
1. Product Guide

How the Product Guide is organized

The Product Guide section of the Desktop Integration Developer's Guide is organized to provide an executive level overview of the product, followed by sections that describe the product features. Each section of features is followed by case studies that provide step-by-step examples that show you how to use each feature.

The features are introduced in order of increasing complexity through the guide. After starting with a basic introduction, you will progress through the features and be able to create a production quality desktop integration solution by the time you finish the Product Guide.

The Product Guide is divided in the following sections:

Introduction to Web ADI

This section provides a high-level explanation of what desktop integration is and how Web ADI provides the ability to create your own solutions.

Getting started

This section gets you started by showing you how to define several working integrators and understand the concepts behind them.

Improving spreadsheet presentation

This section shows how easily the working integrator defined in the previous section can be improved to a professional level of presentation.

Enforcing business rules

This section shows how business rules can be defined and enforced within your solution.

Leveraging E-Business Suite features

This section describes the built in support Web ADI provides for E-Business Suite features including flexfields and concurrent programs.

Integrating with your application

This section shows how to include desktop integration within your own applications page flows and forms.

Using additional capabilities

This section describes how to configure more complex features including support for multiple tables, PL/SQL import and master-detail data structures.

Going beyond built-in capabilities

This section explains how you can extend Web ADI beyond its built-in capabilities using Java extensions.

Introduction to Web ADI

What is Desktop Integration?

Desktop integration is the use of desktop applications, such as Microsoft Office, to access and update information in the Oracle E-Business Suite of applications. One example is a financial controller using Microsoft Excel to analyze and update Oracle General Ledger financial budgets.

Note. Desktop integration using Web ADI is available for many desktop applications, not just Microsoft Excel. For the sake of simplicity, the example of a spreadsheet is used throughout this guide.

End-users prefer desktop integration as it provides them access to E-Business Suite applications using desktop software that they are familiar with. Additionally, products like Microsoft Excel provide cut and paste as well as other facilities that make it easy to manipulate large amounts of data.

The end-user is able to access desktop integration using Oracle Self-Service Web Applications. The only software they need installed on their PC is the desktop application, for example, Microsoft Excel and a web browser such as Internet Explorer.

To access desktop integration within Self-Service Web Applications, the user has to log on using the E-Business Suite Home page. A set of page flows is displayed to the user. The user is able to select the desktop application, as well as other options.

Here is a screen shot of the Self-Service Web Applications page flow used to select the desktop application.

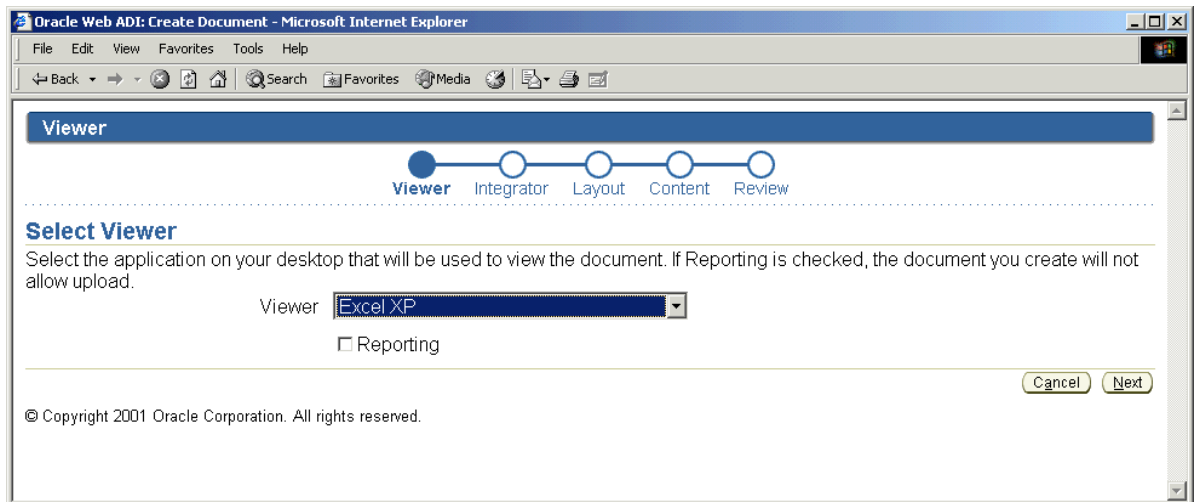


Figure 1-1, Select Desktop Application page (SelectViewer.gif)

Once the end-user completes the selections using the Self-Service Web Applications page flows, the document is created dynamically and opened on the desktop of the end-users PC:

Microsoft Excel - Book1

File Edit View Insert Format Tools Data Oracle Window Help

17 =

A	B	C	D	E	F	G	H	I	J	K	M	N
2	View context		View header									
3	Balance Type		Actual									
4	Database		WEBADI.US.ORACLE.COM									
5	Set of Books		Vision Operations (USA)									
6												
7	Category		* List - Text									
8	Source		* List - Text									
9	Currency		* List - Text									
10	Accounting Date		* List - Date									
11												
12	Upl Co	Dpt Acct Sub Prd	Debit	Credit	Messages							
13	* List - Text		* Number	* Number								
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24	Totals:		0	0								

Tip: This is not the end of the Template. Unprotect the sheet and insert as many rows as needed.

Sheet1 Sheet2 Sheet3

Figure 1-2, Generated journals spreadsheet (GeneratedSpreadsheet.gif)

The end-user is displayed a layout of columns based on the business rules of the application. In the previous example, a Functional Actuals layout for Oracle General Ledger is displayed.

The layout has various cell groupings as you can see from the example. The top group has read only values, for example, Balance Type, Database and Set of Books. The middle group has data entry cells for Category, Source, Currency and Accounting Date. The bottom group has multiple input lines for the Accounting Flexfield, Debit and Credit.

In this example, a number of the cells include list of values functionality that the end-user can access by double clicking on the cell. The list of values will display based on the business logic of the application, in this case Oracle General Ledger. The end-user can proceed with data entry, or if unsure of the required values, double click on cells to display lists of values. These lists show the valid data for the particular cell. Here is an example of the list of values shown for the *Category* cell:

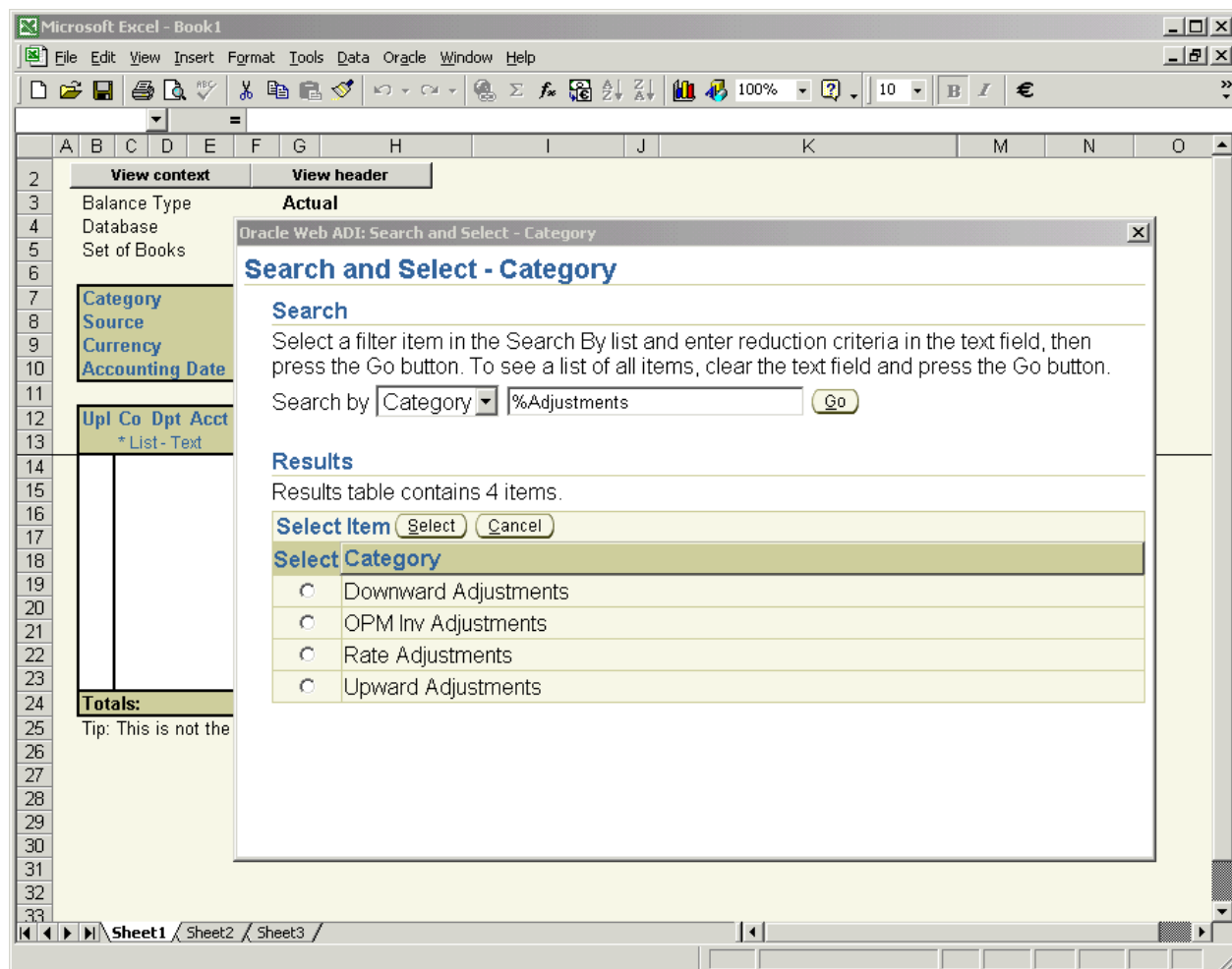


Figure 1-3, Category list of values (CategoryListOfValues.gif)

When the end-user has completed data entry in the spreadsheet, they can either save the spreadsheet for later use, or they can select the upload option.

The upload option will upload the data into the E-Business Suite database according to the business rules of the application. If any data in the spreadsheet does not comply with the business rules, the upload process will highlight the row containing error and allow the user to re-enter the data.

When the end-user selects the upload option, the following upload parameters page is displayed:

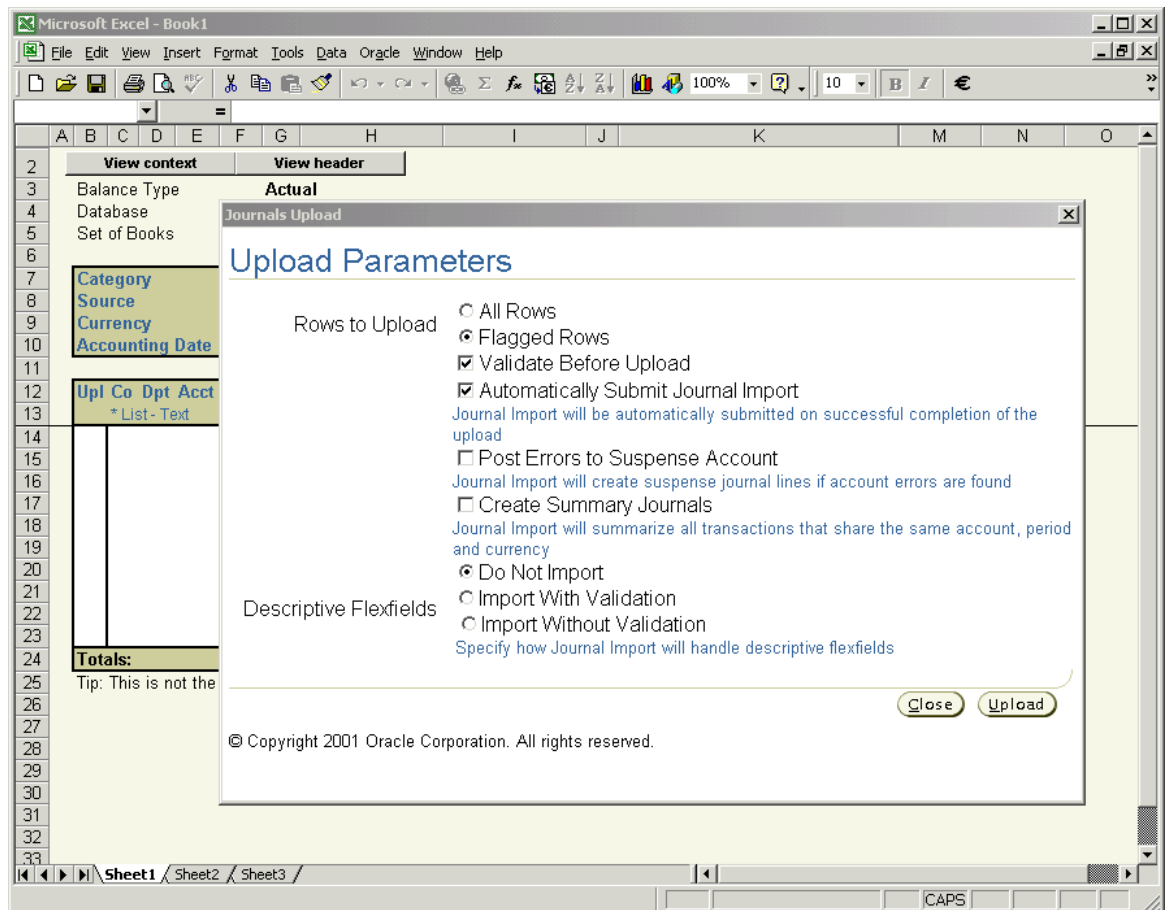


Figure 1-4, Journals Upload Parameters (UploadParameters.gif)

This page showcases the high degree of integration with Oracle E-Business Suite provided by Web ADI.

Notice the parameters specific to Oracle General Ledger; Automatically Submit Journal Import, Post Errors to Suspense Account, Create Summary Journals.

When using Web ADI to create your own desktop integration solutions, you can configure the upload process, and many other features, specific to the business requirements of your solution.

Populate with data

Another feature of desktop integration using Web ADI is the ability to include data in the spreadsheet as it is generated. The data may be obtained from a variety of sources including database queries and text files. This means the General Ledger - Journals layout that you've seen in these examples can be populated with data automatically. The end-user can then analyze and change the data as required before uploading the data to the E-Business Suite.

Similarly, you can also create spreadsheets that are used only for reporting purposes. This is referred to as spreadsheet based reporting. With spreadsheet based reporting, information from the database, or other sources, can be used to populate the generated spreadsheet.

Summary

These examples show how powerful desktop integration is for the end-user. To access the integration, they log in to Oracle Self-Service; they click on a link and are taken through application page flows ending up with a spreadsheet created dynamically on their desktop.

The user enters data with the help of list of values but more importantly using the desktop application they are familiar with. When finished, their data is validated and processed based on the application's business rules but still within the familiar desktop environment.

Now that we've shown you examples that illustrate the capabilities of desktop integration with Web ADI, let's look at its features and the benefits these features provide to end-users:

Table 1-1, Feature / Benefit analysis of Desktop Integration using Web ADI

Feature	Benefit
Easy data entry within applications such as Microsoft Excel using lists of values; pop lists and customized menu items.	User is able to enter data easily using an interface familiar to them. The lists of values are populated directly from Oracle E-Business Suite.
Bulk data entry using copy, paste and formula features in Microsoft Excel.	Large amounts of data much easier for the user to enter manipulate and analyze.
Data downloaded into desktop application from database queries.	User is presented with the latest data directly from the database. Spreadsheet based reports can be generated dynamically.
Data downloaded into desktop application from text files.	User is also able to load additional data using text files; often such files are received in email messages. Web ADI makes it easy to get data into Oracle E-Business Suite even using informal processes like email.
Data updated within desktop application is validated against Oracle E-Business Suite business rules.	Data integrity is fully preserved, regardless of whether user is entering data on-line or off-line.
Data validation errors are returned to the user in the exact place data was entered, in the case of a spreadsheet, the error is displayed in the row containing the invalid data.	Integration is much easier, especially compared to other types of desktop integration, Web ADI desktop integration provides for full integration between the desktop application and the Oracle E-Business Suite.
Fully integrated with Oracle E-Business Suite including support for Single Sign On, Flexfields and Lists of Values.	User logs on to E-Business Suite Home page same as any E-Business Suite log on, business rules enforced, only difference is they are using a familiar desktop application.
Fully supports Internet Computing Architecture.	No client software installs; only requires desktop application and web browser on PC.

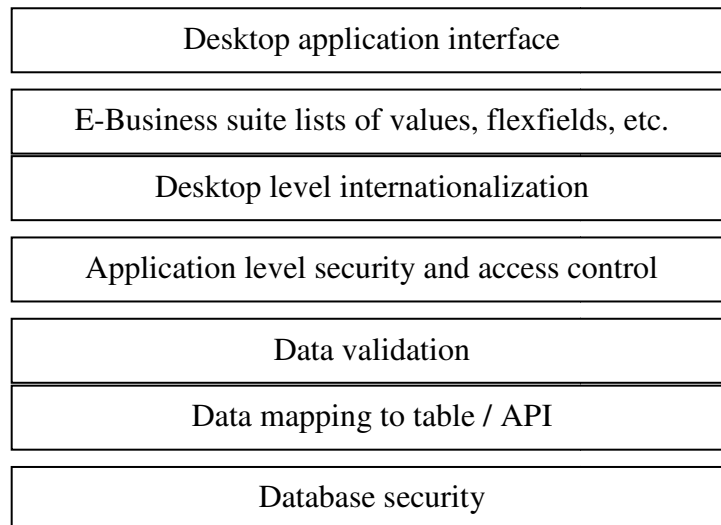
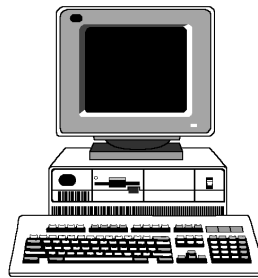
Why use Web ADI instead of building your own desktop integration

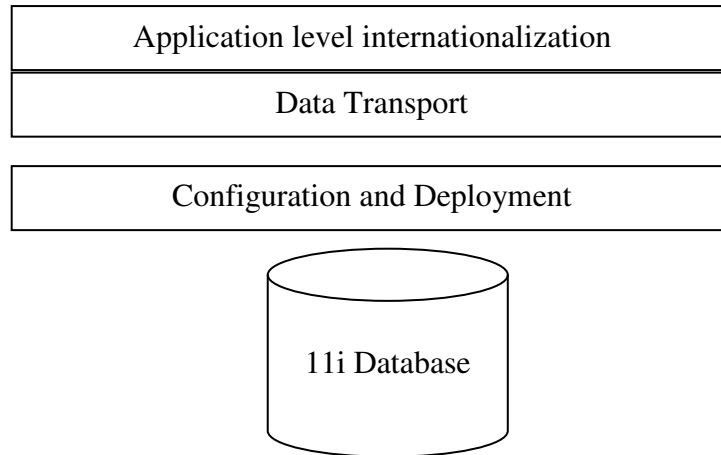
Building your own integration from scratch can seem to be quite easy. Applications like Excel simplify data retrieval and provide access to enterprise wide data. Many teams have developed integration solutions using Excel for data input and text files in comma separate value (CSV) format for data transfer. Microsoft promotes its ODBC technology for data integration.

However, building integration that incorporates E-Business Suite functionality and retains ease of use is more complex than many teams realize.

Desktop integration involves a number of components, these will vary depending on the technology and the desktop application involved, but the basic requirements are shown in the following diagram:

Desktop Computer running Excel





To illustrate the complexity involved in integration, let's consider some of the issues when building a custom integration for Excel. Each of the layers in the previous diagram raises a number of design questions and issues:

- How will the integration be invoked within E-Business Suite security?
- How will the data be transferred from the PC to the database?
- How will the data be physically loaded into the database?
- How should the integration be linked with Excel?
- Does the development team have VBA (Visual Basic for Applications) skills?
- How will spreadsheets be translated to the many languages supported by the E-Business Suite?
- How will the completed solution be configured, managed and deployed within the E-Business Suite Web Applications architecture?

These, and more, are the design issues that were encountered when developing the original Client Server version of the Application Desktop Integrator. Oracle's E-Business Suite features are increasingly sophisticated; it is a huge job to build support for them in a desktop application.

The need to resolve these issues with a framework is what led to the development of the Web ADI SDK. When using the Web ADI SDK, your desktop integration automatically benefits in the following ways:

It is **PART OF** your application.

The features of the SDK mean that your desktop integration can be seamlessly integrated into your application. The page flows within your application can call a spreadsheet, allow the user to enter data, and return to your page flow. Specific features such as flexfields or concurrent programs appear and function within Excel without the user being aware of the complexity behind-the-scenes.

It **LOOKS LIKE** your application.

Desktop integrators developed using the SDK automatically gain the look and feel of E-Business Suite applications. The Web ADI development team has worked together with the Usability and Interface Design Group to ensure all aspects of desktop integration using Web ADI comply with E-Business Suite user interface standards.

It **WORKS LIKE** your application.

The SDK allows you to specify the business rules applicable to your application. This includes full support for E-Business Suite functionality including key flexfields, descriptive flexfields, validation, security, internationalization and lists of values.

With the experience we've gained in building desktop integration solutions, we recognize the significant investment made by development teams in providing desktop application integration to their users. Using the Web ADI SDK, you can protect and leverage this investment in a number of areas:

Future proof

Web ADI continues to make significant investment in the SDK to ensure that current and future desktop applications are supported.

We currently support cross-browser deployment on either Netscape or Internet Explorer. We currently support cross-platform deployment on either Windows or Macintosh platforms. Additionally, we are working with Sun to provide support for Star Office on Windows and Linux in the future. The viewer architecture provided by the SDK was designed to cater for future changes in desktop applications and platforms.

All of these platforms are automatically supported for solutions delivered using the Web ADI SDK. As new platforms become available and are supported by Web ADI, you get support for them automatically. This also applies to changes required for new releases of Excel. We currently support Excel 97, Excel 2000, Excel 2003 and Excel XP, we take care of any compatibility issues with these products so you don't have to.

Leverage our functionality

Using the Web ADI SDK means there is no reason to re-invent the wheel when building your desktop integration solutions. Some of the features we provide include:

- E-Business Suite features
 - Lists of Values
 - Key flexfields
 - Descriptive flexfields
 - Concurrent programs
 - Security including Single Sign On
 - Data Encryption using HTTPS and SSL

- Performance management, including support for large data sets. For example, over 10,000 rows in a single sheet is supported and quite common

Web ADI continues to build additional E-Business Suite functionality into every release of the SDK. This is to cater for new E-Business Suite features and enhancement requests from existing users. Our efforts in this area ensures that you can be confident your desktop integration will be able to support new E-Business Suite features as they become available.

Resourcing appropriate staff

Using the Web ADI SDK significantly changes the resourcing requirements for building desktop integration solutions. Previously, skills in the language applicable to the desktop application, usually Microsoft Visual Basic for Applications, were required. These skills are not commonplace within Oracle development teams where the typical skillset includes SQL, PL/SQL or Java.

One of the key design goals of the SDK is to provide significantly easier integrator development especially compared to building your own integration from scratch. Previously, building integrators required the use of traditional programmers.

Now solutions can be developed using staff with business analysis skills that build the solution by configuring Web ADI Meta data without having to write any code. Removing this restriction makes desktop integration accessible to a much wider number of organizations and development teams.

Extensibility

In addition to the rich functionality of the SDK, we also recognized that some teams would require functionality above and beyond what is currently available.

We have provided extensibility in several areas. Data can be uploaded either directly to a table; a PL/SQL API or a service bean. If uploading to a PL/SQL API, all database level PL/SQL functionality, and access to ATG APIs and your own application APIs, can be leveraged. If uploading to a service bean, any middle tier Java functionality or application level Java APIs can be leveraged.

Additionally, since the core functionality of the SDK was built using the J2EE development environment, you can use the object-oriented features of Java to further extend many areas of the SDK.

Web ADI Software Development Kit (SDK)

The Web ADI Software Development Kit (SDK) enables an integrator developer to configure desktop integration solutions. These desktop integration solutions are called *Integrators*.

The SDK provides the necessary tools and documentation to configure and deploy *Integrators* within the Oracle E-Business Suite. The SDK is installed in a Self-Service Web Applications environment and is based on a multi-tier architecture as shown in the following diagram:

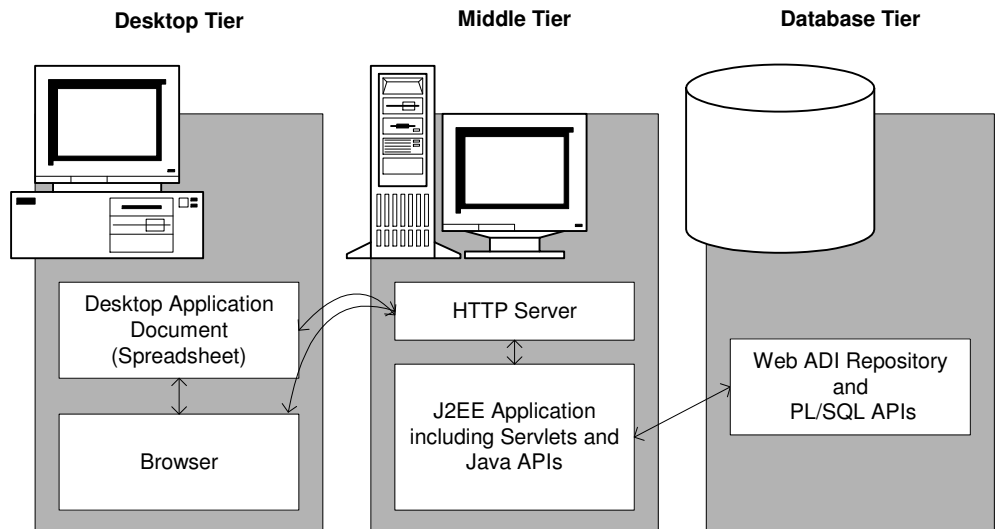


Figure 1-5, Multi-tier Web ADI Architecture

This diagram represents the Web ADI architecture in a logical form. The components shown on the middle and database tiers may physically exist on the same host computer.

The Web ADI repository is accessed by the Web ADI J2EE Java code that runs within the Self-Service Web Applications environment on the middle tier. The integrator developer configures an integrator by changing or adding Meta data to the repository.

The SDK allows an integrator developer to:

- Create a spreadsheet document on the client tier containing columns that represent interface table columns, PL/SQL API parameters or service bean properties.
- Determine what columns are included in the spreadsheet.
- Determine where the columns are placed in the spreadsheet.
- Determine how the spreadsheet is formatted (colors, fonts, cell width, etc.)
- Download data into the spreadsheet using a variety of sources including text files and database queries.
- Define Lists of Values from the database that the user will access by double clicking on the relevant columns in the spreadsheet.
- Upload information from the spreadsheet into a database table, API or service bean.

- Validate uploaded information on the middle tier before it is inserted back into the Oracle E-Business Suite.
- Automatically run standard concurrent programs to import the data into application database tables.

Using the Web ADI SDK significantly reduces the time it takes for a developer to build desktop integration solutions. Creating a desktop integration solution can be as easy as specifying the name of the integrator, the name of the database table to insert the data into and labels for the columns to be displayed in the desktop document.

An example of an integrator would be a spreadsheet allowing a user to upload data into Oracle General Ledger's GL_INTERFACE table. Such an integrator is presented to the user as a 'Journals Integrator'. The columns displayed in the spreadsheet are labeled according to the business terms the user is familiar with. The user can double click on selected cells in order to display a list of values to assist with data entry. Finally, the user is able to upload the entered data to the Oracle database.

History

Web ADI was developed based on lessons learned with the General Ledger Desktop Integrator (GLDI), Applications Desktop Integrator (ADI), Projects Connect and Applications Desktop Exchange (ADE). GLDI and ADI provide integration between Microsoft Excel and Oracle E-Business Suite, Projects Connect provides integration to Microsoft Project and ADE provides integration to Microsoft Word. All of these products were originally developed using client-server architecture.

Design Goals

Web ADI was developed with the following design goals:

- Support for Internet Computing Architecture
The client-server based integrators require configuration and installation on each PC they are used on. In large organizations, this is a significant issue. Using client-server applications through firewalls is restrictive and performance issues exist on slow, dial-up network connections. Web based architecture provides significant advantages in these areas.
- Significantly easier integrator development
Provide a Meta data repository to enable easier configuration and development of integrators. Previously each integrator was hard-coded to support specific business rules and interface tables.
With a Meta data repository in place, the most important goal of easier integrator development is possible which is the goal of enabling staff with business analysis skills to build integrators without having to write any program code.
Previously, building integrators required the use of traditional programmers. Removing this restriction makes desktop integration accessible to a much wider number of organizations and development teams.
- Support for other desktop applications
Users have requested support for other applications such as Lotus 1-2-3 and more recently Sun's Star Office. With growing support for open systems such as Linux, Web ADI saw the opportunity to build an extensible software architecture that supports a variety of desktop

applications and provide users with alternatives to using Microsoft applications on the desktop.

Extensible Software Development Kit

In addition to meeting these design and architectural goals, the SDK has been developed with extensibility in mind.

You can create simple integrators by configuring Meta data with no coding required. You can extend the functionality of simple integrators by developing pluggable components in Java or referencing existing or new PL/SQL and Java code. You can even build support for new desktop applications using the extensible viewer architecture.

You choose the degree of extensibility based on your requirements. Most requirements can be met with the configuration of Meta data. The first level of extensibility beyond this is accomplished by leveraging existing or new PL/SQL code. This is well suited to teams familiar with the more traditional Oracle technologies.

Further extensibility is available by developing Java components. These are referred to as minor SDK extensions. These components are used to extend validation; list of values and data generation functionality but within the default integration applications like Microsoft Excel and Word.

SDK extensibility also allows development teams to customize the main components of Web ADI. This is referred to as major SDK extensions. One example of a major extension is the development of custom viewers that enables support of integration to other desktop applications.

The extensibility model of the SDK ensures that integrator developers are not limited to a single programming language or a single desktop application.

Table 1-2, Feature / Benefit analysis of Web ADI Software Development Kit

Feature	Benefit
Meta data driven Web ADI repository based on Oracle database tables, partitioned by application id.	Integrators are easy to configure, only need understanding of SQL and Web ADI business rules to build complete, professional solutions. Integrators are easy to deploy. Distribute Web ADI Integrators along with seed data for your application.
Support for desktop applications with scripting capabilities.	No real limit to the applications Web ADI can be used with. Development groups have already leveraged this to provide support for Excel, Word, Project and Sun's Star Office.
Open standards architecture, HTTP, WWW, XML, HTML and Java beans.	No application code to deploy to individual client PCs. Only require desktop application and web browser on client PC. Fully functional over the internet, including through firewalls and on 56k dial up modem connections.

Feature	Benefit
Extensible APIs.	Integrator functionality can be extended using business logic developed with SQL, PL/SQL and Java programming languages. Leverage existing skills in traditional Oracle tools via PL/SQL support. Also fully supports development with J2EE using Java.
End to end integration	Web ADI integration is far more advanced than other integration solutions. Spreadsheets can be preloaded with data, the data uploaded is validated and lists of values are provided, concurrent request automatically run and errors from import program are displayed in the row where the invalid data was entered. Compare that to the functionality of other integration solutions that often use comma separated text files to transfer the data and provide limited or no error handling.
Desktop document generated dynamically by the Application Server.	Simplifies integrator deployment for development teams and users. No need to track individual spreadsheet files, users can generate new ones.
Fully integrated with Oracle E-Business Suite.	Leverage development in existing PL/SQL or Java development. Business rules already developed for Open Interface import programs can be used without modification for data validation.

Web ADI Repository

Overview

The Web ADI Repository holds the Meta data that defines a desktop integration solution. Desktop integration solutions are called *Integrators*.

The repository uses Meta data as the primary means to configure an integrator. This provides a data-driven approach to building and deploying *Integrators* within the E-Business Suite. The data-driven approach is more flexible than traditional code-driven integration solutions.

The repository is central to the Web ADI SDK and is based on a set of Oracle database tables and related objects. In addition to the definition of an integrator, the repository also contains configuration data and APIs for modifying the behavior of Web ADI.

Implementation

The repository is a collection of Oracle database objects owned by the database user BNE. The BNE schema consists of database tables, indexes and constraints, however, repository PL/SQL packages, procedures and views are owned by the APPS database user, which is standard and part of the E-Business Suite database architecture.

The repository can access other E-Business Suite database objects, data and APIs in a secured environment. Some of the items currently accessed include profile options; flexfield definitions and the security model.

The repository is a key component within the SDK and is represented by J2EE objects, also within the SDK. A corresponding Java object represents each entity in the repository. These objects provide an abstraction layer over the repository and can be referenced by integrator developers if required.

A collection of PL/SQL APIs is maintained within the repository. These APIs are provided for configuring the data within the repository. An integrator developer is able to define the Meta data for an integrator by using one or more PL/SQL procedures.

The repository is partitioned by application id. Application id is common to all E-Business Suite applications and defines a unique identifier within the database for an application. This partitioning was done to simplify the deployment of integrators and provide separation of data across integrators developed by other teams.

A single set of repository tables exists within each E-Business Suite instance. The partitioning by application id means that each development group can release their own Meta data into the repository independent of any other team and independent of Web ADI. Using the standard patch mechanisms, any team can release a new or updated integrator as required.

Repository records are uniquely identified by a combination of Application ID and a unique, non-translated text code. Code is a unique non-translated text value. The value for code is determined by the integrator developer and must conform to a format similar to the Oracle table naming conventions. (Upper Case with a format of 'A...Z', '0..9', '_', no spaces and no special characters.)

Translation tables and corresponding views exist for all tables requiring translation. All translatable character data, stored in tables, is translated using a separate translation table for each base table.

For example, a table named BNE_INTEGRATORS_B has a separate table named BNE_INTEGRATORS_TL that contains all translatable columns. The BNE_INTEGRATORS_TL table has a primary key composed of the original primary key from BNE_INTEGRATORS_B and an additional LANGUAGE column. This table will therefore contain a row for each base table row for each language the data is translated into. A view named BNE_INTEGRATORS_VL joins these two tables based on the primary key and the current language (USERENV('LANG')). This view is referred to as a *virtual translated table* for this entity.

Key Repository Logical Entities

- Integrators

A desktop integration solution is defined as an *Integrator*. Includes attributes for the application id, integrator code and description.

- Interfaces

The interface is a database table, PL/SQL API or service bean that you will be updating with the integrator. Any number of interfaces can be defined for an integrator; typically it is a small number, usually only one.

- Interface Columns

Contains data type, validation and Meta data about each column in an interface. The columns are either table columns or API parameters. Includes attributes that define column validation and lists of values.

- Layouts

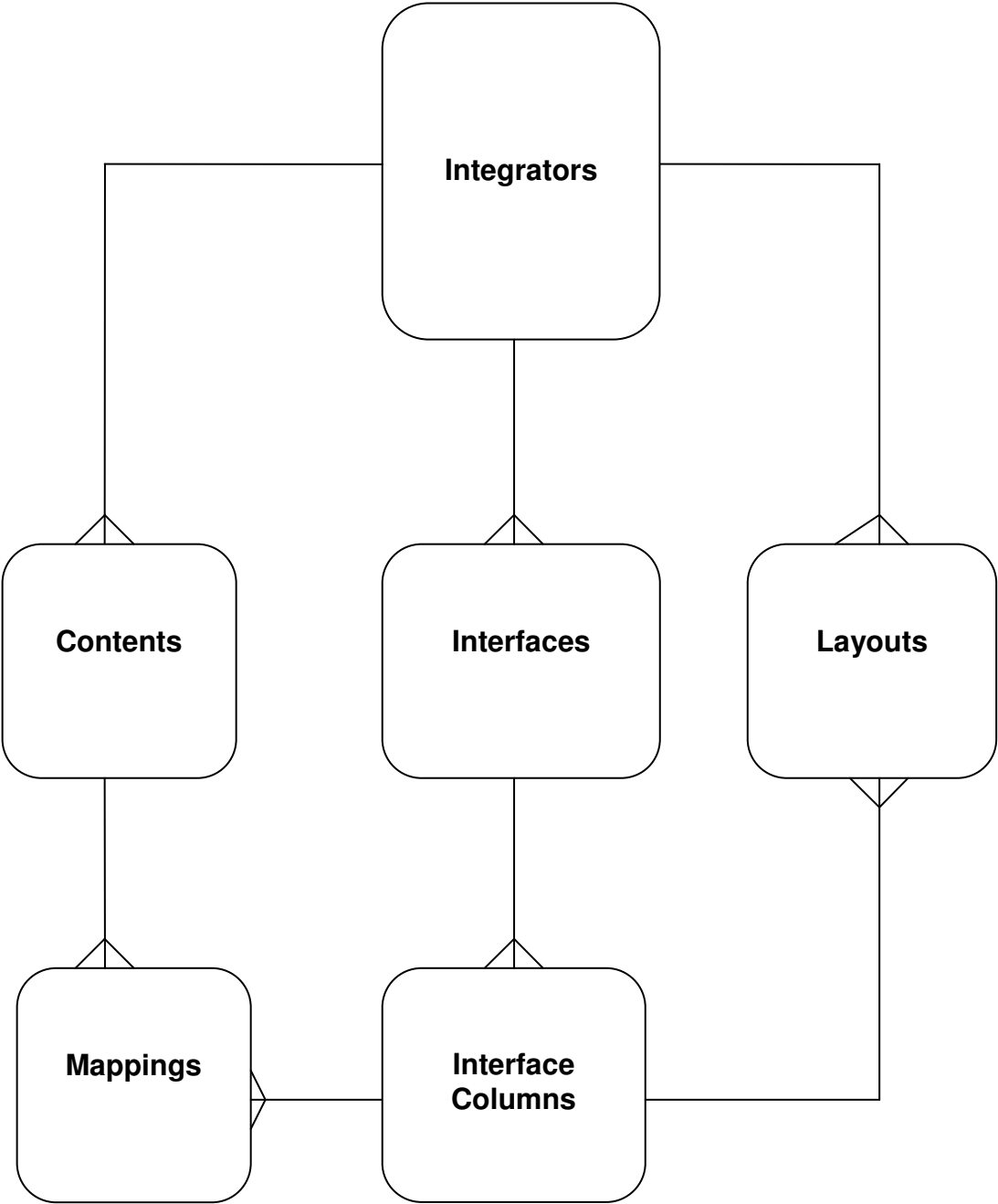
The layout is a definition of how columns are visually represented in the desktop application document. If the document is a spreadsheet, the layout defines which areas of the spreadsheet are used for specific interface columns.

- Contents and Mappings

Contents contain details about the source of data to be downloaded into the desktop application document. This allows the document to be pre-loaded with data from database tables; text files or other data sources.

Mappings define the relationship between the content and the interface columns defined in the layout. For example, a mapping would define how values from each column of a comma separated text file are placed into specific columns within a spreadsheet.

**Repository ERD Diagram
(Logical Model)**



How Web ADI Works

This section provides an explanation of how the Web ADI SDK uses its service-oriented architecture to provide desktop integration. A number of terms specific to Web ADI are introduced and explained.

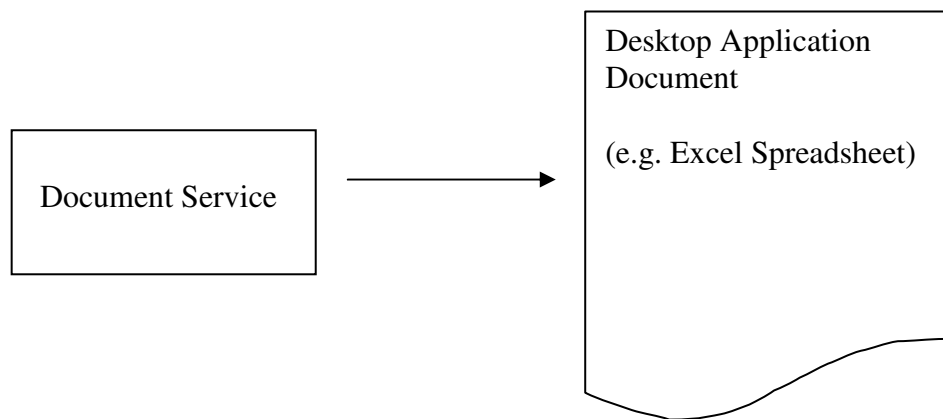
Included in this section are examples that show how Web ADI uses the document service to dynamically generate a spreadsheet and then uses the download service, component service and upload service to provide Web ADI functionality to the user within the spreadsheet environment.

Service Oriented Architecture

The Web ADI SDK includes APIs that are accessible in a variety of ways. At the highest level, these APIs are referred to as services. The desktop application document (spreadsheet, word document, project plan) is central to the end-users interaction with Web ADI and the Oracle E-Business Suite. The spreadsheet itself interacts with Web ADI via a number of services.

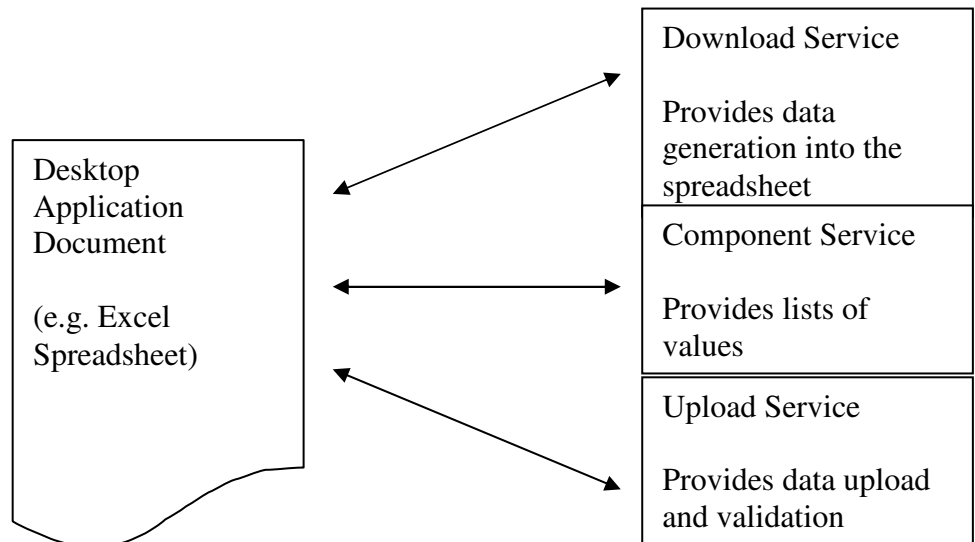
The Web ADI services are available to both the integrator developer and the end-user. The integrator developer configures the service using the Meta data in the Web ADI repository or calls the service directly. The end-user indirectly calls the service and will not be aware they are using the service; it is behind-the-scenes and transparent to them.

The desktop application document is generated dynamically by the document service. By dynamically generating this document, we are able to automatically provide the document in the local language for the specific user and in the version of the desktop application they are using. For example, while one user is generating a spreadsheet in Korean for Excel 2000, another user can be generating a spreadsheet in French for Excel XP.



The interaction between the document service and the spreadsheet is a one-way interaction, once the spreadsheet is generated, the code in the spreadsheet then interacts with the other Web ADI services and does not interact with document service any more.

The interaction with the other services provides the user with automatic data generation into their document (download service), lists of values (component service) and the ability to upload the data within the spreadsheet into the application server (upload service).



The interaction between the download, component and upload service is two-way, each service can be called numerous times and each service returns data in the case of the download and component service or messages in the case of the upload service to the spreadsheet.

When the integrator developer uses the services, a number of parameters are used to control or customize each service. For example, you can pass a parameter to the document service that indicates which desktop application the document is being generated for. In some cases you will want to generate a Microsoft Excel document, in other cases you will want to generate a document for Microsoft Word. Either way, it is controlled by a specific, named parameter for the document service.

The Web ADI services are available to the end-user through a variety of user interfaces. These include Self-Service page flows, Lists of Values within the spreadsheet and menu options within the spreadsheet.

The rest of this section provides an example of how these services operate. When working through this example, keep in mind that all the services are controlled by a set of parameters specific to each service. The service parameters are discussed regularly in this document and are referred to in the context of the corresponding service, for example, “upload parameters” or “download parameters”.

Document Generation

The document service is responsible for dynamically generating documents for the target desktop application. The Self-Service page flows for Web ADI include a *Create Document* page flow that prompts the end-user for each of the parameters required by the document service. The end-user will not be aware of the document service or the concept of parameters, to the end-user the page flow is a series of questions or lists of values that must be answered in order to generate the desired document.

The *Create Document* page flow can also be completely hidden from the end-user by passing values for each of the parameters to the document service from your application. This allows you to completely embed Web ADI within your application.

When the *Create Document* page flow is complete, it calls the Web ADI document service in order to generate the document for the end-user. Depending on the selections made during the *Create Document* page flow, the generated sheet may then automatically call the download service to populate the spreadsheet with data. This data can come from a number of sources including SQL queries.

Document Service

The document service uses a list of named parameters to control the generation of the document. The most important of these parameters are listed in the following table:

Table 1-3, Main Document Service Parameters

Parameter	Description
bne:language	The language used to create the document, typically determined by E-Business Suite profile option values.
bne:integrator	Reference to the primary key of the integrator as defined in the Web ADI repository.
bne:layout	Reference to the layout of columns as defined in the repository.
bne:content	Reference to the type of content, if any, to be downloaded into the document.
bne:viewer	The desktop application to be used to access the generated document, typically, but not limited to, Microsoft Excel.
bne:reporting	Indicates if the document is generated in read-only mode. A read-only, or reporting document, cannot be uploaded.
bne:map	Reference to a predefined mapping that links the layout columns to the downloaded content.

Create Document Page Flow

The *Create Document* process begins with the end-user selecting the *Create Document* function from the E-Business Suite Home Page. This function is available from the *Desktop Integration* responsibility. *Create Document* is a generated HTML page flow that allows the users to enter the parameters that are needed to generate the document. Including parameter values in the form function definition may circumvent part or all of this page flow.

The following pages provide examples of each of the pages in a basic *Create Document* page flow. Take note as you review these examples of the tracks displayed towards the top of each page. These tracks help the user keep the overall context of where the particular page currently displayed fits into the overall page flow:



Figure 1-5, Create Document tracks (TrainTracks.gif)

Viewer Page

This page displays a list of values for the viewer selection. The viewer is the desktop application used to access the document. The available viewers are Excel 97, 2000, XP, 2003 and Word 97, 2000, XP and 2003. The word viewers are typically used to perform mail-merge functionality.

Figure 1-6, Create Document, Select Viewer Page (SelectViewer.gif)

Integrator Page

This page displays a list of values for the integrator selection.

The Web ADI repository is queried to retrieve the description of the integrators. The Meta data defined for the integrator determines which interface table or API the spreadsheet will update.

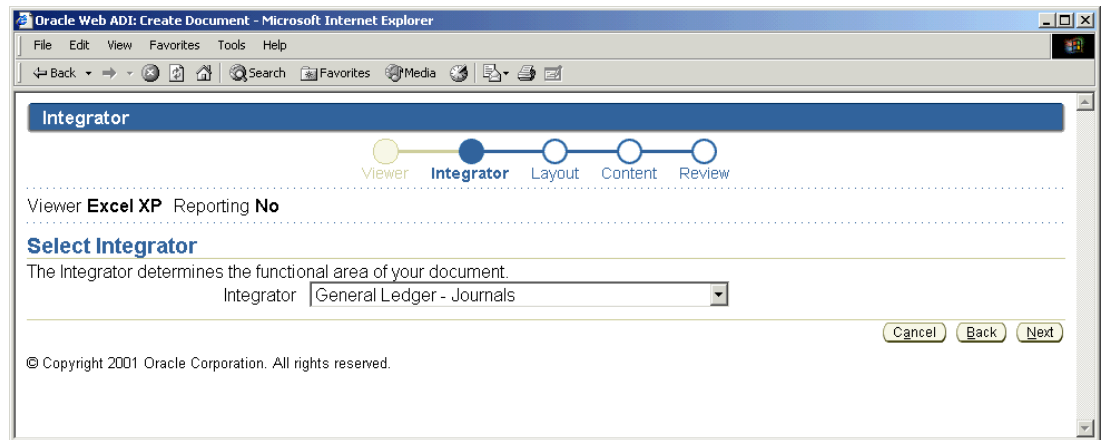


Figure 1-7, Create Document, Select Integrator Page (SelectIntegrator.gif)

Layout Page

This page displays a list of values for the layout selection. The layouts shown will depend on the integrator selected on the previous page. There is no limit to the number of layouts that can be created for an integrator. The layout defines the columns that will be included in the spreadsheet.

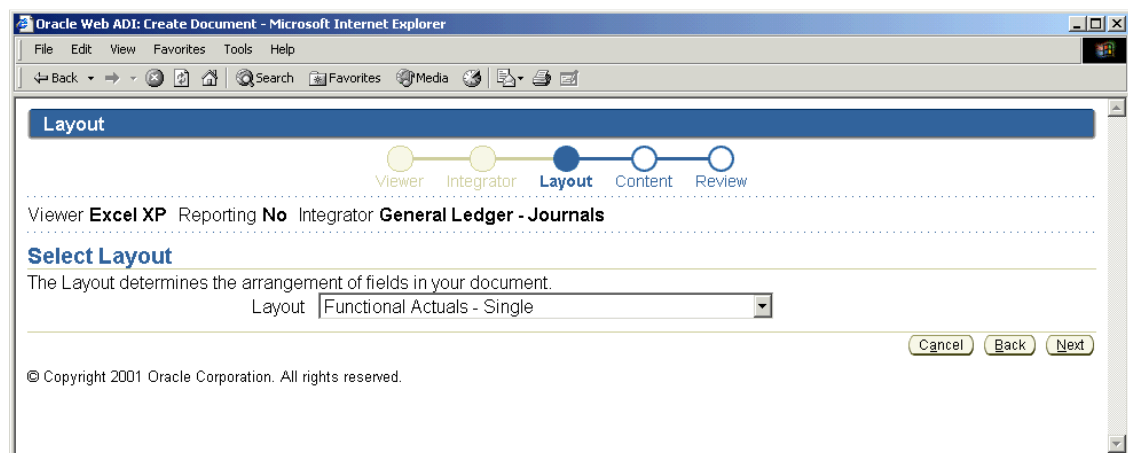


Figure 1-8, Create Document, Select Layout Page (SelectLayout.gif)

Content Page

This page displays a list of values for the type of content to be downloaded into the spreadsheet. This allows the spreadsheet to be pre-loaded with data before the end-user accesses the sheet. This is typically used for solutions where the desktop application is being used to update or analyze existing data. Selecting content is optional, in this example the content type of *None* is selected.

When a specific content is selected, a *Mapping Page* is shown where the user selects the source of the content and a mapping that links the content to the layout. The *Mapping Page* is not shown in this example.

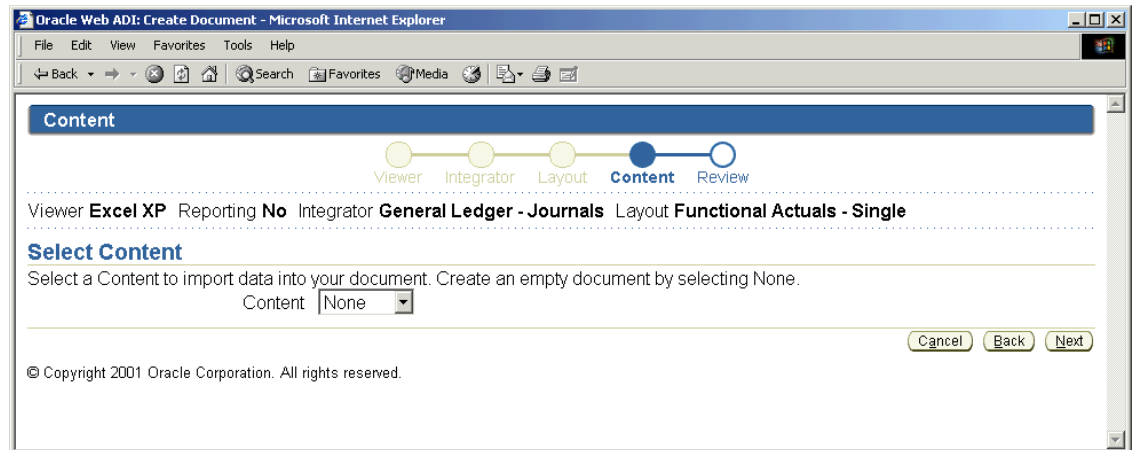


Figure 1-9, Create Document, Select Content Page (SelectContent.gif)

The *Select Content* page is establishing download parameters that will be used by the download service after the spreadsheet is generated. Once the document service generates the spreadsheet, the download service will be automatically called to generate data into the document.

Review Page

The final page in the page flow displays the parameter values selected in the previous pages. The end-user can proceed with document creation, go back and change any of the values or save the selections as a shortcut. Shortcuts are useful because they can be referenced directly in a Self-Service function that allows the user to create a document without going through the entire page flow.

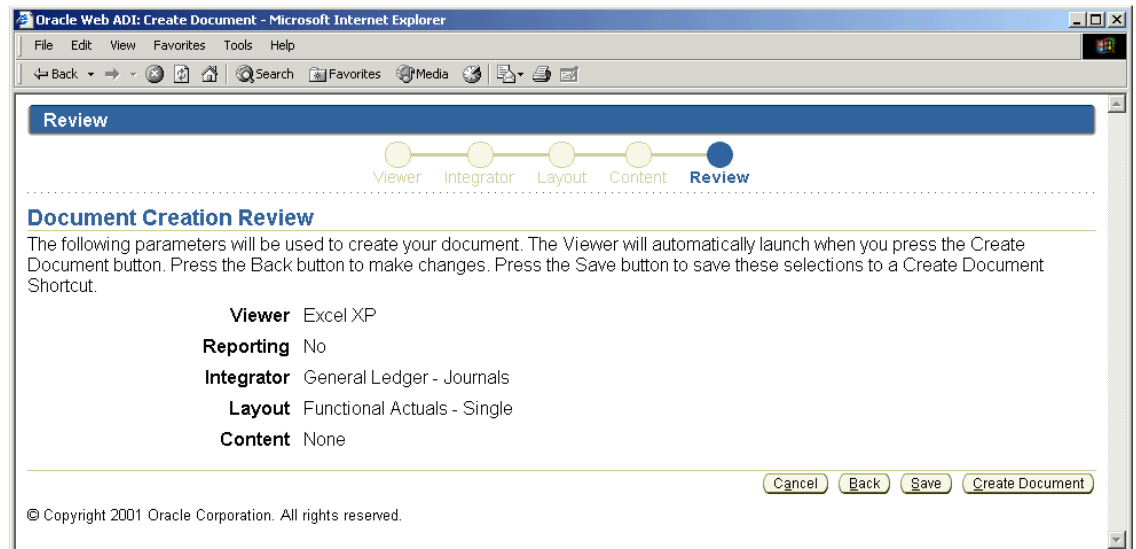


Figure 1-10, Create Document, Review Page (CreateDocumentReview.gif)

At this stage, the *Create Document* page flow has completed the process of obtaining the parameter values for the document service. The *Create Document* page flow now calls the document service and passes these parameter values to the document service. The user is unaware of this behind-the-scenes operation.

Document Service Pages

The pages displayed previously were part of the *Create Document* page flow. When the *Review Page* is completed, the *Create Document* page flow then calls the *Document Service*.

The end-user is briefly shown the next two pages; they may notice the *Document Service* term as displayed in the page title.

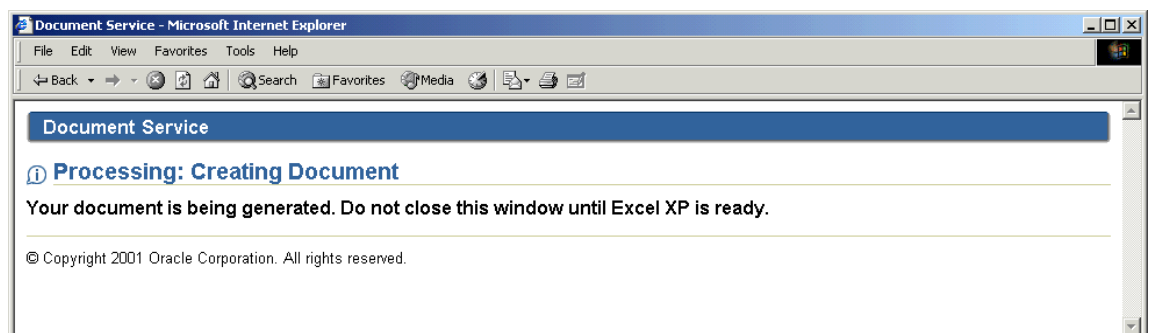


Figure 1-11, Document Service, Creating Document Page (CreatingDocument.gif)

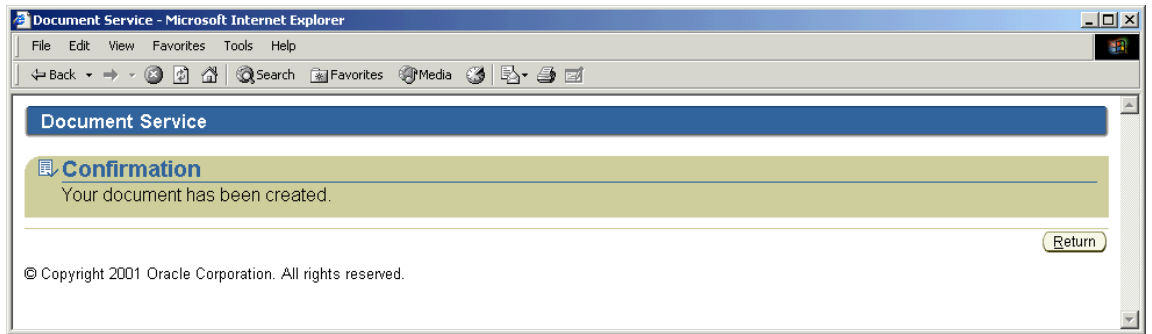


Figure 1-12, Document Service, Confirmation Page (DocumentCreated.gif)

A new window will be opened on the users PC at this stage and the browser's display will return to the E-Business Suite Home Page. The new window will contain the generated document opened in the appropriate desktop application.

Generated Document

The generated document is shown in the new window on the users PC. The following example shows a *Functional Actuals - Single* layout from the *General Ledger - Journals* integrator.

The layout consists of three sections, the context section that includes the Balance Type, Database and Set of Books columns, the header section that includes the Category, Source, Currency and Accounting Date columns and the lines section that includes the Upl, Co, Dpt, Acct, Sub, Prd, Debit, Credit and Messages columns.

The layout is structured in a visual hierarchy, the columns in the context section are read-only, and the columns in the header and lines section are expected to be entered with values entered by the user or copied from other documents. The header and lines section is quite similar to the concept of a master-detail form as is commonly used in Oracle Developer/Forms.

When the data entered in the spreadsheet is uploaded, the values in the context and header sections of the layout are duplicated for each row in the lines section of the layout. Effectively the visual hierarchy is flattened. This use of a hierarchy is done to optimise data-entry. Values that are common for all rows of the interface need only be entered a single time by the end user.

As described earlier, once the spreadsheet is generated, the interaction with document service is now over. Now, the spreadsheet will interact with either the download service, the component service or the upload service.

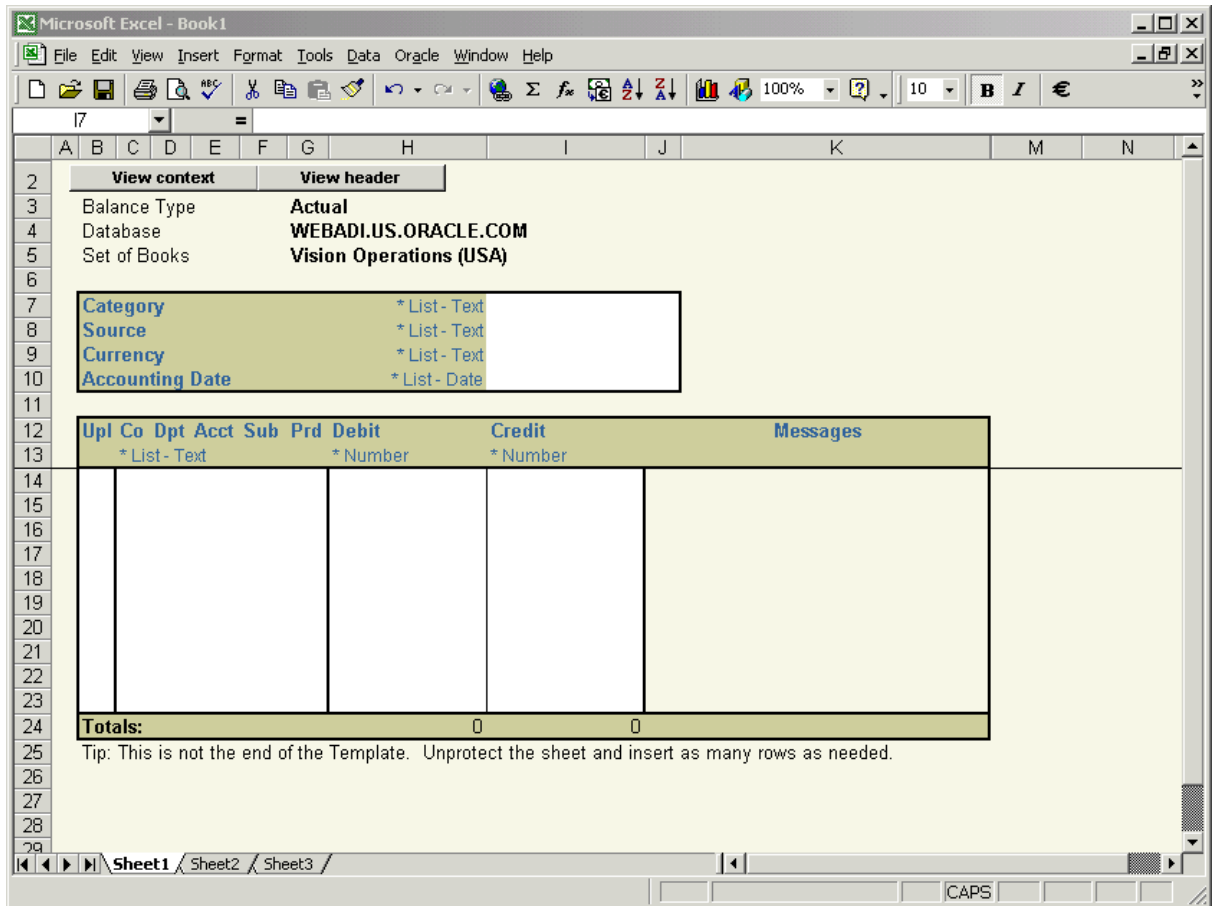


Figure 1-13, Document Service, Generated Document (GeneratedDocument.gif)

The combination of the Co, Dpt, Acct, Sub and Prd columns is an Accounting Flexfield. The messages column is where upload validation errors, if any, are shown. Columns with the *List* label have a list of values available to the end-user. The list of values is accessible by double-clicking on the related cell as shown in the following examples:

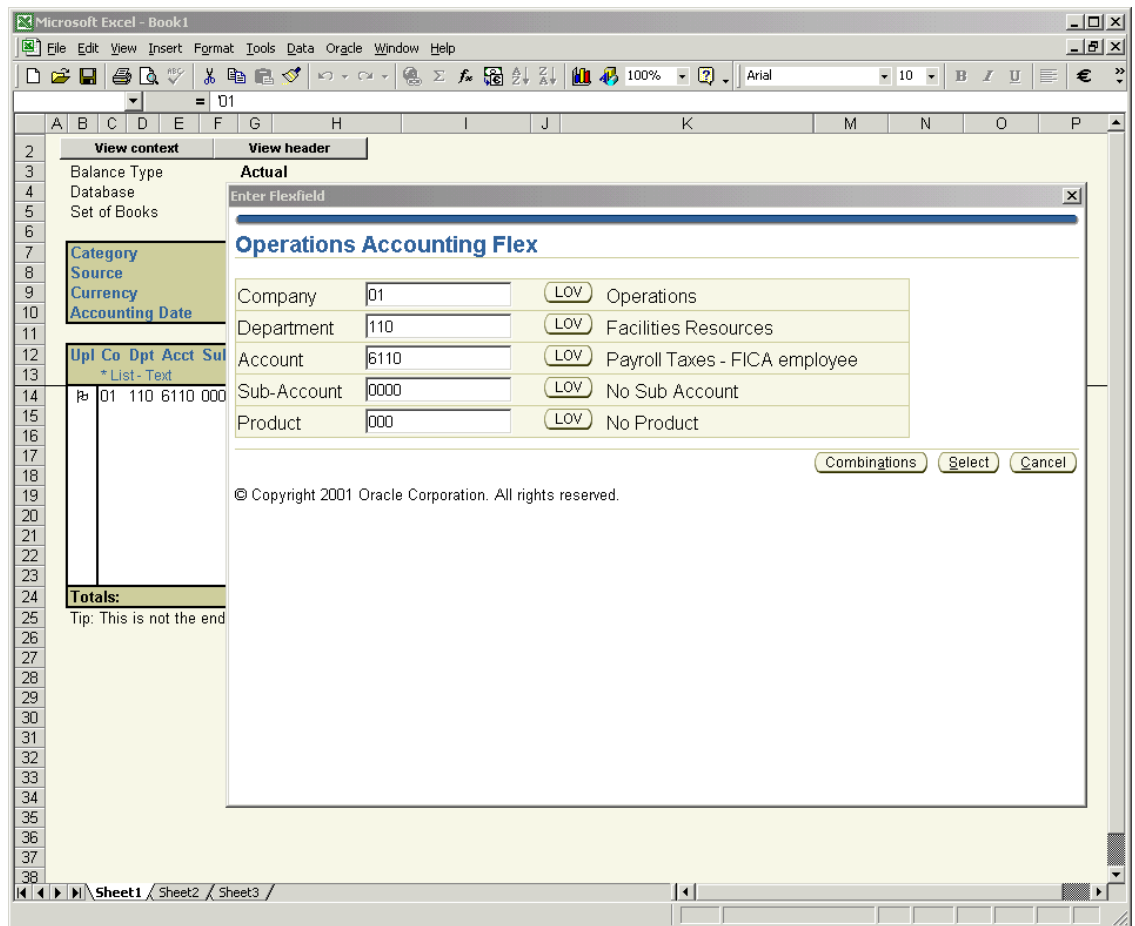


Figure 1-14, Accounting Flexfield list of values (AccountingFlexLOV.gif)

The list of values is created by the component service within the SDK. The integrator developer is able to configure using Meta data a number of parameters for the component service. These parameters are used to control the operation and presentation of the list of values. As you can see from the prompts provided in this list of values, Web ADI provides tight integration with the E-Business Suite.

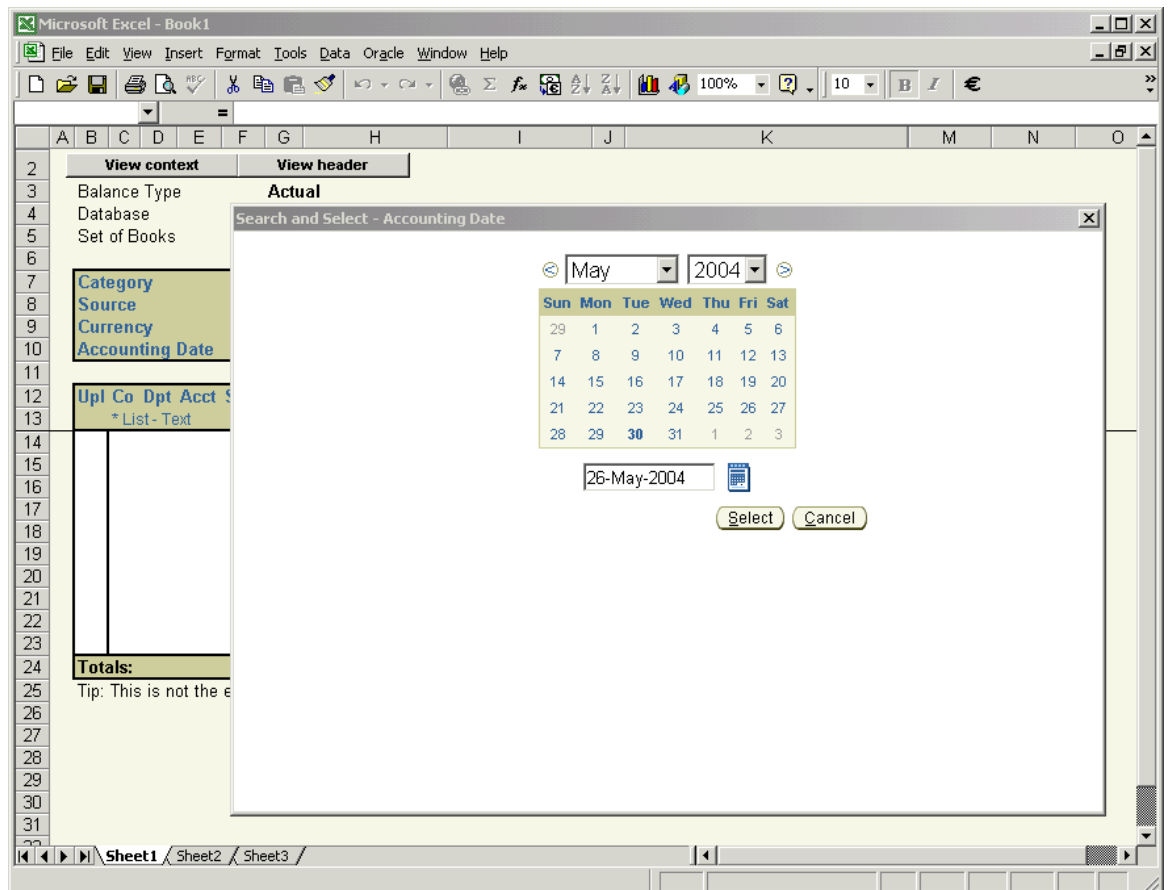


Figure 1-15, Accounting Date list of values (AccountingDateLOV.gif)

This shows an example of the Accounting Date list of values. When the user double clicks on the Accounting Date cell, Web ADI makes a call to the component service, which displays the list of values. This example is quite similar to the Accounting Flexfield list of values in that it is a call by Web ADI to the component service. For the integrator developer, parameters are used to control the operation and presentation of the list of values. One example of the types of behavior that can be customized is that the value selected by the user may be placed into more than one cell in the sheet.

Once the user has completed any data entry within the spreadsheet, the values may then be uploaded to the E-Business Suite by using the upload service. The user selects the *Upload* option from the *Oracle* menu that is available from the Excel menus at the top of the Excel window. When the user selects *Upload*, the following *Upload Parameters* page is displayed:

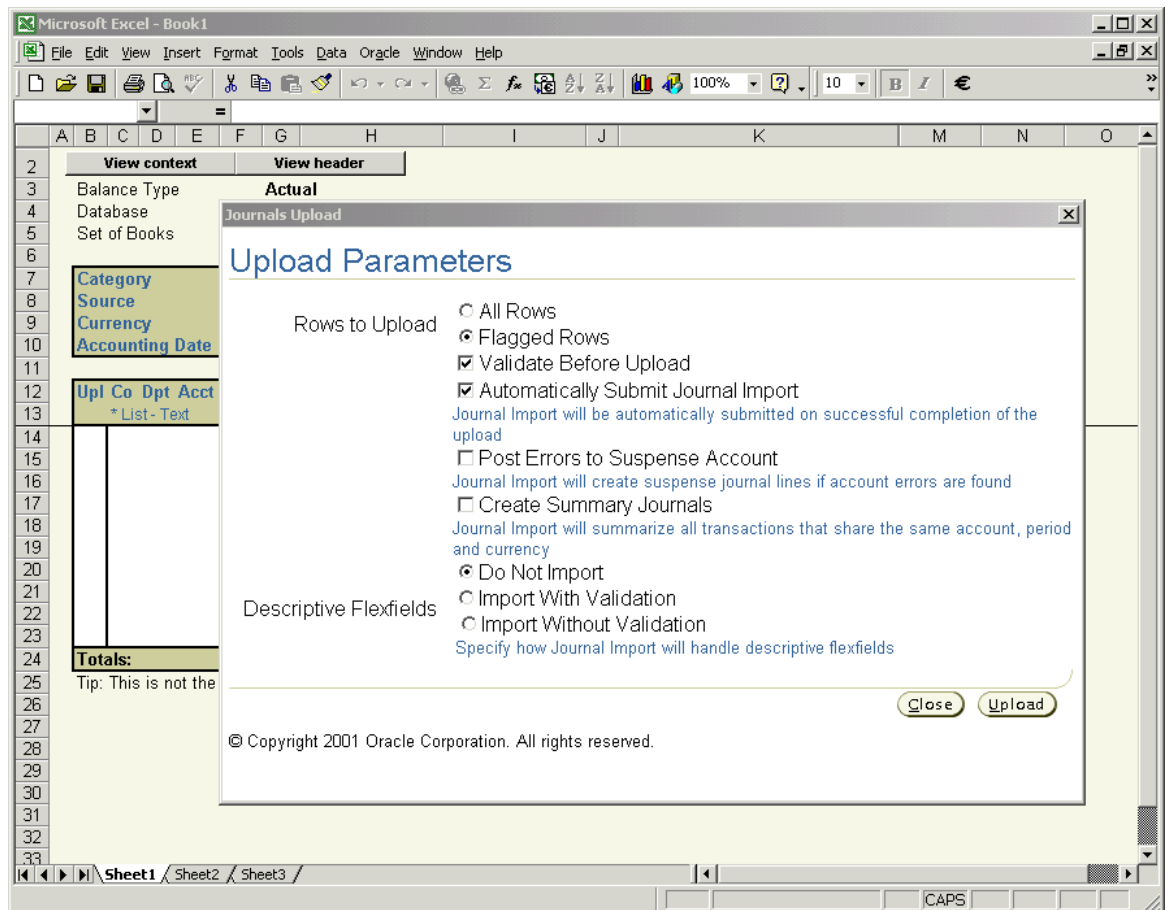


Figure 1-16, Upload Parameters page (UploadParameters.gif)

The *Upload Parameters* page provides prompts to the user according to the business rules of the integrator. In this example you will notice a number of prompts that are specific to Oracle General Ledger; Automatically Submit Journal Import, Post Errors to Suspense Account and Create Summary Journals. This is another example of the tight integration with the E-Business Suite that is available through Web ADI.

Each of these prompts provides values for parameters to the upload service. Some of these parameters are specific to Web ADI; some are specific to the integrator. Any or all of these parameters can be hidden from the user by establishing default or derived values.

Summary

This completes the introduction to the Web ADI SDK. Desktop integration using the SDK provides exciting new capabilities for integration within the Oracle E-Business Suite. The data-driven approach of the Web ADI repository means that professional quality solutions can be built without writing code; all that is required is the definition of Meta data.