition to built-in destinations such as e-mail, Web browser, OracleAS Portal, file system, FTP, and WebDAV, you can define access to your own custom destination by using the Oracle Reports Java APIs.

If you want to share your reports with other people, it is helpful to do so using a familiar format that does not require people to install additional software. Oracle Reports runs a report directly to the output format you specify, with no additional setup required. By following the step-by-step instructions in this chapter, you can send a report to anyone who has a browser and Microsoft Office installed.

Note: You can use Microsoft Excel as an ODBC data source using the JDBC-ODBC driver provided with Oracle Reports. For more information on the JDBC-ODBC driver, refer to the chapter "Configuring and Using the JDBC PDS" in *Oracle Application Server Reports Services Publishing Reports to the Web*.

To enhance the out-of-the-box integration with Microsoft Office, you can use the Oracle Reports Java API to create your own custom destinations and data sources. Refer to the Oracle Reports 10*g* Release 2 (10.1.2) Java API documentation (http://download.oracle.com/docs/html/B14049_01/toc.htm)

13.2 Prerequisites

To perform the steps outlined in this chapter, you must first ensure the following prerequisites are available on your machine:

- Oracle Application Server 10*g* Release 2 (10.1.2). The steps in this chapter rely on the Business Intelligence and Forms installation. When working with other install types, refer to the Oracle Application Server documentation.
- Optionally (to use Reports Builder), Oracle Developer Suite 10*g* Release 2 (10.1.2).
- Access to an Oracle Database 10*g* database with the Sales History sample schema installed. To access an Oracle Database 10*g* database, your `tnsnames.ora` file (in *ORACLE_HOME*`\network\admin`) must include an entry for the database. If you do not know if the database has the Sales History sample schema installed, contact your database administrator. All sample schemas provided with Oracle Database 10*g* installation are described in *Oracle Database Sample Schemas*.
- Microsoft Office 2000 or later.
- A Web browser that supports displaying documents in Microsoft Excel and Microsoft Word.
- A running instance of OC4J to enable deployment of the report on the Web:
  - For Oracle Application Server installations, start Reports Server using Oracle Process Manager and Notification Server (OPMN).
  - For Oracle Developer Suite installations, start an OC4J instance on your machine before you submit a request, as follows:
    - From the Start menu, select **All Programs**, **Oracle iDS Home**, **Reports Developer**, and then **Start OC4J Instance**.
    - Or, open a command prompt, and enter:

```
cd %ORACLE_HOME%\j2ee\home
runoc4j.bat
```

The OC4J instance starts once the containers for J2EE have been initialized.

To confirm that OC4J is running and your environment is set up correctly to run a report request on the Web, enter the following URL in your browser to display the Reports Servlet (`rwservlet`) help page:

```
http://hostname:port/reports/rwservlet
```

If you see the help page, you are ready to run report requests on the Web.

- The `pluginParam` element in the server configuration file (*`ORACLE_HOME`*`\reports\conf\`*`server_name`*`.conf`) and the Reports Builder configuration file (*`ORACLE_HOME`*`\reports\conf\rwbuilder.conf`) specifies your outgoing SMTP mail server name. For example:

  ```
  <pluginParam name="mailServer">smtpserver.mycompany.com</pluginParam>
  ```

- The example report named `reports\mypaperreport.rdf`. For details on how to locate the example files, see Accessing the Demonstration Support Files in the Preface. The example file used in this chapter is listed and described in Table 13–1.

Table 13–1 Example Report File

File	Description
`reports\mypaperreport.rdf`	The sample paper report.

13.3 Step-by-Step Procedures

The steps in these sections show how to use Oracle Reports to develop reports and deploy them to Microsoft Office applications:

- Creating a Report
- Displaying Report Output in Microsoft Excel
- Displaying Report Output in Microsoft Word
- Sending Report Output to E-Mail Recipients

13.3.1 Creating a Report

To create a report with Oracle Reports, you must install Oracle Developer Suite, which includes Reports Builder.

Reports Builder provides user-friendly wizards that guide you through the report design process to develop dynamic reports for the Web and e-business requirements, as well as high-fidelity printed reports. You can also edit existing JSP and HTML files in Reports Builder to add report layouts.

Refer to *Oracle Reports Tutorial* and *Oracle Reports Building Reports* to learn how to use Reports Builder to develop Web-based or paper-based reports customized to your needs, using data from any data source. Once you have developed a report, you can deploy it to any destination. The sections that follow include the steps to deploy the example report that we have provided, `mypaperreport.rdf`, to Microsoft Excel, Microsoft Word, and e-mail recipients.

13.3.2 Displaying Report Output in Microsoft Excel

This section illustrates the spreadsheet output capability introduced with Oracle Reports 10*g* Release 2 (10.1.2), which enables you to generate output with rich formatting from paper layout reports to Microsoft Excel-compatible HTML format that can be directly opened with Microsoft Excel 2000 or later.

Note: You can also deploy a JSP-based Web report using Reports Server under OC4J to display it in Microsoft Excel in your Web browser. For the steps to do this, refer to the chapter "Building a Report for Spreadsheet Output" in *Oracle Reports Building Reports*.

To display a report in Microsoft Excel using `rwservlet`, perform the following steps:

1. As described in Section 13.2, "Prerequisites", ensure that your OC4J instance is started to enable the deployment of the report.

   Note: In Oracle Developer Suite installations, a standalone OC4J instance is provided for testing purposes to manually deploy an Oracle Reports application for running reports with `rwservlet` and running JSP reports. In Oracle Application Server installations, the `OC4J_BI_Forms` instance automatically deploys an Oracle Reports application.

2. In a Web browser (for example, Internet Explorer), enter either of the following URLs:

   - To generate report output that is displayed in Microsoft Excel in the Web browser:

     ```
     http://hostname:port/reports/rwservlet?report=report_name
     &userid=username/password@database&destype=cache&desformat=spreadsheet
     ```

     For example:

     ```
     http://myas.us.oracle.com:8888/reports/rwservlet?report=mypaperreport.rdf
     &userid=sh/sh@ora10g&destype=cache&desformat=spreadsheet
     ```

     The output should look as shown in Figure 13–1.

   - To generate an HTML file that you can open in Microsoft Excel:

     ```
     http://hostname:port/reports/rwservlet?report=report_name
     &userid=username/password@database&destype=file&desformat=spreadsheet
     &desname=output_filename.htm
     ```

     For example:

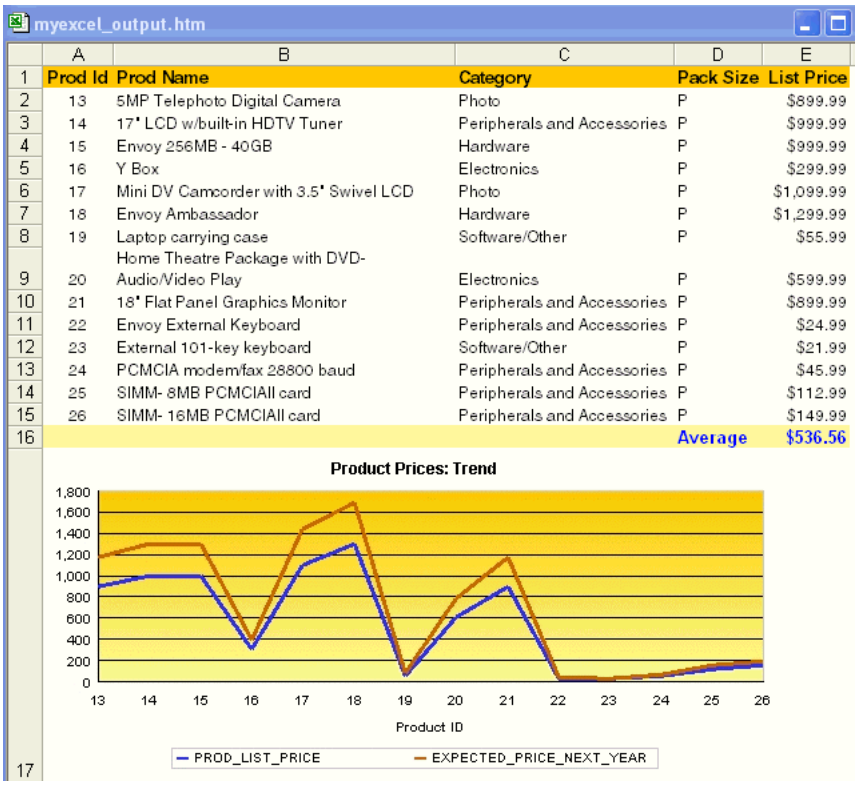
     ```
     http://myas.us.oracle.com:8888/reports/rwservlet?
     report=mypaperreport.rdf&userid=sh/sh@ora10g&destype=file
     &desformat=spreadsheet&desname=C:\temp\myexcel_output.htm
     ```

     Note: If the `rwservlet` command fails, refer to Section 13.4, "Troubleshooting" to resolve the error message that is displayed.

3. If you generated an HTML file that you can open in Microsoft Excel, start Microsoft Excel and open the report output file (for example, `C:\temp\myexcel_output.htm`).

   The output should look as shown in Figure 13–1.

Figure 13–1 Report Output in Microsoft Excel



	A	B	C	D	E
1	Prod Id	Prod Name	Category	Pack Size	List Price
2	13	5MP Telephoto Digital Camera	Photo	P	$899.99
3	14	17" LCD w/built-in HDTV Tuner	Peripherals and Accessories	P	$999.99
4	15	Envoy 256MB - 40GB	Hardware	P	$999.99
5	16	Y Box	Electronics	P	$299.99
6	17	Mini DV Camcorder with 3.5" Swivel LCD	Photo	P	$1,099.99
7	18	Envoy Ambassador	Hardware	P	$1,299.99
8	19	Laptop carrying case	Software/Other	P	$55.99
9	20	Home Theatre Package with DVD-Audio/Video Play	Electronics	P	$599.99
10	21	18" Flat Panel Graphics Monitor	Peripherals and Accessories	P	$899.99
11	22	Envoy External Keyboard	Peripherals and Accessories	P	$24.99
12	23	External 101-key keyboard	Software/Other	P	$21.99
13	24	PCMCIA modem/fax 28800 baud	Peripherals and Accessories	P	$45.99
14	25	SIMM- 8MB PCMCIAII card	Peripherals and Accessories	P	$112.99
15	26	SIMM- 16MB PCMCIAII card	Peripherals and Accessories	P	$149.99
16				Average	$536.56
17					

Note: The graph embedded in the spreadsheet output is a static image file, and is not interactive. You may notice other differences between your report layout in Reports Builder and the Microsoft Excel output. These differences are caused by the way Microsoft Excel interprets the report layout. For detailed usage notes and restrictions about spreadsheet output, see "About spreadsheet output" in the *Oracle Reports online Help*, or the chapter "Advanced Concepts" in *Oracle Reports Building Reports*.

It is not possible to generate spreadsheet output from Reports Builder. Instead, you run the report using Reports Server clients (`rwservlet` or `rwclient`) or `rwrun`, with `desformat=spreadsheet`.

13.3.3 Displaying Report Output in Microsoft Word

Oracle Reports can generate report output to Rich Text Format (RTF) files, containing the formatted data and all objects. RTF can be read by many different word processing software packages, such as Microsoft Word. You can use the software's editing and graphics features to modify and enhance your report output. When you generate your report output to an RTF file, you can distribute the output to any RTF destination, including e-mail, printer, OracleAS Portal, and Web browser.

This section illustrates the RTF output capability available with Oracle Reports, which enables you to generate output from paper-based reports to RTF files that can be opened with Microsoft Word.

To display a report in Microsoft Word using `rwservlet`, perform the following steps:

1. As described in Section 13.2, "Prerequisites", ensure that your OC4J instance is started to enable the deployment of the report.

   Note: In Oracle Developer Suite installations, a standalone OC4J instance is provided for testing purposes to manually deploy an Oracle Reports application for running reports with `rwservlet` and running JSP reports. In Oracle Application Server installations, the `OC4J_BI_Forms` instance automatically deploys an Oracle Reports application.

2. In a Web browser (for example, Internet Explorer), enter either of the following URLs:

   - To generate report output that is displayed in RTF format in the Web browser:

     ```
     http://hostname:port/reports/rwservlet?report=report_name
     &userid=username/password@database&destype=cache&desformat=rtf
     &mimetype=application/msword
     ```

     For example:

     ```
     http://myas.us.oracle.com:8888/reports/rwservlet?report=mypaperreport.rdf
     &userid=sh/sh@ora10g&destype=cache&desformat=rtf
     &mimetype=application/msword
     ```

     Note: The `mimetype=application/msword` option is needed to open the RTF document with Microsoft Word.

     The output should look as shown in Figure 13–2.

   - To generate an RTF file that you can open in Microsoft Word:

     ```
     http://hostname:port/reports/rwservlet?report=report_name
     &userid=username/password@database&destype=file&desformat=rtf
     &desname=output_filename.rtf
     ```

     For example:

     ```
     http://myas.us.oracle.com:8888/reports/rwservlet?
     report=mypaperreport.rdf&userid=sh/sh@ora10g&destype=file
     &desformat=rtf&desname=C:\temp\myword_output.rtf
     ```

     Note: If the `rwservlet` command fails, refer to Section 13.4, "Troubleshooting" to resolve the error message that is displayed.

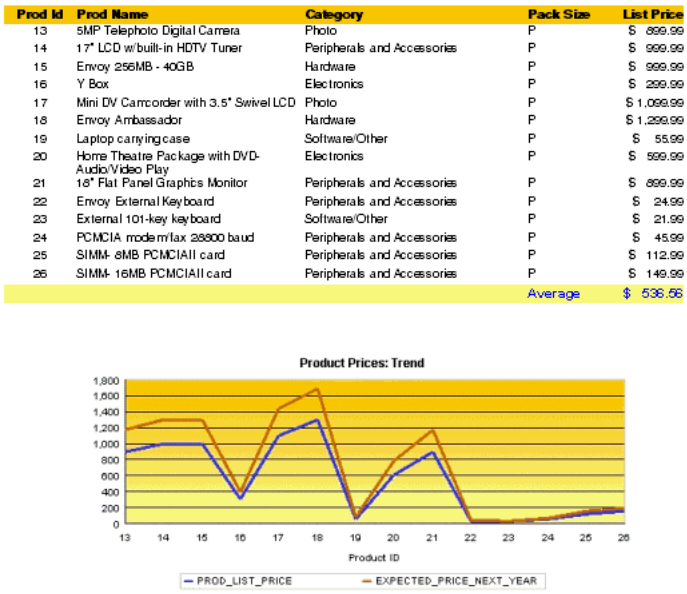
3. If you generated an RTF file that you can open in Microsoft Word, start Microsoft Word and open the report output file (for example, `C:\temp\myword_output.rtf`). The output should look as shown in Figure 13–2.

Alternatively, if you have Oracle Developer Suite installed, you can use Reports Builder to generate RTF output to a file, as follows:

1. Start Reports Builder (either by selecting **Start**, **All Programs**, **iDS Home**, **Reports Developer**, and then **Reports Builder**, or by opening a command prompt window and typing `rwbuilder`).

2. In the Welcome to Reports Builder dialog box, select **Open an existing report**, then click **OK**.
3. In the Open dialog box, locate and open the example report `mypaperreport.rdf`.
4. Select **File**, **Connect** to connect to a database that includes the Sales History schema.
5. Click the **Run Paper Layout** button in the toolbar to run the report.
6. To preview your report output in a Microsoft Word document, select **File**, **Preview Format**, and then **RTF**.
7. To save your report output as an RTF file, select **File**, **Generate to File**, and then **RTF**.
8. In the Save dialog box, specify a location and file name (for example, `C:\temp\myword_output.rtf`). Click **Save**.
9. Start Microsoft Word, and open the report output file (for example, `C:\temp\myword_output.rtf`). The output should look as shown in Figure 13–2.

Figure 13–2 Report Output in Microsoft Word



Prod Id	Prod Name	Category	Pack Size	List Price
13	5MP Telephoto Digital Camera	Photo	P	$ 899.99
14	17" LCD w/built-in HDTV Tuner	Peripherals and Accessories	P	$ 999.99
15	Envoy 256MB - 40GB	Hardware	P	$ 999.99
16	Y Box	Electronics	P	$ 299.99
17	Mini DV Camcorder with 3.5" Swivel LCD	Photo	P	$ 1,099.99
18	Envoy Ambassador	Hardware	P	$ 1,299.99
19	Laptop carrying case	Software/Other	P	$ 55.99
20	Home Theatre Package with DVD-Audio/Video Play	Electronics	P	$ 599.99
21	18" Flat Panel Graphics Monitor	Peripherals and Accessories	P	$ 899.99
22	Envoy External Keyboard	Peripherals and Accessories	P	$ 24.99
23	External 101-key keyboard	Software/Other	P	$ 21.99
24	PCMCIA modem/fax 28800 baud	Peripherals and Accessories	P	$ 45.99
25	SIMM- 8MB PCMCIAII card	Peripherals and Accessories	P	$ 112.99
26	SIMM- 16MB PCMCIAII card	Peripherals and Accessories	P	$ 149.99
			Average	$ 536.56

Note: For detailed usage notes about RTF output, see "About RTF output" in the *Oracle Reports online Help*, or the chapter "Advanced Concepts" in *Oracle Reports Building Reports*.

13.3.4 Sending Report Output to E-Mail Recipients

You can e-mail reports in a variety of formats (including PDF, HTML, HTMLCSS, XML, RTF, ASCII, and delimited text) using any Internet Standard Protocol SMTP mail application, to recipients using e-mail utilities such as Microsoft Outlook.

To e-mail a report, perform the following steps:

1. In your browser (for example, Internet Explorer), enter the following URL:

   ```
   http://hostname:port/reports/rwservlet?report=report_name
   &userid=username/password@database&destype=mail
   &desformat=pdf|html|htmlcss|xml|rtf|ascii|delimited
   &desname=email_address[&from=email_address]
   ```

   For example:

   ```
   http://myas.us.oracle.com:8888/reports/rwservlet?
   report=c:\orawin\examples\mypaperreport.rdf&userid=sh/sh@ora10g
   &from=tom.smith@oracle.com
   &desformat=pdf&destype=mail&desname=jan.jones@oracle.com
   ```

   Note: If the `rwservlet` command fails, refer to Section 13.4, "Troubleshooting" to resolve the error message that is displayed.

2. The e-mail recipient(s) can open the e-mail using their e-mail utility, such as Microsoft Outlook.

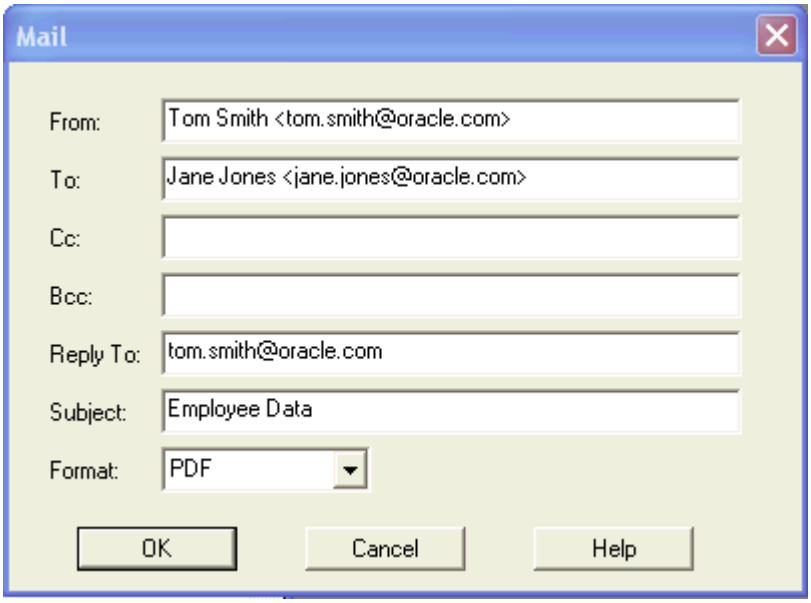
If you have Oracle Developer Suite installed, you can use Reports Builder to test sending e-mail output, as follows:

1. Start Reports Builder (either by selecting **Start**, **All Programs**, **iDS Home**, **Reports Developer**, and then **Reports Builder**, or by opening a command prompt and entering `rwbuilder`).

   Note: If already open, restart Reports Builder to activate the change you made to `rwbuilder.conf`. If your outgoing mail server was already previously defined in `rwbuilder.conf`, it is not necessary to restart Reports Builder.

2. In the Object Navigator, select or open the example report `mypaperreport.rdf`.
3. If you are not already connected, select **File**, **Connect** to connect to a database that includes the Sales History schema.
4. Select **File**, **Mail**.
5. In the Mail dialog box, specify appropriate values in each field for your e-mail report, as shown in Figure 13–3.

Figure 13–3 Mail Dialog Box



6. Click **OK** to send the report in the specified output format to the recipient e-mail address(es).

> **Note:** You can also use Oracle Reports advanced distribution capability to burst and distribute a single report to multiple e-mail recipients in a required format. Using the `mail` element in your distribution XML file, you can specify many additional e-mail options. For more information, see the chapter "Creating Advanced Distributions" in *Oracle Application Server Reports Services Publishing Reports to the Web*.

13.4 Troubleshooting

This section discusses some of the errors or setup issues that you may encounter when performing the steps in this chapter. For additional troubleshooting information, refer to the appendix "Troubleshooting" in *Oracle Application Server Reports Services Publishing Reports to the Web* and to the *Oracle Reports online Help* for error messages, available in Reports Builder and on the Oracle Reports 10*g* page on the Oracle Technology Network (`http://www.oracle.com/technology/products/reports/index.html`).

> **Note:** The recommended way to troubleshoot report issues is to enable tracing and the engine diagnosis option. For more information, refer to the chapter "Configuring OracleAS Reports Services" in *Oracle Application Server Reports Services Publishing Reports to the Web*.

If you encounter any of the following errors while performing the steps in the chapter, the information in this section should help you resolve them:

- REP-51002: Bind to Reports Server failed
- REP-110: Unable to open file 'report_name'

- REP-56048: Engine rwEng-0 crashed
- REP-50159: Executed successfully but there were some errors when distributing the output

REP-51002: Bind to Reports Server failed

Cause: Reports Servlet (`rwservlet`) is not able to locate Reports Server.

Action: This error can occur for a number of reasons. Any of the following actions may resolve the error:

- If you are connected to the network through VPN or if a firewall is used, then the Oracle Reports built-in broadcast mechanism for Reports Server discovery will not work. In this case, you must configure the discovery mechanism to use the Common Object Service (COS) naming service `orbd`, provided by Sun Microsystems' JDK. For details on using the COS naming service, refer to the chapter "Configuring OracleAS Reports Services" in *Oracle Application Server Reports Services Publishing Reports to the Web*.
- If you are using the Oracle Reports in-process server, check that `SERVER_IN_PROCESS=YES` in the servlet configuration file (*`ORACLE_HOME`*`\reports\conf\rwservlet.properties`).
- If you are using a standalone server, check if the server is running through Oracle Enterprise Manager 10*g*.
- If the Oracle Reports built-in broadcast mechanism for Reports Server discovery (`multicast`) is used, find out how much time it takes to locate a Reports Server in your network using the `rwdiag` utility. Reports Servlet (`rwservlet`) may be timing out while locating the Reports Server. This scenario is most likely to occur when the Reports Server is running on a different network subnet.

  For example:

```
$ rwdiag.sh -find server_name

server_name found in the network
Time taken - 101 millisecond
```

If the time taken is more than (`timeout * retry`) values in your `rwnetwork.conf` file, then increase the `timeout` value in `rwnetwork.conf` and restart your Reports Server and `OC4J_BI_Forms` instance (or standalone OC4J in Oracle Developer Suite installations). This scenario is more likely to occur when Reports Server is running on a different network subnet.

REP-110: Unable to open file '*report_name*'

Cause: Reports Server is not able to locate the report definition file.

Action: Modify `REPORTS_PATH` to include the folder that contains your report definition file.

For example, suppose `c:\myreports` contains your report definition files:

```
REPORTS_PATH=c:\myreports;C:\OraHome_5\reports\templates...
```

REP-56048: Engine rwEng-0 crashed

Cause: This error occurs if memory allocated for the engine process (JVM) is not sufficient to generate spreadsheet output. If your spreadsheet report contains a large number of pages (several hundreds of pages), then default heap memory allocated is not sufficient to generate the report.

Action: Increase the heap memory for the engine process (JVM) by setting the `jvmOptions` attribute of the `engine` element in the server configuration file (*ORACLE_HOME*`\reports\conf\`*server_name*`.conf`). For example:

```
<engine id="rwEng" jvmOptions="-Xmx512M"
class="oracle.reports.engine.EngineImpl"...
```

For more details on setting the `jvmOptions` attribute, refer to the chapter "Configuring OracleAS Reports Services" in *Oracle Application Server Reports Services Publishing Reports to the Web*.

Note: The Reports engine may crash for a number of reasons. If you encounter an engine hang or crash, or a job hangs in Reports Server, the recommended resolution is to set the `engineResponseTimeout` attribute of the `engine` element in the server configuration file (*ORACLE_HOME*`\reports\conf\`*server_name*`.conf`), as described in the chapter "Configuring OracleAS Reports Services" in *Oracle Application Server Reports Services Publishing Reports to the Web*.

REP-50159: Executed successfully but there were some errors when distributing the output

This error can occur for a number of reasons when sending report output to an e-mail destination.

Cause 1: Mail server is not configured properly.

Action 1: Configure the mail server in the server configuration file (*ORACLE_HOME*`\reports\conf\`*server_name*`.conf`) and the Reports Builder configuration file (*ORACLE_HOME*`\reports\conf\rwbuilder.conf`). The `pluginParam` element should specify the outgoing SMTP server name. For example:

```
<pluginParam name="mailServer">smtpserver.mycompany.com</pluginParam>
```

For more details, refer to the chapter "Configuring OracleAS Reports Services" in *Oracle Application Server Reports Services Publishing Reports to the Web*.

Cause 2: The mail server is not responding or is not up and running.

Action 2: Check if the mail server is up and running and is responding in a timely manner. You can use your Microsoft Outlook client to connect to the mail server and check the status.

Cause 3: A valid recipient e-mail address is not specified for the `desname` keyword.

Action 3: Specify a valid recipient e-mail address for the `desname` keyword.

Cause 4: The mail server is SSL-enabled.

Action 4: Use a non-SSL mail server. Oracle Reports does not support SSL-enabled mail servers to send e-mail.

13.5 Related Documentation

- *Oracle Reports Building Reports*
- *Oracle Application Server Reports Services Publishing Reports to the Web*
- *Oracle Reports online Help*

Part III

Appendixes

Part III contains the following appendix:

- Appendix A, "Code Examples"

A

Code Examples

This appendix contains the following sections:

A.1 Contents of the AutoLoanSmartDocument.cs File

Example A–1 shows the contents of the `AutoLoanSmartDocument.cs` file. See Section 4.3.3, "Creating a Smart Document Form" for more details.

Example A–1 AutoLoanSmartDocument.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Serialization;
using Microsoft.Office.Interop.SmartTag;
using Microsoft.Office.Interop.Word;

namespace AutoLoanSmartDocument
{
   //BASE CLASS
   public class clsActions : Microsoft.Office.Interop.SmartTag.ISmartDocument
   {
      //CONSTANTS
      //You need one constant for the schema namespace, one constant for each
      //of the schema elements for which you want to provide smart document
controls
      //and actions, and one constant for the total number of schema elements
      //for which there are associated actions.
      //Because XML is case-sensitive, the values
      //of these constants must be exact in both spelling and case.
      //Namespace constant
      const String cNAMESPACE = "http://www.autoloan.com/ns/autoloan";
      //Element constants
```

```
      const String cAutoLoanRootElemName = cNAMESPACE + "#loanApplication";
      //Number of types (or element constants)
      const Int32 cTYPES = 2;
      public void SmartDocInitialize(string ApplicationName, object Document,
string SolutionPath, string SolutionRegKeyRoot)
      {
      }
      //SmartDocXMLTypeCount
      public int SmartDocXmlTypeCount
      {
         get
         {
            String message = "SmartDocXmlTypeName" + cTYPES;
            return cTYPES;
         }
      }
      //SmartDocXMLTypeName
      public string get_SmartDocXmlTypeName(int XMLTypeID)
      {
         String strTypeName = "";
         String message = "SmartDocXmlTypeName" + XMLTypeID;
         switch (XMLTypeID) {
               case 1:
                  strTypeName = cAutoLoanRootElemName;
                  break;
               default:
                  break;
         }
         return strTypeName;
      }
      //SmartDocXMLTypeCaption
      public string get_SmartDocXmlTypeCaption(int XMLTypeID, int LocaleID)
      {
         String strTypeCaption = "";

         switch (XMLTypeID) {
            case 1:
               strTypeCaption = "Please submit loan application";
               break;
            default:
               break;
         }

         return strTypeCaption;
      }
      //ControlCount
      public int get_ControlCount(string XMLTypeName)
      {
         Int32 intNumberOfControls = 0;

         switch (XMLTypeName) {
            case cAutoLoanRootElemName:
               intNumberOfControls = 1;
               break;
            default:
               break;
         }

         return intNumberOfControls;
      }
```

```
      //ControlID
      //The ControlID for the first control you add will be 1.
      //For more information on specifying the ControlID, see the ControlID
reference
      //topic in the References section of this SDK.
      public int get_ControlID(string XMLTypeName, int ControlIndex)
      {
         Int32 intControlID = 0;
         switch (XMLTypeName) {

            case cAutoLoanRootElemName:
               intControlID = 1;
               break;
            default:
               break;
         }

         return intControlID;
      }
      //ControlNameFromID
      public string get_ControlNameFromID(int ControlID)
      {
         String strControlName = "";
         strControlName = cNAMESPACE + ControlID;
         return strControlName;
      }
      //ControlCaptionFromID
      public string get_ControlCaptionFromID(int ControlID,
         string ApplicationName, int LocaleID, string Text,
         string Xml, object Target)
      {
         String strControlCaption = "";
         switch (ControlID){
            case 1:
               strControlCaption = "Submit for Approval";
               break;
            default:
               break;
         }
         return strControlCaption;
      }
      //ControlTypeFromID
      public C_TYPE get_ControlTypeFromID(int ControlID,
         string ApplicationName, int LocaleID)
      {
         C_TYPE type = new C_TYPE();
         switch (ControlID)
         {
            case 1:
               type = C_TYPE.C_TYPE_BUTTON;
               break;
            default:
               break;
         }
         return type;
      }
      public void PopulateHelpContent(int ControlID,
         string ApplicationName, int LocaleID, string Text, string Xml,
         object Target, Microsoft.Office.Interop.SmartTag.ISmartDocProperties
Props, ref string Content)
```

```
      {
         switch (ControlID)
         {
            case 1:
               Content = "This document is an XML smart document that submits an
loan application";
               break;
            default:
               break;
         }
      }
      // OnSubmitDocument
      public void InvokeControl(int ControlID,
         string ApplicationName, object Target, string Text, string Xml, int
LocaleID)
      {
         try {
            // Get the xml node
            Microsoft.Office.Interop.Word.Range objRange =
(Microsoft.Office.Interop.Word.Range)Target;
            XMLNode objNode = objRange.XMLNodes[1];
            // MessageBox.Show (objNode.BaseName, "Base name",
MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            // MessageBox.Show (objNode.get_XML(true), "Data xml from objNode",
MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            // From the xml node, create a AutoLoan object for the web service
input
            String xmlString = objNode.get_XML(true).ToString();
            // MessageBox.Show (xmlString, "xmlString",
MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            XmlDocument doc = new XmlDocument();
            doc.LoadXml(xmlString);
            XmlNode root = doc.DocumentElement;
            // MessageBox.Show (root.ToString(), "loan application",
MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            // MessageBox.Show ("Serialization to XML doc success", "My
Application", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            XmlReader xmlReader = new XmlNodeReader(doc);
            XmlSerializer serializer = new
XmlSerializer(typeof(LoanApplicationType));
            LoanApplicationType loanApplicationMessage = (LoanApplicationType)
serializer.Deserialize(xmlReader);
            // MessageBox.Show (loanApplicationMessage.ToString(), "loan
application", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            // MessageBox.Show ("Serialization to input message success", "My
Application", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            // Invoke the web service
            AutoLoanFlowBinding autoLoanFlowBindingProxy = new
AutoLoanFlowBinding();
            autoLoanFlowBindingProxy.initiate(loanApplicationMessage);
            MessageBox.Show ("The loan application was successfully submitted for
approval", "Application Submission Status", MessageBoxButtons.OKCancel,
MessageBoxIcon.Asterisk);
         } catch(XmlException xe){
           MessageBox.Show (xe.Message, "XML Parse Error",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
         } catch(InvalidOperationException ioe){
           MessageBox.Show (ioe.InnerException.Message, "XML Serialization Error",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
         } catch(Exception ioe){
```

```
            MessageBox.Show (ioe.Message, "XML Serialization Error",
                             MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
      }
      public void PopulateCheckbox(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props, ref bool
Checked)
      {
         // do nothing
      }
      public void PopulateTextboxContent(int ControlID, string ApplicationName,
int LocaleID,
         string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props, ref string
Value)
      {
         // do nothing
      }
      public void PopulateListOrComboContent(int ControlID, string
ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props, ref
System.Array List,
         ref int Count, ref int InitialSelected)
      {
         switch (ControlID)
         {
            case 101:
                 Count = 6;
                 List.SetValue("AirFare", 1);
                 List.SetValue("Rental", 2);
                 List.SetValue("Hotel", 3);
                 List.SetValue("Meals", 4);
                 List.SetValue("Phone", 5);
                 List.SetValue("Other", 6);
                 InitialSelected = -1;
                 break;
            default:
              break;
         }
      }
      public void OnCheckboxChange(int ControlID, object Target, bool Checked)
      {
         // do nothing
      }
      public void OnTextboxContentChange(int ControlID, object Target, string
Value)
      {
         // do nothing
      }
      public void OnListOrComboSelectChange(int ControlID, object Target, int
Selected, string Value)
      {
         switch (ControlID)
         {
            case 101:
              Range objRange = (Range) Target;
              objRange.XMLNodes[1].Range.Text = Value;
              break;
```

```
            default:
              break;
         }
      }
      public void PopulateDocumentFragment(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
         ref string DocumentFragment)
      {
         // do nothing
      }
      public void PopulateActiveXProps(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties ActiveXPropBag)
      {
         // do nothing
      }
      public void PopulateImage(int ControlID, string ApplicationName, int
LocaleID,
         string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
         ref string ImageSrc)
      {
         // do nothing
      }
      public void ImageClick(int ControlID, string ApplicationName, object Target,
         string Text, string Xml, int LocaleID, int XCoordinate, int YCoordinate)
      {
         // do nothing
      }
      public void PopulateRadioGroup(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
         ref System.Array List, ref int Count, ref int InitialSelected)
      {
         // do nothing
      }
      public void OnRadioGroupSelectChange(int ControlID, object Target, int
Selected, string Value)
      {
         // do nothing
      }
      public void OnPaneUpdateComplete(object Document)
      {
         // do nothing
      }
      public void PopulateOther(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props)
      {
         // do nothing
      }
   }
}
```

A.2 Contents of the ManagedManifest.xml File for Chapter 4

Example A–2 shows the contents of the `ManagedManifest.xml` file. See Section 4.3.3, "Creating a Smart Document Form" for more details.

Example A–2 ManagedManifest.xml for Chapter 4

```
<SD:manifest
xmlns:SD="http://schemas.microsoft.com/office/xmlexpansionpacks/2003">
  <SD:version>1.1</SD:version>
  <SD:updateFrequency>20160</SD:updateFrequency>
  <SD:uri>http://www.autoloan.com/ns/autoloan</SD:uri>
  <SD:solution>
    <SD:solutionID>AutoLoanSmartDocument.clsActions</SD:solutionID>
    <SD:type>smartDocument</SD:type>
    <SD:alias lcid="*">Smart Word Document to submit loan application for a
car</SD:alias>
    <SD:file>
      <SD:type>solutionActionHandler</SD:type>
      <SD:version>1.0</SD:version>
      <SD:filePath>AutoLoanSmartDocument.dll</SD:filePath>
      <SD:CLSNAME>AutoLoanSmartDocument.clsActions</SD:CLSNAME>
      <SD:runFromServer>True</SD:runFromServer>
      <SD:managed/>
    </SD:file>
  </SD:solution>
  <SD:solution>
    <SD:solutionID>schema</SD:solutionID>
    <SD:type>schema</SD:type>
    <SD:alias lcid="*">Sample schema</SD:alias>
    <SD:file>
      <SD:type>schema</SD:type>
      <SD:version>1.0</SD:version>
      <SD:filePath>AutoLoanTypes.xsd</SD:filePath>
      <SD:runFromServer>True</SD:runFromServer>
    </SD:file>
  </SD:solution>
</SD:manifest>
```

A.3 Contents of the ManagedManifest.xml File for Chapter 6

Example A–3 shows the contents of the `ManagedManifest.xml` file. See Section 6.3.4, "Attaching the XML Schema and the Expansion Pack to the Smart Document" for more details.

Example A–3 ManagedManifest.xml for Chapter 6

```
<SD:manifest
xmlns:SD="http://schemas.microsoft.com/office/xmlexpansionpacks/2003">
  <SD:version>1.1</SD:version>
  <SD:updateFrequency>20160</SD:updateFrequency>
  <SD:uri>http://xmlns.oracle.com/SecureSmartDocument</SD:uri>
  <SD:solution>
    <SD:solutionID>SecureSmartDocument.clsActions</SD:solutionID>
    <SD:type>smartDocument</SD:type>
    <SD:alias lcid="*">Smart Word Document </SD:alias>
    <SD:file>
      <SD:type>solutionActionHandler</SD:type>
      <SD:version>1.0</SD:version>
```

```
      <SD:filePath>SecureDoc.dll</SD:filePath>
      <SD:CLSNAME>SecureSmartDocument.clsActions</SD:CLSNAME>
      <SD:runFromServer>True</SD:runFromServer>
      <SD:managed/>
    </SD:file>
  </SD:solution>
  <SD:solution>
    <SD:solutionID>schema</SD:solutionID>
    <SD:type>schema</SD:type>
    <SD:alias lcid="*">Sample schema</SD:alias>
    <SD:file>
      <SD:type>schema</SD:type>
      <SD:version>1.0</SD:version>
      <SD:filePath>SecureDocument.xsd</SD:filePath>
      <SD:runFromServer>True</SD:runFromServer>
    </SD:file>
  </SD:solution>
</SD:manifest>
```

A.4 Contents of the AutoLoanTypes.xsd File

Example A–4 shows the contents of the `AutoLoanTypes.xsd` file. See Section 4.3.3, "Creating a Smart Document Form" for more details.

Example A–4 AutoLoanTypes.xsd

```
<?xml version="1.0"?>
<schema attributeFormDefault="qualified" elementFormDefault="qualified"
 targetNamespace="http://www.autoloan.com/ns/autoloan"
 xmlns:tns="http://www.autoloan.com/ns/autoloan"
 xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="loanApplication" type="tns:LoanApplicationType"/>
  <element name="loanOffer" type="tns:LoanOfferType"/>
  <element name="loan" type="tns:LoanType"/>
  <complexType name="LoanType">
      <sequence>
        <element ref="tns:loanApplication"/>
        <element ref="tns:loanOffer"/>
      </sequence>
  </complexType>
  <complexType name="LoanOfferType">
      <sequence>
<element name="providerName" type="string"/>
<element name="selected" type="boolean"/>
<element name="approved" type="boolean"/>
<element name="APR" type="double"/>
      </sequence>
  </complexType>
  <complexType name="LoanApplicationType">
      <sequence>
<element name="SSN" type="string"/>
<element name="email" type="string"/>
<element name="customerName" type="string"/>
<element name="customerAge" type="string"/>
<element name="customerAnnualIncome" type="string"/>
<element name="city" type="string"/>
<element name="state" type="string"/>
<element name="country" type="string"/>
<element name="loanAmount" type="double"/>
```

```
<element name="carMake" type="string"/>
<element name="carModel" type="string"/>
<element name="carYear" type="string"/>
<element name="creditRating" type="int"/>
      </sequence>
  </complexType>
</schema>
```

A.5 Contents of the SecureDocument.xsd File

Example A–5 shows the contents of the `SecureDocument.xsd` file. See Section 6.3.4, "Attaching the XML Schema and the Expansion Pack to the Smart Document" for more details.

Example A–5 SecureDocument.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<schema targetNamespace="http://xmlns.oracle.com/SecureSmartDocument"
     xmlns:tns="http://xmlns.oracle.com/SecureSmartDocument"
     xmlns="http://www.w3.org/2001/XMLSchema"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified" blockDefault="#all">
  <element name="Report">
    <complexType>
      <sequence>
        <element name="name" type="string" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </complexType>
  </element>

</schema>
```

A.6 Contents of the SecureSmartDocument.cs File

Example A–6 shows the contents of the `SecureSmartDocument.cs` file. See Section 6.3.3, "Creating the Smart Document DLL" for more details.

> **Note:** For information on how to locate the example files, see Accessing the Demonstration Support Files in the Preface.

Example A–6 SecureSmartDocument.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Serialization;
using Microsoft.Office.Interop.SmartTag;
using Microsoft.Office.Interop.Word;
using Microsoft.Web.Services2.Security.Tokens;
using Microsoft.Web.Services2.Security.X509;
using Microsoft.Web.Services2.Security;


namespace SecureSmartDocument
{
```

```
   //BASE CLASS
   public class clsActions : Microsoft.Office.Interop.SmartTag.ISmartDocument
   {

      //CONSTANTS
      //You need one constant for the schema namespace, one constant for each
      //of the schema elements for which you want to provide smart document
controls
      //and actions, and one constant for the total number of schema elements
      //for which there are associated actions.

      //Because XML is case-sensitive, the values
      //of these constants must be exact in both spelling and case.

      //Namespace constant
      const String cNAMESPACE = "http://xmlns.oracle.com/SecureSmartDocument";

      //Element constants

      const String cElemName = cNAMESPACE + "#name";

      //Number of types (or element constants)
      const Int32 cTYPES = 1;


      public void SmartDocInitialize(string ApplicationName, object Document,
string SolutionPath, string SolutionRegKeyRoot)
      {

      }

      //SmartDocXMLTypeCount
      public int SmartDocXmlTypeCount
      {
         get
         {
            String message = "SmartDocXmlTypeName" + cTYPES;
            return cTYPES;
         }
      }

      //SmartDocXMLTypeName
      public string get_SmartDocXmlTypeName(int XMLTypeID)
      {
         String strTypeName = "";

         String message = "SmartDocXmlTypeName" + XMLTypeID;

         switch (XMLTypeID)
         {
            case 1:
               strTypeName = cElemName;
               break;

            default:
               break;
         }
         return strTypeName;
      }
```

```
      //SmartDocXMLTypeCaption
      public string get_SmartDocXmlTypeCaption(int XMLTypeID, int LocaleID)
      {
         String strTypeCaption = "";

         switch (XMLTypeID)
         {
            case 1:
               strTypeCaption = "Please Enter Name  ";
               break;

            default:
               break;
         }
         return strTypeCaption;
      }

      //ControlCount
      public int get_ControlCount(string XMLTypeName)
      {
         Int32 intNumberOfControls = 0;

         switch (XMLTypeName)
         {
            case cElemName:
               intNumberOfControls = 1;
               break;

            default:
               break;
         }
         return intNumberOfControls;
      }

      //ControlID
      //The ControlID for the first control you add will be 1.
      //For more information on specifying the ControlID, see the ControlID
reference
      //topic in the References section of this SDK.
      public int get_ControlID(string XMLTypeName, int ControlIndex)
      {
         Int32 intControlID = 0;

         switch (XMLTypeName)
         {

            case cElemName:
               intControlID = 1;
               break;

            default:
               break;
         }
         return intControlID;
      }

      //ControlNameFromID
      public string get_ControlNameFromID(int ControlID)
      {
```

```
            String strControlName = "";
            strControlName = cNAMESPACE + ControlID;
            return strControlName;
         }

         //ControlCaptionFromID
         public string get_ControlCaptionFromID(int ControlID,
            string ApplicationName, int LocaleID, string Text,
            string Xml, object Target)
         {
            String strControlCaption = "";

            switch (ControlID)
            {
               case 1:
                  strControlCaption = "Name";
                  break;

               default:
                  break;
            }
            return strControlCaption;
         }

         //ControlTypeFromID
         public C_TYPE get_ControlTypeFromID(int ControlID,
            string ApplicationName, int LocaleID)
         {
            C_TYPE type = new C_TYPE();

            switch (ControlID)
            {
               case 1:
                  type = C_TYPE.C_TYPE_TEXTBOX;
                  break;

               default:
                  break;
            }
            return type;
         }

         public void PopulateHelpContent(int ControlID,
            string ApplicationName, int LocaleID, string Text, string Xml,
            object Target, Microsoft.Office.Interop.SmartTag.ISmartDocProperties
Props, ref string Content)
         {
            switch (ControlID)
            {
               case 1:
                  Content = "This document is an XML smart document";
                  break;

               default:
                  break;
            }
         }

         public void InvokeControl(int ControlID,
            string ApplicationName, object Target, string Text, string Xml, int
```

```
LocaleID)
      {
         //do nothing
      }

      public void PopulateCheckbox(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props, ref bool
Checked)
      {
         // do nothing
      }

      public void PopulateTextboxContent(int ControlID, string ApplicationName,
int LocaleID,
         string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props, ref string
Value)
      {
         //do nothing
      }

      public void PopulateListOrComboContent(int ControlID, string
ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props, ref
System.Array List,
         ref int Count, ref int InitialSelected)
      {
         //do nothing
      }

      public void OnCheckboxChange(int ControlID, object Target, bool Checked)
      {
         // do nothing
      }

      public void OnTextboxContentChange(int ControlID, object Target, string
Value)
      {
         // Add code later ….
      }


      public void OnListOrComboSelectChange(int ControlID, object Target, int
Selected, string Value)
      {

      }

      public void PopulateDocumentFragment(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
         Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
         ref string DocumentFragment)
      {
         // do nothing
      }

      public void PopulateActiveXProps(int ControlID, string ApplicationName,
         int LocaleID, string Text, string Xml, object Target,
```

```
            Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
            Microsoft.Office.Interop.SmartTag.ISmartDocProperties ActiveXPropBag)
         {
            // do nothing
         }

         public void PopulateImage(int ControlID, string ApplicationName, int
LocaleID,
            string Text, string Xml, object Target,
            Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
            ref string ImageSrc)
         {
            // do nothing
         }

         public void ImageClick(int ControlID, string ApplicationName, object Target,
            string Text, string Xml, int LocaleID, int XCoordinate, int YCoordinate)
         {
            // do nothing
         }

         public void PopulateRadioGroup(int ControlID, string ApplicationName,
            int LocaleID, string Text, string Xml, object Target,
            Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props,
            ref System.Array List, ref int Count, ref int InitialSelected)
         {
            // do nothing
         }

         public void OnRadioGroupSelectChange(int ControlID, object Target, int
Selected, string Value)
         {
            // do nothing
         }

         public void OnPaneUpdateComplete(object Document)
         {
            // do nothing
         }

         public void PopulateOther(int ControlID, string ApplicationName,
            int LocaleID, string Text, string Xml, object Target,
            Microsoft.Office.Interop.SmartTag.ISmartDocProperties Props)
         {
            // do nothing
         }
      }
}
```

A.7 Contents of the UsernameTokenDialog.cs File

Example A–7 shows the contents of `UsernameTokenDialog.cs`. See Section 6.3.5.1.1, "Securing the Client Side" for more details.

> **Note:** For information on how to locate the example files, see Accessing the Demonstration Support Files in the Preface.

Example A–7 UsernameTokenDialog.cs

```
using System;
using System.Windows.Forms;
namespace SecureSmartDocument
{

   /// <summary>
   /// Dialog.
   /// </summary>
   class UsernameTokenDialog : System.Windows.Forms.Form
   {
      private String uname;
      private System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.TextBox textBox2;
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.Button button2;
      private System.Windows.Forms.Label label1;
      private System.Windows.Forms.Label label2;
      private System.Windows.Forms.Label label3;
      private String pwd;
      public UsernameTokenDialog():base()
      {
         InitializeComponent();
      }
      private void InitializeComponent()
      {
         this.textBox1 = new System.Windows.Forms.TextBox();
         this.textBox2 = new System.Windows.Forms.TextBox();
         this.button1 = new System.Windows.Forms.Button();
         this.button2 = new System.Windows.Forms.Button();
         this.label1 = new System.Windows.Forms.Label();
         this.label2 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.SuspendLayout();
         //
         // textBox1
         //
         this.textBox1.Location = new System.Drawing.Point(120, 104);
         this.textBox1.Name = "textBox1";
         this.textBox1.TabIndex = 0;
         this.textBox1.Text = "";
         this.textBox1.TextChanged += new System.EventHandler(this.textBox1_
TextChanged);
         //
         // textBox2
         //
         this.textBox2.Location = new System.Drawing.Point(408, 104);
         this.textBox2.Name = "textBox2";
         this.textBox2.PasswordChar = '*';
         this.textBox2.TabIndex = 1;
         this.textBox2.Text = "";
         this.textBox2.TextChanged += new System.EventHandler(this.textBox2_
TextChanged);
         //
         // button1
         //
         this.button1.Location = new System.Drawing.Point(96, 192);
         this.button1.Name = "button1";
         this.button1.Size = new System.Drawing.Size(88, 32);
         this.button1.TabIndex = 2;
```

```
            this.button1.Text = "Ok";
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(320, 192);
            this.button2.Name = "button2";
            this.button2.TabIndex = 3;
            this.button2.Text = "Cancel";
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // label1
            //
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif",
14.25F, ((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Underline))), System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.label1.Location = new System.Drawing.Point(176, 24);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(216, 48);
            this.label1.TabIndex = 4;
            this.label1.Text = "User Credentials";
            //
            // label2
            //
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.label2.Location = new System.Drawing.Point(16, 104);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(80, 16);
            this.label2.TabIndex = 5;
            this.label2.Text = "Username : ";
            //
            // label3
            //
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.label3.Location = new System.Drawing.Point(296, 104);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(88, 32);
            this.label3.TabIndex = 6;
            this.label3.Text = "Password :";
            //
            // UsernameTokenDialog
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(576, 266);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.textBox1);
            this.Name = "UsernameTokenDialog";
            this.ResumeLayout(false);
        }
```

```
        private void button1_Click(object sender, System.EventArgs e)
        {
           Console.WriteLine(" Ok button clicked ");
           uname = textBox1.Text;
           pwd = textBox2.Text;
           Console.WriteLine(" Username : "+ uname + " pwd : " + pwd);

           this.Close();
           this.DialogResult = DialogResult.OK;
        }

        private void button2_Click(object sender, System.EventArgs e)
        {
           uname = null;
           pwd = null;
           this.DialogResult = DialogResult.Cancel;
        }
        private void textBox1_TextChanged(object sender, System.EventArgs e)
        {

        }
        private void textBox2_TextChanged(object sender, System.EventArgs e)
        {

        }

        public void setDefaultUsername (String uname)
        {
           textBox1.Text = uname;
        }
        public String getUsername()
        {
           return uname;
        }
        public String getPassword()
        {
           return pwd;
        }
     }
  }
```

Index

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X