This presentation gives an overview to the exciting new features of Oracle9i Reports. The key themes of the release are:

**Any** Data
- Allow developers to build reports on ALL data – wherever it is

**Any** Format
- Allow developers to produce content of the highest fidelity – and also be flexible

**Anywhere**
- Allow developers to securely distribute the content wherever it's needed in a performant manner.
Oracle9i Reports has 2 components – the development environment, and the deployment environment.

The development environment is a part of Oracle9i Developer Suite and is installable from the ‘Business Intelligence (BI)’, ‘Rapid Application Development (RAD)’, or ‘Complete’ install options. The deployment environment is a part of Oracle9i Application Server (Oracle9iAS) and is installable from the ‘Business Intelligence (BI)’ or ‘Complete’ install options.

We’ll take a look at the new features of Oracle9i Reports in the context of development and deployment.
Oracle9i Reports New Features

Development
We will look at each of these in the following slides.
Oracle9i Reports has always been a powerful tool to publish data in a variety of formats. Oracle Reports 9i combines the strengths of scaleable reporting architecture with the additional advantages of JSP (1.1) custom tag technology to deliver high quality HTML pages for Web publishing – whilst at the same time retaining it’s strengths in Paper publishing. A Web developer can use his favorite Web authoring tool to design a Web page. The dynamic data that has to be presented in this page can then be “injected” in through the use of JSP tags into the appropriate sections of the page. Alternatively, designers may use templates that are supplied by Oracle Reports to build the page. The Reports metadata can be stored within the JSP source or even in a traditional RDF and just referenced with a single tag. With this approach, Oracle Reports does not own the entire page, and hence it becomes easier to incorporate HTML content from other JSPs, or any other HTML provider.

Oracle9i Reports Developer has been enhanced to permit the loading and saving of Report definitions in XML or JSP (which embeds XML). An XML-based definition of the report means developers now have an api to report definition – either for reading (eg for automated impact analysis) or even for Report creation.

So now developer’s have a report writing tool that understand the requirements and differences between web and paper publishing, whilst permitting business logic and data-modeling to be shared between the formats. Note: a JSP doesn’t HAVE to be HTML – could be WML for example. Microsoft Excel 2000 (or later) also supports an HTML/XML definition of a spreadsheet – thus it is relatively easy to create a template with formatting in Excel 2000, save as HTML/XML, and then use Reports JSP tags to add content from various data sources.
What about RDF’s?

RDF’s are report definition files from previous releases of Oracle Reports. These are not going away – so jsp, xml and rdf are all equivalent as report definition files.
**JavaServer Pages (JSP’s)**

- Separate out dynamic and static parts of a Web page
- Compiled
- Standards-based (J2EE)
- Extensions using JSP Custom Tag library
- Reference paper layouts

```html
<HTML>
<%@ taglib uri="reports.tld" prefix="rw" %>

<B>Expression.</B><BR>
Your hostname:<BR>
<%=request.getRemoteHost()%>.  

<B>Declaration (plus expression). </B><BR>
<%! private int accessCount = 0; %>
Accesses to page since server reboot:<BR>
<%= ++accessCount %><BR>

Employee Name:<BR>
<rw:field id="abc" src="ENAME"/>
```

---

**What is a JSP?**

JavaServer Pages (JSP) lets you separate the dynamic part of your pages from the static HTML. You simply write the regular HTML in the normal manner, using whatever Web-page-building tools you normally use. You then enclose the Java code for the dynamic parts in special tags, most of which start with "<%" and end with ">%".

**Why use JSP?**

They are easy to use, fast in performance with just-in-time compilation, and are rapidly becoming the defacto standard for dynamic enterprise web applications. As previously mentioned, they do not have to produce HTML – you can use any markup language with the Reports JSP tags.

**How do they work?**

When a JSP is executed by a URL requesting it, it is executed under the JSP engine of the web listener (ie mod_OC4J). The JSP processing occurs in two phases:

- **Translation phase:** The JSP engine included with Oracle9iAS determines if this is the first request for the JSP, or if the source file has changed since it was last requested. If so, the OC4J engine translates the JSP file into a pure Java source file containing an instance of the `javax.servlet.Servlet` class. This intermediate source file is then compiled to create an executable Java class file.

- **Request processing phase:** The servlet engine included with Oracle9iAS determines if the JSP class file has not yet been loaded or if it has changed since last executed. If either is the case, the servlet engine will load or reload the class file for execution. Once loaded, the class is immediately executed.

This design minimizes latency in responding to requests, as after the first request is processed, the code is available in memory and ready for immediate execution on any subsequent requests.
New Query Types

Information that needs to be published is often derived from data in corporate data sources. These data sources may be SQL based (relational databases) or non-SQL based, such as XML data that has been generated from outside the corporation, OLAP databases, JDBC data sources, or metadata that may be available as part of another system. It is often relevant and necessary to combine data from one or more of these data sources in order to derive meaningful information. For example, in a business intelligence scenario, it may be highly desirable to be able to combine and present together data that exists in a relational database with data in a multi-dimensional database in order to compare trends, performance, and so forth. Equally important is the need to join data from different data sources based on a valid condition. New query types have been added to Reports – XML (XML schema, DTD’s and xsql), Text Files, JDBC, and Oracle9i Olap. In addition, we’ve moved to a Pluggable Data Source architecture (and it’s in this architecture these new data sources have been developed).

Pluggable Data Sources

A Pluggable Data Source (PDS) is a set of design-time and runtime Java api’s that provide openness to Reports by enabling data input from numerous sources through the implementation of the PDS Java Interface. The PDS feature enables developers to leverage Reports’ aggregation, summarization, formatting, and scheduling capabilities not only on data that is accessed through SQL, but on data that is available elsewhere. Access to these data sources can be simplified since the PDS architecture incorporates connectivity to Oracle9iAS Single Sign on.
Java Importer

- Interface between Reports and Java
- PL/SQL wrappers generated for Middle Tier Java
- Use Java as an Integration layer:
  - Web Services / WSDL
  - SOAP
  - CORBA
  - XML

These features are key to the Reports integration strategy, giving you simple to use Java integration from PL/SQL. The Java Importer can be used with 1.3 of the Java Runtime, giving you the ability to integrate your Reports applications with the latest external programs and services.

You can use the power of the Java Importer to integrate Reports with Web Services/WSDL, SOAP, CORBA, XML and it provides an interface for Arrays, Exceptions, Objects, Methods, Attributes, Static, Methods / Attributes, Object lifetime & Persistence.
Java developers may wish to leverage the powerful publishing capabilities of Oracle9i Reports within their Java applications. To facilitate this, Oracle9i Reports has been integrated within JDeveloper 9i in several ways:

• Developers may create a new Reports JSP or Reports Pluggable Destination from within Oracle9i JDeveloper (they appear as objects in the object gallery and have a dialog to assist in their creation.)

• Reports can be coded by hand, or the Reports9i development environment (the Reports builder) can be launched directly from within Oracle9i Jdeveloper. The report can then be declaratively created as before, then developers may exit back straight into JDeveloper, and automatically have the JSP they created brought with them.

• Developers may run and debug Reports JSP’s from within Oracle9i JDeveloper – setting breakpoints, etc.

• Reports’ JSP custom tags appear within Oracle9i Developer’s JSP tag palette to facilitate easy report development/editing
Oracle9i Software Configuration Manager Integration

Oracle9i Software Configuration Manager is integrated with the source control options in the Report Builder to provide Check-in/Check-out Capability out-of-the-box.
Graphing

- 52 different graphs and styles available
- Data-aware hyperlinks

The BI Graph bean is now available within Oracle9i Reports – offering 52 different types of graph, and with the same features previously available such as graph hyperlinks for in-context drilling from chart.

Note that these charting capabilities replace the Oracle Graphics charting engine from Reports 6i. Reports using Oracle Graphics will still continue to execute when moved to Oracle9i Reports as long as Oracle Graphics 6i is installed on the same machine in a different Oracle_Home. However no new Oracle Graphics charts may be created with the chart wizard in Reports9i.
Oracle9iAS Portal allows you to build quick and simple HTML reports. However developers may find they need a PDF version of that report, or want to enhance it (e.g., add more queries, make the layout more complex, etc). Oracle9i Reports Developer makes this easy—developers can simply pick a report from the list of reports in Portal and import it. The report is then converted on-the-fly to an Oracle Report, and now developers have the wealth of features and wizards in Oracle9i Reports at their disposal. With the Oracle9iAS Portal integration, the report may then be deployed back within the Portal.
Oracle9iAS Discoverer Plus and Viewer are now able to export discoverer worksheets to XML that can be opened in Oracle9i Reports Developer. This export contains the query plus all the formatting information necessary to create a paper report (including conditional formatting). Developers can take this report and now do everything they can with an Oracle Report - produce PDF, or enhance it (eg add more queries, make the layout more complex, etc).
New Features - Deployment

- Servlet and JSP Runtime Architecture
- Open API’s – Pluggable Facilities
- Oracle9iAS Reports Services Clustering
- Reports as Portlets
- PDF 1.4
- Enhanced Email support
- Event-driven publishing
- Enhanced Report Bursting
- OEM Integration
- SSO Integration

We will look at each of these in the following slides.
New Architecture

The new Reports server uses multi-threaded Java-based servlets. A servlet is a Java application that runs in a Web server and provides server side processing. The Reports CGI is replaced by a Java servlet in release 9i (though the CGI is available for backward compatibility). Servlets are fast - with a CGI implementation, each time a user initiates a request the CGI is loaded by the Web server and a CORBA connection has to be established with the Reports Server every time. A servlet persists between invocations and thus establishes the CORBA connection only once. This gives the servlet tremendous performance benefits over the CGI.

How does it work?

When a request is received for the servlet to run a job, it communicates with the reports server. After checking security with SSO server and obtaining the appropriate credentials, the server spawns an engine of the appropriate type to perform the task. In the case of a report, the reports engine spawns interprets the XML for the paper layout (if running a JSP) or executes the RDF directly to produce PDF, RTF, Postscript, PCL, etc. The engine also connects to the various different data source(s) to fetch the data and can get this authentication information from the SSO server. After the output is created, the Server securely caches and distributes it, and provides notification(s) of the success/failure of the job. Only authenticated users may see the cached output, and if the same request comes in for the same content but to a different destination, the cached content can be used. The Server also has a persistent cache – so it will ‘remember’ content that has been cached between server restarts.

URL Engine

Supplied along with the reports server and the report engine is another type of engine – the URL engine. This is executed in a similar way to the reports engine, but takes one argument – a URL, and will fetch the content returned from this URL and then distribute it to any destination the server supports. So in this way reports Jsp's as well as any web content may be emailed, printed, pushed into Oracle Portal, etc.

In-Process Server

Traditionally, the CGI client and the Reports Server have been two separate entities and all communication between them was over SQL*Net and across processes. With in-process server, the client (a servlet in 9i) and the reports server can be in the same process i.e. with normal Java function calls and thus the overhead of CORBA invocation has been eliminated. Thus in-process communication is much faster. When you configure the servlet to use the in-process server, the servlet will start and stop the server automatically, using that server to run its jobs.
As discussed, the JSP programming model allows Web content to be generated dynamically during program execution through Java scriptlets, declarations, and expressions interleaved with the static content in a Web page. The static content in the page can be any markup language, including HTML, XML, and WML. JSP’s are also executed by the JSP engine of the Web listener (and not by Reports).

When the JSP engine is executing the JSP, the Reports custom JSP tags invoke the the Reports Server to make sure the user is authenticated to run the JSP. Once that succeeds, the reports engine is then spawned to fetch data from the various data source(s) and perform any calculations/aggregations. Optionally, if there is also a paper layout in the JSP, then the reports engine can also format the pieces of the paper layout. All the data, and any referenced paper layout components are then returned to the JSP engine to assemble the content (which may contain content from other jsp tag libraries also), and then the content is returned as the response to the URL.

Since JSP’s may contain calls to other JSP tag libraries etc., they must be executed in the web listener and jsp engine, and not in the reports server or engine. This essentially means that it is always an HTTP request that is being sent and this will always return the content to the issuer of the HTTP request. However if the content needs to be distributed to another destination (ie not a browser) then the Reports URL engine can be used – allowing developers to produce high-fidelity content for other destinations (eg Email).
Out-of-the-box

For each of these facilities, we provide an 'out-of-the-box' implementation(s):
- Security = Access Control Lists (ACL’s) stored in Oracle9iAS Portal
- Destinations = Web, Printer, Email, File, Oracle9iAS Portal. WebDAV will also be available (on OTN) shortly after production.
- Engines = Report, URL Engine
- Notifications = Email
- Cache Manager = FIFO policy

Open APIs

Open Java APIs enable users to plug in their own implementation for each of these facilities into Oracle9i Reports. While Reports offers distribution to various destinations such as cache, mail, printer, etc., users will also be able to specify their own destinations, such as fax and FTP. You can create your own Security mechanisms/storage for ACL’s. By plugging in new engines, you can transform the Reports Server into an all-purpose server environment. Your engine could e.g. execute a given OS command. By plugging it into the server, you can leverage all it’s other features like scheduling, security, distribution, notification and cache management. So you only have to worry about the immediate logic that is required for your specific task. Using the notification API you can plug in new notifications. By default the server is capable of email-notification on a job-level, so an administrator can be informed, if a job fails. You can expand this functionality using the Notification API. Together with the API for event-driven publishing you can integrate the Reports Services easily into Oracle Workflow using the notification mechanism as interface to the business-event-model.
Peer-Level Clustering

With the new heterogeneous clustering of Reports Services, there is no single point of failure. Users no longer need a centralized cache when they are using Reports servers in clusters. The cache itself is a persistent cache: even if the Reports server is restarted while jobs are in the queue, the cache will retain information about these jobs, in addition to the scheduled jobs.

Servers configured to work in a cluster work as a single entity. When you submit a job to a cluster, one of the servers will pick up the job produce the output, and cache it in its own cache. Each of the servers in the cluster will have access to the output of this job in the cache – so if a duplicate request is received, then it is just redirected to the server with the cached output to return immediately. Even if one server fails or shuts down, the cluster stays operative and ready for processing requests. If the requested job output was is not found the job will be executed again by another member of the cluster.

Reports has some server configuration options for cache, such as 'cacheshare', 'maxcachefilenum', 'cachedir', and now 'expiration' (so content may be proactively expired). Users can also implement their own cache algorithm to make space available when the cache is full.
Oracle9iAS Portal is a web-based application for building and deploying e-business portals. It provides a secure, manageable environment for accessing and interacting with enterprise software services and information resources.

Oracle Reports release 6i has been integrated with Oracle9iAS Portal to have reports available to be run dynamically from within the portal, or scheduled and the reports output pushed directly into pages (which were called folders in previous releases). Note: The reports security functionality also uses Portal as a front end, however that functionality will be covered later as part of Single Sign-on integration.

New for Oracle9i Reports is the ability to place report content on Portal pages as portlets – as shown by the slide. Reports portlets have access to all the standard facilities that portlets have e.g. page-level parameters.
PDF 1.4 Support offers the following:

Compression

By default, PDF output in Reports 9i will be compressed. Files that compressed files are approximately one fifth the size of the original file.

Multi-byte Character Support

Reports 9i supports multi-byte characters for reports formatted in PDF. You can produce reports that display text in Chinese Simplified, Chinese Traditional, Japanese, and Korean.

First, you need to install the desired Asian font pack on your system. You can download the font pack from Adobe’s Web site. Then, you will have to set the NLS_LANG parameter on your system to the appropriate value for that language.

Font Embedding & subsetting

This is where the font is embedded right in your report pdf output – so regardless of the fonts used, it is portable to another machine.

Note: For multi-byte fonts and Unicode fonts, the size of the font runs to megabytes. For this reason, font embedding is supported for single byte fonts only. These fonts average between 30 and 50K and can be compressed in Reports.

ADA/Section 508

The American’s with Disabilities Act requires that output be accessible to users of assistive technologies (e.g. screen readers). Adobe has added features in PDF 1.4 to permit screen readers to “read” PDF documents with Adobe Acrobat reader 5.0 (with an accessibility plug-in). Oracle9i Reports has added properties to allow developers to specify the tags to use when producing accessible PDF output.
EMail Enhancements

**SMTP:** Reports has changed to a single cross-platform model using SMTP only. Existing support for MAPI & Unix Mail has been removed.

Simple Mail Transfer Protocol (SMTP) is the standard e-mail protocol on the Internet. It is a TCP/IP protocol that defines the message format and the message transfer agent (MTA), which stores and forwards the mail.

**Control Delivery of Report Output:** Users are able to send the output of a report to any number of either predefined or data-driven sets of recipients. They can also specify the level of relevance to the recipient - i.e., the “To”, “cc” or “Bcc” lists.

**Provide Identity of the Sender:** When a user mails the output of a report, the report is mailed by the Reports server. Recipients can identify the sender of the mail and the point of contact regarding any details of the report, as in the “Reply To” functionality of e-mail.

**Descriptive “Subject” and “Body” Content:** Professional standards for mail require descriptive “Subject” and “Body” sections of any e-mail. Currently, paper layout sections within Reports are not suitable for “Body” content of a mail message. JSP based output, however, is suitable for producing the “Body” content of a mail message. It can produce either HTML or plain text output and has the free-formatting functionality expected of any mail client.

**Support for Attachments:** Users can attach both dynamic and static content to the mail. Dynamic content is created when the report is run. Static content may be any external file that the user wishes to send with the mail. The requirement for static content satisfies users who wish to attach non-Reports output as well as the collation of the output from several report runs into a single mail message.

**Report Distribution:** When a Report is run, each output that has the same mail DESTINATION can be sent as a separate mail to that destination. Alternatively, the JSP section could be used as the body of the mail with all other sections added as attachments to the mail.
It is often desirable to have a report be initiated by some event that has happened in the organization. For example, when an employee submits an expense report, new data is inserted into the database. When this insert (or 'event') occurs, the employee's manager receives a notification in his portal page or by e-mail asking him to approve/reject this expense report.

With the Oracle Reports 9i feature called event based reporting, this is possible. The notification is a report (paper or Web - or both) that is to be e-mailed to the manager, and/or sent to the manager's portal page. A set of supplied PL/SQL stored procedures can be called from a database trigger to run the report when the data is inserted. The trigger invokes the Reports server and the job is submitted to the server. Using this API you can easily create an integration with Oracle Workflow.
Report Bursting

The capability of producing multiple copies of a given report, or portions of it, in different formats and sending it to different destinations is referred to as report bursting. Distribution allows users to send the output from a single run of the report to multiple destinations.

It is often necessary to distribute a report to different media, such as paper, mail, Web formats or even delimited (Excel) output. Likewise, it may be required that different sections of a single report be distributed in different formats to different sets of people. Starting in Oracle Reports 6.0, Reports has been able to “burst” reports based on the 3 different sections of a report (header, main, and trailer), and distribute them to different sets of users in many different formats. You can create distribution lists for the entire report (report property) and for individual sections of the report (section property).

With Reports 9i, bursting and distribution have been further extended. Users now have the ability to burst on repeating data within a report. For example, based on the department, a report can be distributed to employees within that department. This ensures that each user only receives information that is relevant and appropriate for them. For example, the manager of Department 10 will get an e-mail with the salary information for employees in Department 10; the manager of Department 20 will only be e-mailed the salary information for employees in Department 20, while a summary may be printed out for the CEO of the company. More importantly, the report is actually run only once, and the output is sent to the different sets of users according to the requirements. You only have the overhead of a single data fetch and format in order to distribute the report. This is called “one-pass reporting” and is a key feature in Oracle Reports.
Oracle Enterprise Manager is Oracle’s single, integrated solution for administering and monitoring global e-Business enterprises. Oracle9iAS Reports Services can be managed through the new browser-based user interface for Oracle Enterprise manager. Through the console, you can do the following:

1. Start and stop the reports server
2. Monitor response metrics
   - Average Response Time(ms), Scheduled Jobs, Finished Jobs, Current Jobs, Number of Jobs Transferred, Number of Failed Jobs, Number of Running Engines (and types), CPU usage, Memory Usage
3. Edit server configuration
4. View Server trace information
5. View Server log file
6. See the Finished Job, Current Job, Scheduled Job, Failed Jobs queues.
With the increasing use of Web-based e-business applications, businesses are considering a single sign-on functionality. Without single sign-on, each user must maintain a separate identity and password for each application they access. Maintaining multiple passwords for each user is expensive and insecure. With respect to the single sign-on functionality for Oracle9iAS Reports Services, the authentication to the Reports Server as a resource and the authentication to subsequently accessed data sources has been seamlessly integrated with Oracle9i Single Signon (which uses the Oracle Internet Directory (LDAP) server which can also connect to other LDAP servers if necessary) so there is NO extra code that needs to be written.

There are two levels of security. **Data-level security** defines and optionally stores the logons to the various data source(s) that provide data for reports. This ensures that users can sign-on once and view the reports, and behind the scenes have the Single-signon server provide all the appropriate authentication for all the remote data sources. To further secure your reporting environment, **application-level security** (or access control lists – ACL’s) can be put in place to specify what report requests the users or groups can generate. This application-level security is necessary to ensure that only those authorized users or groups can generate a specific report. Specifically:

- **Who** has privileges to access a report, Report Servers, and printers.
- **What** report definition files, Reports Servers, and printers are available for use.
- **When** reports definition files, Reports Servers, and printers are available for report submission.
- **Where** particular instances of reports may be run, with specific parameters, in a particular format, against particular Reports Servers, on specific printers, etc.

Oracle Reports 6i introduced an open infrastructure for report administration and security with an out-of-the-box implementation using Oracle9iAS Portal. All of the utilities in Portal use wizards for creating, editing, and deleting access control information. Once entered, the Oracle9iAS Portal repository stores the access control information as metadata. Only those users who have Oracle Reports system administrator privileges can access this security information. As previously mentioned, this ACL storage may be ‘plugged out’ and replaced with a user-defined ACL system.
## New Features - Summary

### DEVELOPMENT
- JSP and XML-based Report Definition
- XML, JDBC, Oracle® Olap, Text Files
- Pluggable Data Sources
- PL/SQL to Java Bridge
- Integration with Oracle® JDeveloper
- Integration with Oracle® SCM
- Graphing
- Oracle® AS Portal Report Import
- Edit Oracle® AS Discoverer Worksheet

### DEPLOYMENT
- Servlet and JSP Runtime Architecture
- Open API’s – Pluggable Facilities
- Oracle® AS Reports Services Clustering
- Reports as Portlets
- PDF 1.4
- Enhanced Email support
- Event-driven publishing
- Enhanced Report Bursting / Distribution
- OEM Integration
- SSO Integration