# Oracle Developer/2000 - Beyond Client/Server Reporting

*An Oracle White paper*

August 1997

ORACLE®
Enabling the Information Age™

## Introduction

With the development of the desktop computing environment and the graphical user interface in particular, there has been a corresponding growth in PC centric reporting tools. These tools have, through simple design environments, made access to corporate information a relatively simple operation. However, as productive these development tools have become, it is the limitations of the client/server environment itself that has prevented most companies from taking advantage of these tools for their mission critical or operational reporting needs. That is, the size, complexity and speed requirements of operational reporting for most organisations dictate the need to be server/host based, and hence it is difficult to take advantage of the development productivity that PC-desktop software provides.

The requirement for large scale operational reporting therefore may be simplified down to two main criteria: ***performance*** and ***scalability***.

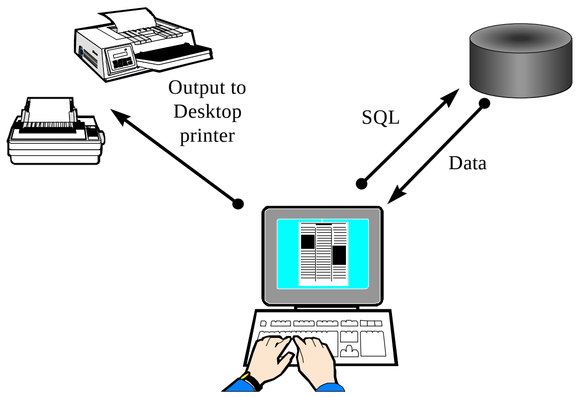


**Figure 1**: The Standard Client/Server Reporting model

Currently, with most client-server implementations, the execution of a report is a highly client-intensive process. Although the data of a report is extracted from the database server, all report formatting is still done on the client machine, which is often a personal computer with limited processing power and memory capacity. This can lead to the situation where a report being formatted locally on the client causes local machine resource to be entirely consumed by the report execution itself. Likewise, the spooling nature of most desktop operating systems does not lend itself to the printing of large scale reports. That is, the required spool file created can severely restrict the amount of disk storage available to the user, causing inconvenience or even operating system errors (e.g., when using dynamic virtual memory sizing on Windows 95 ™).

Furthermore, if the same report is desired by multiple users, as is generally the case within an application environment, it will be executed for each user request. As such, each user will re-execute the query, return the same data across the network and be formatted independently of all other users of the application. Hence duplicating the load on the database, network and workstation in order to allow multiple people to access the same information. The converse would be to generate the output once, then determine a method of passing it to those to whom it is of interest (sneaker-net?). Neither method is ideal.

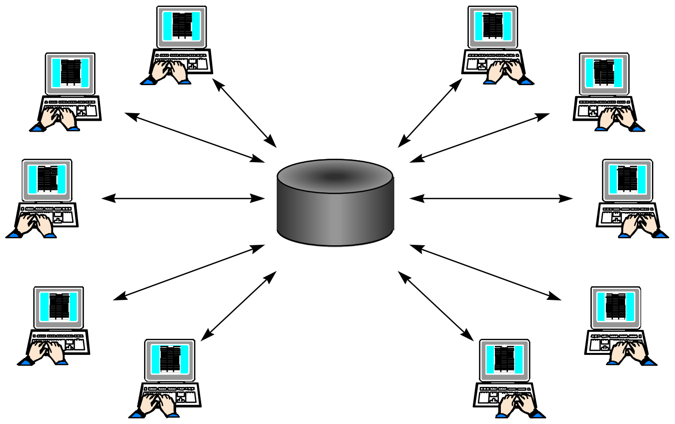


**Figure 2:** Users Acting in Isolation

This paper will discuss the features available within Oracle's Developer/2000 that address the requirements for enterprise level reporting and introduce how the product may take part in a multi-tier environment.

## Web Publishing

### Static Publishing

It was the need to share information throughout their user community that lead the Physicists and Engineers at CERN to develop the technology that has evolved into what is now known as the World Wide Web. The ability to publish a document in Hyper Text Markup Language (HTML) allowed for a single copy of a document to be accessed by multiple users simply by use of a browser. This publishing model is the simplest form of multi-tier reporting, relying as it does on the pre-creation of the report output based on static criteria.

Oracle's Developer/2000 has established a solid reputation for the ease of development and generation of complex reports across multiple platforms. Release 1.3.2 extended this portability to enable the utilization of Web technology as a publishing medium by it's ability to generate report output in the standard formats used by the World Wide Web. That is, HTML (version 3.0) and Adobe's Portable Document format (PDF).

As such, long running/operational reports may be generated periodically, the output placed on a web server and accessed by many users simultaneously.

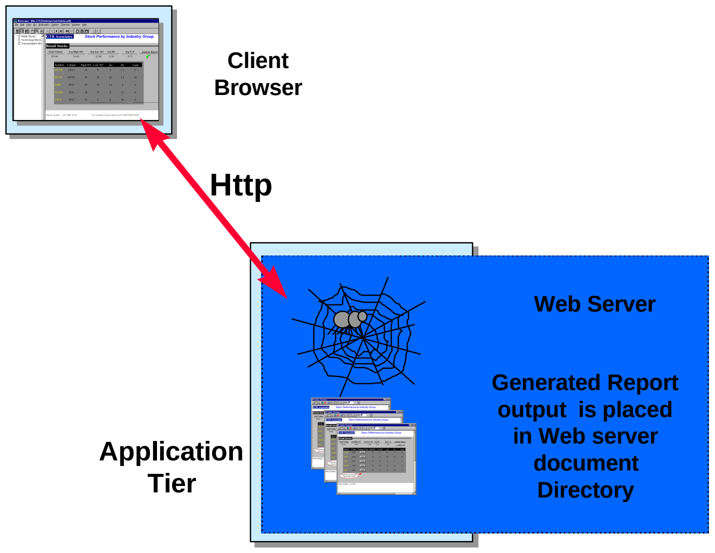

Client
Browser
Http
Web Server
Generated Report output is placed in Web server document Directory
Application Tier

Figure 3: Web Model of Static report publishing

## Dynamic Publishing

Whereas the data dynamics required allow many applications to take advantage of a static publishing methodology (such as Financial reporting or Data warehouses where the data is only refreshed at a specific point in time), they are immediately constrained by the need to either restrict the possible queries that may be issued, or otherwise by the sheer number of static documents that need to be generated in order to meet user requests. In order to negate these limitations, **Developer/2000 Release 1.4W** introduces the ability to define at runtime the parameters and criteria of a given report.

By the implementation of the reports runtime engine as a CGI (common gateway interface) executable, a URL (Uniform Resource Location) now mimics the standard command line as used in a client/server two-tier environment.

e.g.:

**Http://d2kserv.us.oracle.com/r25cgi32?REPORT=FRED.RDF+ USERID=scott/tiger+DESFORMAT=HTML+EMPNO=10**

When a URL of this nature is received, a runtime engine is spawned and the specified report executed. The output is then returned in the desired format to the calling browser.

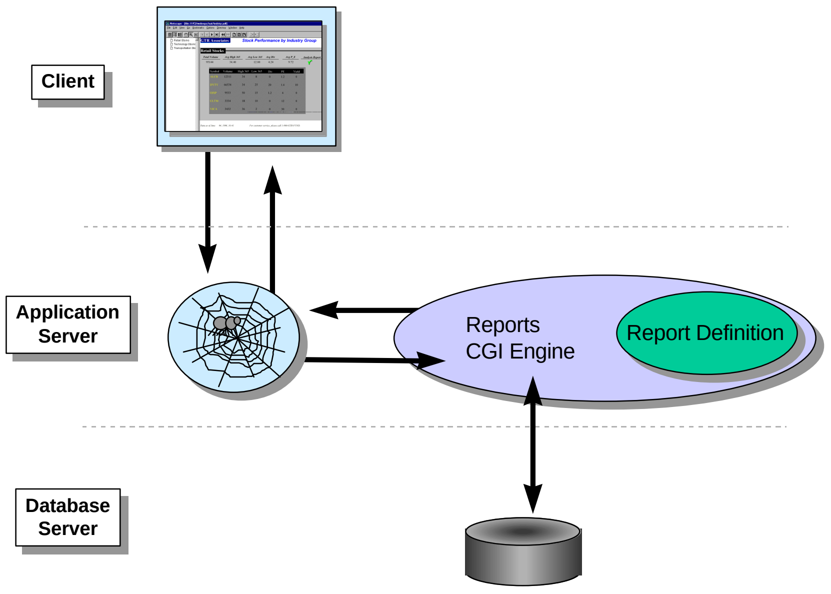


**Figure 4: Developer/2000 R1.4 Dynamic web reporting With Reports Web Interface Component**

In order to prevent confidential information from being displayed within the URL Developer/2000 is able to take advantage of report parameters stored on the server by use of the CGICMD.DAT command file. This will be discussed in more detail later in this paper.

## Reporting within Developer/2000 Web Cartridge based applications

Many applications developed for client/server utilize Developer/2000's Forms component to develop sophisticated parameter forms for their reporting needs. With the release of the Developer/2000 Web Cartridge those applications may now be used with the web environment fundamentally unchanged. That is, the RUN_PRODUCT procedure used to initiate a report from within a form is supported within the web based framework.

In order to successfully invoke a report from a Web based form the Forms Server must first know from where to pick up the generated report and in which default format it was generated. This is done by the setting of the following environment variables prior to the starting of the forms server process:

| | |
|---|---|
| FORMS45_OUTPUT | Physical directory on the server for report output. |
| FORMS45_MAPPING | The virtual directory defined to the web server which points to the physical directory defined above. |
| FORMS45_REPFORMAT | The output format for the generated report. |

When the RUN_PRODUCT command is issued from the Forms module, the report is generated as normal with the output being directed based on the value of destination type (DESTTYPE). If this is set to FILE, the generated report is sent to the directory specified in FORMS45_OUTPUT. If, however, this is set to either SCREEN or PREVIEW, a temporary file is created (also in FORMS45_OUTPUT) and passed to the web server via it's virtual mapping (FORMS45_MAPPING). Based on the value of FORMS45_REPFORMAT, the web server will then display the report on the user's browser window.

## Scaleable Multiple-Tier Reporting

Use of the Common Gateway Interface has enabled the World Wide Web to become a much more dynamic place. However, as this dictates spawning a new process for each request, it limits the effective scalability of the application, while at the same time increasing server overhead and subsequently decreasing performance.

**Developer/2000 Releases 1.5 & 2.0** introduce a true multi-tier architecture for executing and distributing your reports. The centerpiece of this new architecture is the Developer/2000 Multi-tier Report Server. With this, reports may be executed remotely on much more capable UNIX or NT server platforms where resources are greater, and at the same time significantly reduce the load on the client machine.

The Multi-tier Report Server is a multithreaded executable which generates report output dynamically based on requests received from clients. These requests may be issued from end user PCs, from an interface with any web server, or from third-party applications via the Reports ActiveX Control. The Reports Server handles client requests by entering all report submissions into a job queue, as one of the server's runtime engines becomes available, the next job in the queue is dispatched to that engine and executed.

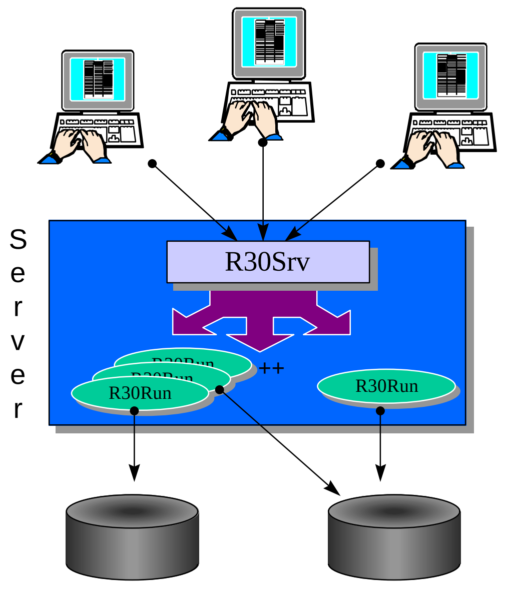


**Figure 5: Developer/2000 Report Server Architecture**

.

## Intelligent Load Management

Requests issued from clients will be distributed to a dynamic and configurable number of Report Server runtime engines, hence multiple reports may be executed concurrently from a given Developer/2000 Reports Server. As the number of jobs in the queue increases, the Multi-tier Report Server will dynamically invoke additional engines to execute queued jobs as necessary up to a maximum limit **`[maxengine]`** set in a Report Server configuration file. During idle periods in which there are more server engines than job requests in the Report Server queue, idle reports runtime engines can be automatically to conserve machine resources. The maximum idle period **`[maxidle]`** along with several other parameters may be defined in the Report Server configuration file **`servername.ora`**.

| | |
|---|---|
| **maxconnect=20** | Number of maximum concurrent process that can connect to the server (client + runtime engines). |
| **sourcedir="/WORK/REPSRC"** | Directory containing report source files (server will look here first then along the path defined by REPORT30_PATH) . |
| **cachedir="/ORACLE/OWS20/DOCS"** | Where the cache should be stored. (NOTE: This must point to the Web Server Document directory if using the Web interface). |
| **cachesize=50** | Maximum size of the cache in MB. |
| **minengine=1** | Minimum number of report engines running at any one time. |
| **maxengine=3** | Maximum number of engines that will be started as load on the server increases . |
| **initengine=1** | The number of engines started up when the server is first started. |
| **maxidle=30** | Length of time (mins) before an idle engine is shutdown. |
| **security=1** | Defines the access privilege level to retrieve documents from the queue (e.g., via queue manager) 1=owner, 0=all. |

**Table1:Typical Servername.ora file with descriptions of entries**

## Cached Report Output

The benefits of the static model for web publishing was the immediacy of the returned output (being pre-generated). The disadvantage, as previously mentioned, being the administration and lack of a current view of the data. The Developer/2000 Report Server solves many of these issues by use of an output cache for generated reports. If a given report has had multiple submission requests with the same parameters and is within a defined time period (as defined by the command line argument **TOLERANCE=#min**), the server will determine that they are duplicates and deliver to the requestor the output from the cache instead of executing the report multiple times. (NOTE: the server determines a duplicate request based on the following parameters: report, userid, desformat or desname where it specifies output format, paramform, currency, thousands, decimal, pagesize, orientation, mode, and all user parameters.)

## Submitting Reports to the Server

Client requests for reports will be prioritized on a first-come, first served basis and my be submitted to the server via the following methods:

- **Lightweight Client [R30CLI] -** A lightweight executable - r30cli is all that is needed on a client machine to submit a report to the Report Server. The following is a typical command line for submitting a report to the Report Server using r30cli.

```
r30cli REPORT=trades.rdf MODE=character DESTTYPE=file
        DESNAME=C:\out\tradenew.lis SERVER=d2kserv
```

- **Web Server CGI Interface [R30CGI] -** Web Server independent interface to the Report Server that takes a given URL as input and communicates to the Report Server. Hence allowing users of any web server to generate and receive dynamic reports from their web browser.

- **Web Application Server Cartridge Interface** - Plug-in cartridge for Oracle's Web Application Server, has a similar syntax to the CGI based interface.

- **Reports ActiveX Control -** Submit and displays reports from any third party application development tool that supports the use of ActiveX (OCX) controls.

- **Direct Developer/2000 Forms Interface**

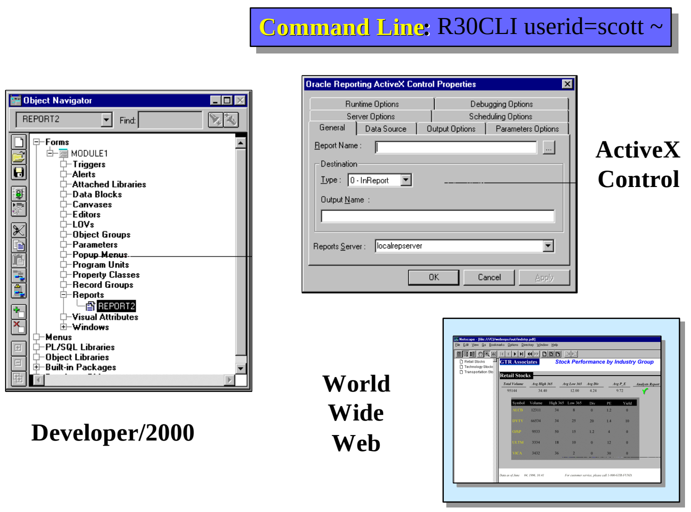

Command Line: R30CLI userid=scott ~
ActiveX Control
World Wide Web
Developer/2000
Object Navigator
Oracle Reporting ActiveX Control Properties

**Figure 6: Client Interfaces to Reports Server**

Note the new command line argument **SERVER=<name>**. The purpose of this is to specify to which Report Server process the report should be submitted. The remote procedure call (RPC) mechanism, which is used to communicate between client and application server, uses SQL*Net version 2.3 as it's transport layer and hence is able to take advantage of all the Domain Name Services and Fail Over benefits which are inherent with in it. As such, the naming of a Report Server is managed in the same manner as a SQL*NET V2 alias to a database. That is via the TNSNAME.ORA file.

e.g.
For a report server called 'd2kserv' the entry would, assuming a domain of world, look like the following;

```
            d2kserv.world=(
                        ADDRESS=
                        (PROTOCOL=tcp)
                        (HOST=d2kserv.us.oracle.com)
                        (PORT=1949)
                        )
```

With high performance, document caching and easy access methods the Developer/2000 Report Server lends itself to the implementation of an application server, particularly in the area of high end printing. At the present, this type of large scale report generation is generally the domain of procedural tools/languages and hence many of the benefits of the graphical development environment are not available. With the Report Server the user is able to take advantage of the appropriate technology for the task. GUI development on a client work station, performance and scaleable deployment.

## Scaleable Web Publishing

The enterprise reporting needs of an organization  generally requires access by a large number of concurrent users, hence scaleablity is a major concern. However it is in the confines of a dynamic web site that scaleability becomes an even greater criteria. Unlike an application aimed at a user community within an organization, the user base for a web site is not necessarily a known quantity. Nor are the times of peak load as easily determined.

If a web site is to be successful it must be sufficiently informative, visually pleasing, and most importantly performant enough to entice the user to a given web page. The use of dynamic load balancing of job requests and especially the report output cache with Developer/2000 means that web requests may be processed instantaneously (in the case of cached reports) or at least processed without being impacted by the number of requests coming from other web users.

As more and more users access their reports via a web browser from the server, the administration of the application server becomes more difficult. If a user is required to enter all report arguments as part of the calling URL the likelihood of error increases. Likewise the need to return confidential information (such as username/password) as part of that URL  is a possible security issue. The Developer/2000 Reports Server addresses these issues through the following mechanisms;

1) Use of a protected server side configuration file CGICMD.DAT.

   This file is used to map KEYWORDS with appropriate command strings, such that if a given keyword is issued as the first argument of the URL the CGI component will substitute it for the corresponding command line.

```
rfund:          REPORT=rfund USERID=scott/tiger@NTO8 SERVER=d2kserv
                TOLERANCE=0 DESTTYPE=cache DESFORMAT=PDF %*

rfundht:        REPORT=rfund USERID=scott/tiger@NTO8
                SERVER=d2kservTOLERANCE=0 DESTTYPE=cache
                DESFORMAT=HTML %*

rfundcss:       rfund scott/tiger@NTO8 server=d2kservNT tolerance=0
                destype=cache desformat=htmlcss %*

mailmrg:        REPORT=mailmrg USERID=scott/tiger@NTO8
                SERVER=d2kservNT TOLERANCE=60 DESTTYPE=CACHE
                DESFORMAT=PDF %*

mailcust:       REPORT=mailmrg USERID=scott/tiger@NTO8
                SERVER=d2kservNT TOLERANCE=0 DESTTYPE=CACHE
                DESFORMAT=PDF %P
```

**Table 2: Example CGICMD.DAT file**

By use of the URL keyword mapping available with the Developer/2000 Report server, a complex URL such as:

**HTTP://d2kserv.us.oracle.com/r30cgi32?REPORT=rfund+ USERID=scott/tiger@NTO8+SERVER=d2kserv+TOLERANCE=0+ DESTTYPE=cache+DESFORMAT=PDF**

Becomes:

**HTTP://d2kserv.us.oracle.com/r30cgi32?rfund**

The %* at end of the command line entries within the CGICMD.DAT file further indicates to the Report Server Web Interface Component (WIC) to additionally add any arguments that the user may have entered via the URL, with any duplicates being ignored.

2) Using a automatically generated HTML parameter form to allow the end user to see and fill in the required parameters themselves at runtime. *Note: This feature is available with the 2.0 release of the Developer/2000 Reports* server.

   In order to notify the server that a runtime parameter form is required, an extra command argument is added to the command line within the CGICMD.DAT key map file. The addition of **'%P'** as seen above is all that is required to have the server generate an HTML parameter submission form equivalent to the parameter form defined in the report itself.

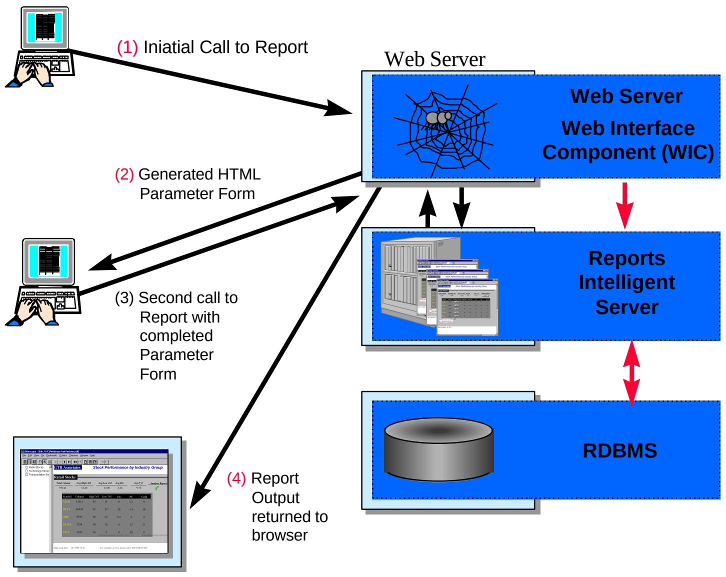

(1) Iniatial Call to Report
Web Server
Web Server
Web Interface Component (WIC)
(2) Generated HTML Parameter Form
Reports Intelligent Server
(3) Second call to Report with completed Parameter Form
RDBMS
(4) Report Output returned to browser

**Figure 7: Sequence of steps to generate a report with an HTML parameter form.**

Step(1) The initial call to the report via the URL executes the report, the %P flag indicates to server that a parameter form should be generated.

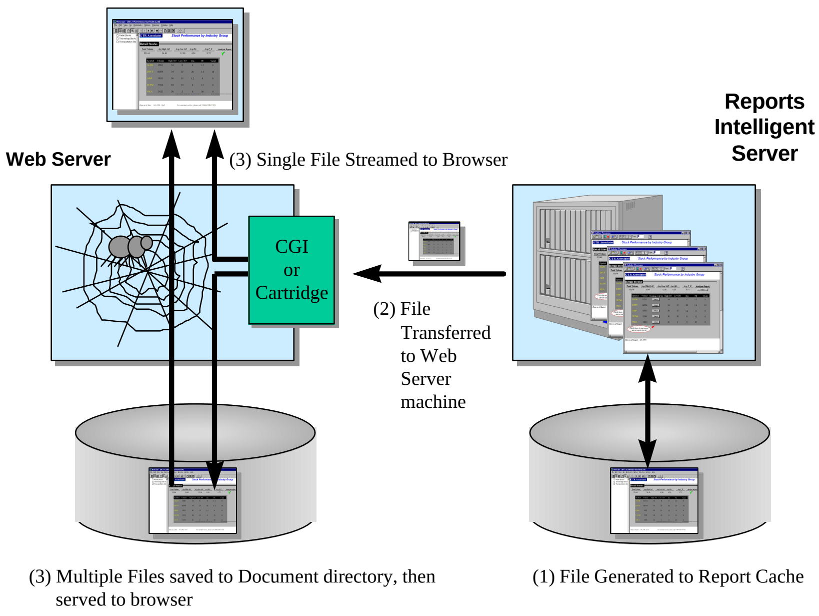
Step(2) HTML parameter form is generated containing information indicating that this is the initial call to the server for this report. All specified parameter fields containing pop lists are populated (either statically or from the database).

Step(3) User submits completed parameter form. Server interrogates the submitted URL information to determine if this is the initial call to the server.

Step(4) Having determined that the URL contains the appropriate flag to indicate that it was submitted from the parameter form, the server generates the report output and passes it back to the calling browser.

NOTE: the use of the %P flag requires that a CGICMD.DAT file exist. If such a file is not being used then the use of the parameters **PARAMFORM=HTML** or **PARAMFORM=HTMLCSS** in the URL itself will have the same result. That is a parameter form (in the specified format) will be generated.

The Developer/2000 Report server further enhances scalablity of the web based reporting environment by allowing the Report Server to reside on a different machine than the web server itself (as shown in Figure 7). Hence spreading the load on machine resources.


Web Server
(3) Single File Streamed to Browser
Reports Intelligent Server
CGI or Cartridge
(2) File Transferred to Web Server machine
(3) Multiple Files saved to Document directory, then served to browser
(1) File Generated to Report Cache

By the definition of two environment variables on the web server platform the WIC is able to tell the web server from which directory it should retrieve the generated document.

The **WEBLOC** variable defines the virtual directory (as specified within the web server) where report output will be placed and hence read by the web server. This variable is mandatory even if the two servers reside on a single machine.

The setting of **WEBLOC_TRANSLATED** to the physical path specified by **WEBLOC,** has the effect of informing the WIC that the report output is physically located on a separate server platform, and hence should be automatically copied to the directory specified by this environment variable. Note, If the report output is in a format where it is within a single file (e.g., PDF), a file is not created on the Web Server, but rather redirected (streamed) to the calling browser.

Whereas using a shared file system would also allow for the two server processes to physically reside on different platforms, the use of **WEBLOC_TRANSLATED** allows the two servers to be totally independent of each other. As such the web server could reside outside a company's network firewall while the Report Server (and it's file cache) could reside within it's security umbrella (requires firewall to support SQL*Net pass through).

## Extended WEB Publishing with Developer/2000 Release 2.0

Whereas the use of HTML has enabled the publishing of vast amounts of data on the web, the limitations of the original language specification are now becoming more apparent. Users are requesting more and more that their web applications have a similar look and feel as those in the client/server environment. If their desktop application supported multiple font types then it was reasonable to assume that the same should be available under the web, and so on.

In order to rectify this limitation the W3 committee (the organization which directs the specification of the HTML standard) is extending the specification of HTML to further the publishing capabilities of the standard. The draft specification of HTML 3.2 introduces the concept of an HTML style sheet. With this web page format authors will not only have greater access to many more fonts than the eight currently supported by HTML, but also much greater control over the physical placement of objects on a page and their text and fill colours (standard HTML currently has very limited support for background colours on individual text items). Support for style sheets is currently found in Microsoft's Internet Explorer 3.0 and is expected to be included in the upcoming release of Netscape 4.0 and other Browsers.

Developer/2000 Reports is at the forefront of database publishing for the web and will, with Release 2.0, support the use of HTML style sheets as output. In order to take advantage of this emerging HTML standard the user need only change the value of the DESFORMAT command line argument.

| | |
|---|---|
| DESFORMAT=HTML | Output based on HTML 3.0 tables |
| DESFORMAT=HTMLCSS | Output based on the draft HTML 3.2 style sheets. |
| DESFORMAT=PDF | Will create the output in Adobe's portable document format for desktop publishing quality output on the web (requires Adobe's Acrobat plug-in). |

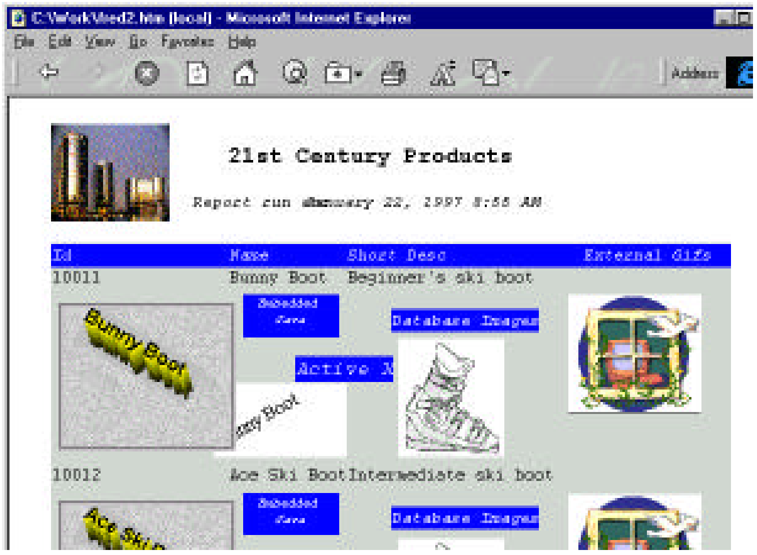


**Figure 8: Style Sheet based report with embedded ActiveX controls, Java Applets & Animated GIFs**

With release 2.0, Developer/2000 will allow web developers to further enhance the interactive nature of web pages generated by Reports by supporting the automatic in place publishing of both images based on an image source tag, as well as user defined HTML script.

Developer/2000 Reports has always had the ability to 'READ FROM FILE' a given data column specified within a query. That is, rather than returning the character string that is name of file, actually load the contents of the file at that point in the report. The ability to read from URL now allows for the use of such things as animated GIF images to be included in a report (e.g. Animated Logo) or having the image files themselves on a separate machine from where the report was generated.

e.g.

| | |
|---|---|
| Source File Format : | URL |
| Source File Location: | HTTP://www.oracle.com/tools/images/image.gif |

Likewise, in order to allow for interactively within the generated web output, the author may wish to take advantage of either JAVA Applets or Active X controls (or both) within the report itself. By setting the property 'HTML Tag' on the field (or boilerplate text object) the developer is indicating that the contents are to be treated as user defined HTML and not formatted as standard text. Furthermore, as external boilerplate may contain field/column references like any other boilerplate text item, any user defined HTML may contain references to data within the report. The upshot being that any applets/controls which are embedded within the report may change dynamically altered to reflect the current data within the report.

e.g., text field containing the following:

```
<APPLET CODE=animator.class
        WIDTH=&<ObjectWidth>HEIGHT=&<ObjectHeight>
        <PARAM NAME=imagesource  VALUE=&<LOCATION>>
        <PARAM NAME=deptno       VALUE=&<DEPTNO>>
</APPLET>
```

would create an applet scaled to the size specified within the report layout and displaying the current value of both DEPTNO and LOCATION columns.  As the area defined to hold the external applet/control  may be different to it's ultimate size within the browser,  the system variables ObjectWidth and ObjectHeight may be used to force the applet/control to scale to the correct fixed size.

NOTE: whereas the example used shows the use of a JAVA class, an ActiveX control may be inserted in the same manner.

## CONCLUSION

As the reporting needs of most organisations increase, and the realization that the client/server model of reporting is not suited for enterprise level reporting, more and more organizations are looking to both application server architectures and/or the web to solve some of the scalability and performance issues that plague them.  With Developer/2000 Report Server organisations now have avaliable a highly scaleable reporting engine which allows for a choice of clients, whether they be traditional client server applications or the World Wide Web.

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
+1.415.506.7000
Fax +1.415.506.7200
http://www.oracle.com

Developer/2000 - Beyond Client/Server Reporting
An Oracle Technical White Paper
Author: Barry Hiern