(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 August 2003 (07.08.2003)

PCT

(10) International Publication Number
WO 03/065174 A2

(51) International Patent Classification^7: G06F

(21) International Application Number: PCT/US03/03095

(22) International Filing Date: 3 February 2003 (03.02.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/353,239 1 February 2002 (01.02.2002) US

(71) Applicant *(for all designated States except US)*: **HARVARD BUSINESS SCHOOL PUBLISHING** [US/US]; 300 North Beacon Street, Watertown, MA 02472 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **ELLSWORTH, Amelia** [US/US]; 700 Huron Ave., Apt. 7C, Cambridge, MA 02138 (US).

(74) Agents: **MORANO, Elizabeth, P.** et al.; Bromberg & Sunstein LLP, 125 Summer Street, Boston, MA 02110-1618 (US).

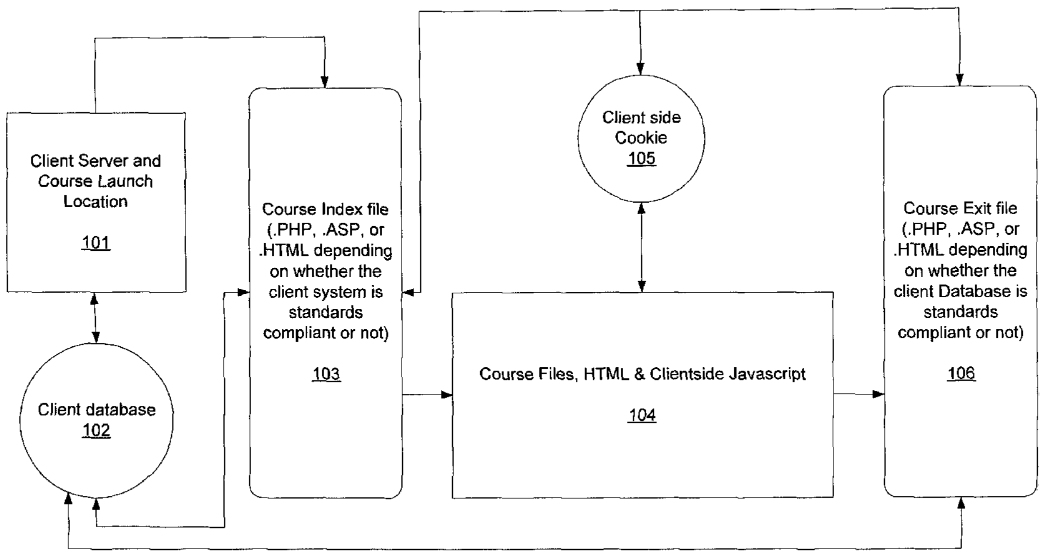
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: APPARATUS AND METHOD FOR PROVIDING INFORMATION


Client Server and Course Launch Location
101
Client database
102
Course Index file (.PHP, .ASP, or .HTML depending on whether the client system is standards compliant or not)
103
Client side Cookie
105
Course Files, HTML & Clientside Javascript
104
Course Exit file (.PHP, .ASP, or .HTML depending on whether the client Database is standards compliant or not)
106

(57) **Abstract:** A computer product provides interactive content for use in a computer system that optionally includes a network of computers. The computer product comprises: a digital storage medium, the storage medium encoded with computer readable files as follows: a first set of files providing interactive content using a markup language format and a script format, the scripting language for controlling data communication between a user and a program source; a second set of files providing compatibility of the interactive content for use with a first e-learning standard environment; and a third set of files providing compatibility of the interactive content for use with a second e-learning standard environment.

WO 03/065174 A2

Apparatus and Method for Providing Information

Technical Field

The present invention relates to network based learning environments and, more particularly to providing an apparatus and methods for distributing interactive content regardless of particular access requirements.

Background Art

It is known in the art to provide Web browsers to display Web pages over a network such as the Internet. While browsers are now so widely distributed that they can be used as interfaces for various programs, Internet access constraints have traditionally prevented the distribution of interactive content in one single universal format. Clients with different types of access needs all want the same interactive content, and companies who want to host a particular content have different server environments into which they want to integrate the content. Maintaining multiple versions of the same interactive content to cover distribution needs is typically messy and expensive.

In the HTML environment, it is known for a web server to utilize a short-form message format, termed a "cookie", for communication with a computer accessing web pages via a web browser. The browser environment will store such a message, for example, in a text file called "cookie.txt." The message may then be sent back to the server each time the browser requests a page from the server. Cookies may be used to identify users and prepare customized Web pages for them. When a user enters a Web site using cookies, he or she may be asked to fill out a form providing such information as the user's name and interests. This information is packaged into a cookie and sent to the Web browser to store for later use. The next time the user enters the same Web site, the browser will send the cookie to the Web server. The server can use the information to present the user with a custom Web page. For example, instead of seeing just a generic welcome page, the user might see a welcome page with his or her name on it. Cookies are derived from UNIX objects called "magic cookies." These are tokens that are attached to a user or program which change depending on the areas entered by the user or program. Cookies are sometimes called "persistent cookies" because they typically stay in the browser for long periods of time. Cookie data can be associated with a user and saved to a database. This allows the data to be written to the client machine the next time a program is accessed, and compensates for the possibility that a cookie could be overwritten by the browser. Information regarding cookies and their use may be found at www.cookiecentral.com and specific programming details and usages appear at www.cookiecentral.com/faq/#3. The documents on this web site are hereby incorporated herein by reference.

Integration flexibility now exists with the introduction of the AICC and SCORM standards into the traditional interactive training and eLearning systems. AICC, the Aviation Industry CBT (Computer Based Training) Committee, is an international association of technology based training professionals. The AICC promotes interoperability standards that software vendors can use across multiple industries. AICC recommendations are fairly general to most types of computer based training, and for this reason are widely used outside of the aviation training industry. Further details are available at the web site of the AICC, www.aicc.org. Technical standards of the AICC are accessible from the web site. At the time of this writing the technical standards could be found at www.aicc.org/pages/down-docs-index.htm#WHITE. SCORM, Sharable Content Object Reference Model, is an XML-based reference model that defines a Web based learning "content model." The SCORM standards are distributed by Advanced Distributed Learning Initiative Network, which has a web site setting forth these standards at www.adlnet.org. See also, for example, " R., Cover, "Shareable Content Object Reference Model Initiative (SCORM)" in *The XML Cover Pages* at http://xml.coverpages.org/scorm.html. The foregoing web-based documents are hereby incorporated herein by reference.

Summary of the Invention

In a first embodiment, computer product provides interactive content for use in a computer system that optionally includes a network of computers. The computer product comprises:

a digital storage medium, the storage medium encoded with computer readable files as follows:

a first set of files providing interactive content using a markup language format and a script format, the scripting language for controlling data communication between a user and a program source;

a second set of files providing compatibility of the interactive content for use with a first e-learning standard environment; and

a third set of files providing compatibility of the interactive content for use with a second e-learning standard environment.

In a related embodiment, the computer product also includes a fourth set of files, which may be included in any of the other sets of files, for identifying the nature of an e-learning environment in which interactive content has been placed, and invoking processes for providing data pass consistent with the thus identified e-learning environment, or providing no data pass at all.

In related embodiments of the invention, the first e-learning standard environment may comply with AICC standards. In other related embodiments of the invention, the second e-learning standard environment may comply with SCORM standards. In additional related embodiments, the scripting language is selected from the group consisting of JavaScript, ASP script, and PHP script.

In accordance with another embodiment of the invention, a method of providing interactive content, for use in a computer system that optionally includes a network of computers, includes:

providing interactive content using a markup language format;

along with the provided content, providing a first set of files providing compatibility of the interactive content for use with a first e-learning standard environment and a second set of files providing compatibility of the interactive content for use with a second e-learning standard environment;

in the course of providing interactive content, identifying the nature of an e-learning environment in which interactive content has been placed;

for each user experiencing the interactive content, using a cookie to provide a record of the user's place in experiencing of the interactive content.

In accordance with a related embodiment, the cookie may also provide a record of the user's data in experiencing of the interactive content. In accordance with another related embodiment, the content is provided in a computing environment devoid of a network of computers.

Brief Description of the Drawings

The foregoing features of the invention will be more readily understood by reference to the following detailed description, taken with reference to the accompanying drawings, in which:

Fig. 1 is a block diagram illustrating an apparatus for providing information in accordance with one embodiment of the present invention;

Fig. 2 is a flow chart illustrating a method for providing information when no database is available;

Fig. 3 is a flow chart illustrating a method for providing information in accordance with proprietary or non-standards based server access;

Fig. 4 is a flow chart illustrating a method for providing information in accordance with standards based server access; and

Fig. 5 is an illustration of a graphical user interface for accessing files in accordance with the embodiments of Figs. 1-4.

Detailed Description of Specific Embodiments

Definitions: For the purpose of the present description, the following terms shall have the indicated meanings unless the context otherwise requires:

"Content" refers to information in an interactive format.

A "content package" refers to content and additional files that permit the content to be compatible with a plurality of interactive content environments in accordance with embodiments of the present invention.

Embodiments of the invention allow interactive content to be delivered to the widest audience possible. In this manner, content remains the focus, and does not have to bend to fit distribution needs.

In accordance with embodiments of the invention, one version of the same course content is allowed to work with a plurality of different distribution methods. Such distribution methods include, but are not limited to: Internet delivery from a server using a standards compliant database or system; Internet delivery from a server using a non-standards compliant database or system; firewall delivery from a server using a standards compliant database or system; firewall delivery from a server using a non-standards compliant database or system; Intranet/Internet delivery from a server with no database; and distribution on CD or from any computer media with enough space to hold the content. In this manner, embodiments of the invention may be used in a stand-alone environment or in a network environment.

The course content itself has been separated from the technical implementation files, allowing this process to apply to virtually any content. Embodiments of the invention allow course content interaction to be the same for all access methods, provide a simple implementation file package to fit all environments mentioned above, and enable the course to detect its environment. Instrumentality may be constrained to degrade gracefully so as to enable the experience of the content, regardless of client limitations. Database systems here may hold data from client side cookies so the user identifier and experience data (the interactive responses) can be saved and passed from machine to machine.

Fig. 1 is a block diagram illustrating components for providing content in accordance with one embodiment of the present invention. Such an embodiment may be used with a course that provides interactivity through an interactive case and short practice activities, along with expert feedback for self-evaluation. A section of the course with interactive online tools also helps the user prepare, practice, and review key concepts as they work. The interactive content is written in HTML, using client side JavaScript and the appropriate server side scripts to go through the steps outlined below. (Although JavaScript has been employed here, any other suitable scripting language for controlling data commands may be employed, such as ASP (Active Server Pages}), PHP (short for PHP: Hypertext Preprocessor) script.

In accordance with embodiments of the invention, a client server **101** may or may not be in communication with a client database **102**. The client server **101** and client database **102**, if present, have access to a content package which includes course Index files **103**, course content **104**, course Exit files **106**, SCORM and AICC files for standards based server integration and PHP and ASP files for a proprietary server implementation.

The implementation files included in the content package **103**, **104** and **106** for this embodiment are shown in Fig. 5 and code pertaining to these files is included in Appendix A attached hereto. This code implements some of following eleven steps described in further detail below:

1. The client starts the content.
2. The content determines what data pass method, if any, is in place.
3. The content determines what identifiers and existing data exist, if any
4. If data pass is available, the content writes a cookie to the user's hard drive using traditional browser conventions. This cookie contains the updated information from the database if the user is returning. If this is the first visit by the user, the cookie is populated with default settings that will change as the experience continues.
5. If data pass is not available, the content checks for a cookie. If one already exists, the user picks up where the cookie left them last. If no cookie is present, the content writes a cookie with default settings to the user's hard drive using traditional browser conventions.
6. The user begins an interactive experience with the content.
7. The process uses client side cookies to hold user experience data as the user proceeds through the content. If the content was launched from a server environment, the content will periodically send data back to the server through the appropriate method to maintain a connection and prevent data loss.
8. When the user exits the content, final data is written to the cookie on the hard drive.
9. The content notes the data pass type initially detected
10. The content passes variable data and cookie text back to the parent database if it is available. Once data pass has been successfully achieved, the cookie is erased to prevent data overflow.
11. The course exits.

These implementation files are the content files, the standard AICC files, the standard SCORM files, Oracle and MSSQL database scripts for those systems without a standards compliant database, P3P example files for IE 6.0 privacy settings, and both index and exit files for PHP, ASP and standard HTML.

The index and exit pages in the content package **103**, **104** and **106** are the most relevant to the embodiment outlined above. The Index.PHP and the Index.ASP do basically the same thing, but for different environments. These files are designed to allow integration of content to occur in a proprietary (or non-standards compliant) server environment (described in detail below). At the end of each one, they point to the

Index.HTML file that finishes up the data detection process, and the course experience begins.

At the end of the user experience, the Exit.PHP and Exit.ASP files are called by the Exit.HTML file upon realization that a proprietary (or non-standards compliant) server passed data into the content at the beginning of the experience. An example of program code associated with these files is provided in the attached Appendix.

A client starts the content package **103**, **104** and **106** via the server **101**. The content package **103**, **104** and **106** will determine what data pass method, if any, is in place. The content package **103**, **104** and **106** will also determine what identifiers and existing data exist, if any exist. If data pass to a data base **102** is available, the content package **103**, **104** and **106** writes a cookie **105** to the user's hard drive using traditional browser conventions. The cookie **105** contains the updated information from the database **102** if the user is returning. If this is the first visit by the user, the cookie **105** is populated with default settings that will change as the experience continues.

If data pass is not available, the content package **103**, **104** and **106** checks for a cookie **105**. If one already exists, the user picks up where the cookie **105** left them last. If no cookie **105** is present, the content package **103**, **104** and **106** writes a cookie **105** with default settings to the user's hard drive using traditional browser conventions. The user then begins an interactive experience with the content in the content package.

The content package **103**, **104** and **106** uses client side cookies **105** to hold user experience data as the user proceeds through the content. If the content package **103, 104** and **106** has been launched from a server environment, the content will periodically send data back to the server through the appropriate method to maintain a connection and prevent data loss. When the user exits the content, final data is written to the cookie **105** on the hard drive. The content package **103**, **104** and **106** notes the data pass type initially detected and passes variable data and cookie text back to the parent database **102** if it is available. Once data pass has been successfully achieved, the cookie may be erased to prevent data overflow. Then the course exits.

Fig. 2 is a flow chart illustrating a method for providing information when no database is available. If no database **102** is available, the client calls **201** the course Index file **103** in .HTML formal (Index.HTML) via a static Web link. The index.HTML does not detect data pass, and so it searches **202** for a cookie **105** on the local disk drive. If a cookie **105** is available, the index.HTML file loads the course at the location specified in the cookie **105** as the last place visited and/or bookmarked. If no cookie **105** is available, one is created, and the course content experience begins. The user has a course content experience, and interactive choices and user specific text entries are saved to the client side cookie **105** in process **203**. The client calls, in process **204,** the course Exit file **106** in .HTML format (Exit.HTML). Data cannot be passed, thus the user will pick up again where they left off as long as the cookie **105** is not overwritten and the next course experience is on the same user machine.

Fig. 3 is a flow chart illustrating a method for providing information in accordance with proprietary or non-standards based server access. Data base expansion scripts are provided to create a table in the users existing MSSQL or Oracle database environment. In accordance with this embodiment, the appropriate course Index file in .PHP or .ASP format (Index.PHP or Index.ASP) is linked to a starting page. In process **301**, the program code in the Index.PHP or Index.ASP file tests database access and checks the identification data passed from the proprietary server **101**. If identification data exists, the existing associated data in a "suspend_data" field is written to the client side cookie **105**. If identification data does not exist, a record is created, and a cookie **105** with default values is placed on the user's machine.

In process **302**, in this particular embodiment, program code contained in the Index.HTML file causes checking of the client (or company) link in the cookie **105**. If it has not been set in the cookie **105**, or a different link exists in the cookie **105**, the data in the cookie **105** is updated to reflect the company link in an "includes_js" file. (This particular implementation permits an optional company link, when present, to modify the structure of the content presentation.) The content experience is then launched. The user has a content experience, in process **303**, and interactive choices and user specific text entries are saved to the client side cookie **105**.

In process **304**, the user completes the content experience, and the Exit.HTML file provides code that accesses the cookie **105** to determine the type of initial data pass that was used. The Exit.HTML code then launches the appropriate exit file (either the Exit.PHP file or the Exit.ASP file) and passes the cookie data to that file. The Exit.PHP file or Exit.ASP file also includes code that sends data back to the fields in a table related to the course, and the course exits in process **305**.

Fig. 4 is a flow chart illustrating a method for providing information in accordance with standards-based server access. In accordance with this embodiment, the Index.HTML file is launched **401** from a standards-based server. The Index.HTML file provides program code that checks to determine if communication is possible through the AICC or the SCORM standard and identifies the API appropriate for the task. The Index.HTML also provides program code that tests the data pass and checks the identification data sent from the standards-based server. If a record exists, the Index.HTML file provides program code to receive data and write a cookie **105** to the client computer from a "suspend_data" field or a "core_lesson" field. If the identification data does not exist, the Index.HTML file provides code to create a record and place a cookie **105** with default information on the user's computer. In this embodiment, the Index.HTML file also provides code to check the company or client link. If a link has not been sent to the cookie **105**, or a different link exists in the cookie **105**, the data in the cookie **105** is updated to reflect the company or client link in an "includes_js" file. The content experience is then launched.

The user has a content experience in process **402**, interaction choices and user specific text entries are saved to the client side cookie and content experience information is periodically sent back to the system **105**. In process **403**, the user completes the content experience, and the Exit.HTML file provides program code that searches the cookie **105** to determine the type of initial data pass. The Exit.HTML file also provides code to address the appropriate API to pass data back to the standards-based server.

In a separate embodiment, a course engine uses XML or Winini to tie course information during course creation to the proper AICC and SCORM terms, and generates the same type of files as demonstrated above automatically, giving the course content the ability to self detect its environment and integrate into a clients environment simply.

The apparatus and methods described above, enable those systems with or without a database to provide an identical interactive experience to their users. They also allow the same content and course version to address all client distribution methods, regardless of their proximity to a central database.

There are differences in technical functionality in each implementation scenario, but they are found in every instance of content delivery in the scenario and not specifically a product of the invention.

Traditional Internet or Intranet delivery (Proprietary or Standards based environment)

This is a solution like a traditional hosting solution available via an on-line site.

Features and trade-offs:

(a) The server hosts the course.
(b) The server manages user data.
(c) Reports are easily made available
(d) Data is persistent from session to session.
(e) Data is persistent from computer to computer.

Non-database distribution through Internet/Intranet or CD/hard drive use

This version allows a company or client to run content from a database-free Intranet, Internet, or hard drive installation or a CD distribution.

Features and trade-offs:

(a) The course lives on the client Intranet/Internet or user CD drive/desktop.
(b) Data is user specific and held on user machine.
(c) No completion reporting is possible.
(d) Data is persistent from session to session as long as the cookie is not over-written.
(e) Data is not persistent from computer to computer.

Fig. 5 is an illustration of a graphical user interface for accessing files in accordance with the embodiments of Figs. 1-4. In accordance with this particular embodiment, "Managing Direct Reports" is a course that provides interactivity through an interactive case and short practice activities, along with expert feedback for self-evaluation. An "@Work" section with interactive online tools also helps the user prepare, practice, and review key concepts as they work. The content is written in HTML, using client side JavaScript to go through the steps outlined above. The course package also includes SCORM and AICC files for standards based server integration as well as PHP and ASP files for a proprietary server implementation.

The implementation files included in the package for this embodiment are provided in appropriate subfolders or stored directly in the directReportsFlat folder. Such files include the content files in folders such as **511, 512, 513, 514, 515, 516** and **517** the standard AICC files in folder **501**, the standard SCORM files in folder **502**, MSSQL (folder **503**) and Oracle database scripts for those systems without a standards compliant database system, P3P example files in folder **504** for IE 6.0 privacy settings, and both index (in pages **505, 507** and **509**) and exit files (**506, 508** and **509**) for PHP, ASP and standard HTML respectively.

The index and exit pages **509 and 510** in the package are the most relevant to the invention outlined above. The Index.PHP **505** and the Index.ASP **507** do basically the same thing, but for different environments. These files are designed to allow integration of content to occur in a proprietary (or non-standards compliant) server environment. At the end of each one, there is a pointer to the Index.html file **509** that finishes up the data detection process, and the course experience begins.

At the end of the user experience, the Exit.php **506** and Exit.asp **508** files are called by the Exit.html **510** file upon realization that a proprietary (or non-standards compliant) server passed data into the content at the beginning of the experience. The Index and Exit files and their associated code are also explained on the pages in Appendix A attached hereto.

In a separate embodiment, a course engine may use XML or Winini to tie course information during course creation to the proper AICC and SCORM terms, and generate the same type of files as described in accordance with the embodiment of Fig. 5. This automatically gives the content the ability to self-detect its environment and integrate into a clients environment simply.

Although various exemplary embodiments of the invention have been disclosed, it should be apparent to those skilled in the art that various changes and modifications can be made which will achieve some of the advantages of the invention without departing from the true scope of the invention.

02480/104WO 237626.1

APPENDIX A

Index.PHP

```
<?php
```
Error reporting turned on in order to test that content is connecting properly to database
```
error_reporting(0);
```

Properties of Database are set
```
//HBSP Connect Code
PutEnv("ORACLE_SID=[enter your SID]");
PutEnv("ORACLE_HOME=[enter your Oracle home directory]");
$db = "[enter your Oracle database address]";
$connection = OCILogon ("[enter your Oracle logon]","[enter
your Oracle password]", $db);
```

Connection to database is tested
```
if(!$connection)
{
     $error=OCIError();
}
```

Test to see if student_id was passed via POST or QUERYSTRING method, if not, generate random username to ensure program will run
```
if(!$HTTP_GET_VARS["student_id"] &&
!$HTTP_POST_VARS["student_id"]){
     list($usec, $sec) = explode(' ', microtime());
    mt_srand((float) $sec + ((float) $usec * 100000));
     $randval = mt_rand();
     $username = $randval;
}
```
set variable to student_id passed in
```
else{
     if(!$HTTP_POST_VARS["student_id"]){
          $username = $HTTP_GET_VARS["student_id"];
     }
     else {
          $username = $HTTP_POST_VARS["student_id"];
     }
}
```

If there was no error in test connection with database, connect to database and determine if there is information for user already there - STEP 3
```
if(!$error) {
     //Check if user already exists, if not then add user
     into the DB
     $querySearch = "Select student_id from
     directreports_table where student_id = '$username'";
     $cursor = OCIParse ($connection,$querySearch);
     $result = OCIExecute ($cursor);
```
Test result of select statement to make sure no errors occurred

```
        if (!$result)
        {
             $error = OCIError($cursor);
        }

        while (OCIFetchInto ($cursor, $values))
        {
             $sUser = $values[0];
        }
        $strUserID = $sUser;

        if($strUserID != '')
        {
        }
        if user does not exist, create new entry for them in
        database table
        else{

             //instantiate record
             $startDate = date('Y-m-d');
             $query = "INSERT INTO directreports_table
        (student_id,lesson_status,suspend_data,started) values
        ('$username','not
        attempted','username~$username|mode~php|firstVisit~yes|
        ',to_date('$startDate','YYYY-MM-DD'))";
             $cursor = OCIParse($connection, $query);
             $result = OCIExecute($cursor);
             Test result of insert statement to make sure no
             errors occurred
             if (!$result)
             {
                  $error = OCIError($cursor);
             }
        }


        Obtain user information from database and put into php
        variables - STEP 3
        $query = "select lesson_status, suspend_data from
        directreports_table where student_id='$username'";
        $cursor = OCIParse ($connection, $query);
        $result = OCIExecute ($cursor);

        Test result of select statement to make sure no errors
        occurred
        if (!$result)
        {
             $error = OCIError($cursor);
        }

        while (OCIFetchInto ($cursor, $values))
        {
             $status = $values[0];
             $cookie = $values[1];
        }
}
```

```
Close database connection
OCIFreeStatement($cursor);
OCILogoff($connection);

?>
<html>
<head>
<script language="JavaScript"
src="jscript/scripts.js"></script>
<script language="JavaScript"
src="jscript/includes.js"></script>
</head>
<script language="JavaScript">
     Set client side cookie to hold information obtained
     from database - STEP 4
     setCookie(mainCookieName,"","<?php echo($cookie);
?>","/");
     setCookie(mainCookieName,"startStatus","<?php
echo($status); ?>","/");

     if a database error was detected, provide user with
     error information
     var errTest = "" + <?php echo($error); ?> + "";
     if(errTest != "0") {
          alert("The Learning Management System is
experiencing problems and is unable to retrieve your data.
Please contact your system administrator for support.  You
may continue on with the course but your data may not be
saved when you exit.");
     }
     Send user to standard index.html to proceed with course
- STEP 6
     document.location.href = "index.html";
</script>
</head>

<body bgcolor="#ffffff">
</body>
</html>
```

Index.ASP

```
<%
     Error reporting turned on in order to test that content
     is connecting properly to database
     on Error Resume Next
     Dim errorVal

     Properties of Database are set
     set cnObj=Server.CreateObject("ADODB.Connection")
     cnObj.ConnectionString="Provider=SQLOLEDB.1;Password=[e
nter your SQL Server password];Persist Security
Info=False;User ID=[enter your SQL Server user ID];Initial
Catalog=[enter your SQL Server database name];Data
Source=[enter your SQL Server data source]"
```

```
      Connection to database is tested
      cnObj.Open
      if cnObj.Errors.Count > 0 then
           For Each objError in cnObj.Errors
                errorVal = objError.Number
           Next
      end if

      Test to see if student_id was passed via POST or
      QUERYSTRING method, if not, generate random username to
      ensure program will run
      if Request.QueryString("student_id") = "" AND
Request.FORM("student_id") = "" then
                'generate random username
                Randomize()
                username = Int((100000000000 * Rnd) + 1)

      set variable to student_id passed in
      elseif Request.QueryString("student_id") <> "" then
           username = Request.QueryString("student_id")
      else
           username = Request.Form("student_id")
      end if

      connect to database and determine if there is
      information for user already there - STEP 3
      strSQL = "Select student_id from directreports_table "
_
      & "where student_id = '" & username & "'"

      set rs = cnObj.Execute(strSQL)

      Test result of select statement to make sure no errors
      occurred
      if cnObj.Errors.Count > 0 then
           For Each objError in cnObj.Errors
                errorVal = objError.Number
           Next
      end if

      if user does not exist, create new entry for them in
      database table
      if rs.EOF then
           'new user
           SQLstring = "INSERT INTO directreports_table
(student_id,lesson_status,suspend_data,started)"
           SQLstring = SQLstring & " values ('" & username &
"','not attempted','username~" & username &
"|mode~asp|firstVisit~yes|','" & Date() & "')"
           cnObj.Execute(SQLstring)

           Test result of insert statement to make sure no
           errors occurred

           if cnObj.Errors.Count > 0 then
```

```
                For Each objError in cnObj.Errors
                     errorVal = objError.Number
                Next
           end if
      end if

      **Obtain user information from database and put into asp
      variables - STEP 3**
      strSQL = "Select lesson_status,suspend_data from
directreports_table " _
      & "where student_id = '" & username & "'"

      set rs = cnObj.Execute(strSQL)

      **Test result of select statement to make sure no errors
      occurred**
      if cnObj.Errors.Count > 0 then
           For Each objError in cnObj.Errors
                errorVal = objError.Number
           Next
      end if

      status = rs("lesson_status")
      cookie = rs("suspend_data")
%>
<html>
<head>
<script language="JavaScript"
src="jscript/scripts.js"></script>
<script language="JavaScript"
src="jscript/includes.js"></script>

<script language="JavaScript">
      **if a database error was detected, provide user with
      error information**
      var errTest = "" + <%=errorVal %> + "";
      if (errTest != 0) {
           alert("The Learning Management System is
experiencing problems and is unable to retrieve your data.
Please contact your system administrator for support.  You
may continue on with the course but your data may not be
saved when you exit.");
      }

      **Set client side cookies to hold information obtained
      from database - STEP 4**
      setCookie(mainCookieName,"","<%=cookie %>","/");
      setCookie(mainCookieName,"startStatus","<%=status
%>","/");
      **Send user to standard index.html to proceed with course
- STEP 6**
      document.location.href = "index.html";
</script>
</head>

<body bgcolor="#ffffff">
</body>
```

```
</html>
```

Index.html

```
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Expires" content="-1">
<html>
<head>
<script language="javascript"
src="jscript/APIWrapper.js"></script>
<script language="javascript"
src="jscript/scripts.js"></script>
<script language="javascript"
src="jscript/includes.js"></script>
<script language="javascript">writeTitle();</script>
<script language="JavaScript">
function accessLMS() {
Tests to see if user came from php or asp index page, if
yes, continue on to proceed function, else test to see if
SCORM or AICC API exists - STEP 2
     if(mode == "php" || mode == "asp") {
          proceed();
     }
     else {
          Test to see if SCORM/AICC API exists, if not, set
          mode variable to empty string and continue on to
          proceed function, else access the API for user
          information - STEP 2
          var result = doLMSInitialize();
          if (result == false) {
               mode = "";
               proceed();
          }
          else {
               Set mode variable to "scorm"
               mode = "scorm";

               Check user's lesson status value from
               SCORM/AICC API, - STEP 3
               var status = "" + doLMSGetValue(
"cmi.core.lesson_status" ) + "";
               if (status == "") {
                    if no status is returned, provide user
                    with error information
                    alert("The Learning Management System is
experiencing problems and is unable to retrieve your data.
Please contact your system administrator for support.  You
may continue on with the course but your data may not be
saved when you exit.");
               }
               if (status == "not attempted") {
                    if user has not attempted course, change
                    status in API to "incomplete" as user is
                    now starting course, also clear any
                    residual values from cookie - STEP 3
                    // the student is now attempting the
lesson
```

```
                    setCookie(mainCookieName,"","","/");
                    doLMSSetValue( "cmi.core.lesson_status",
"incomplete" );
               }
               else {
                    if user has already started course, get
                    Cookie info from API (suspend_data
                    field) and put info into main client-
                    side cookie - STEP 4
                    LMSCookieInfo = "" +
doLMSGetValue("cmi.suspend_data") + "";
                    setCookie(mainCookieName, "",
LMSCookieInfo, "/");
               }
               Set mode variable in cookie to "scorm"
               setCookie(mainCookieName,"mode","scorm","/");
               proceed();
          }
     }
}

function proceed() {
Test to see if mode already set to php,asp,or scorm.  If
not set to any of these, check querystring to see if AICC
HACP information was passed through. - STEP 2
     if (mode != "php" && mode != "asp" && mode != "scorm")
{
          if (args.AICC_SID) {
               set mode variable to "aicc",
               mode="aicc";
          }
     }
}

The finishOpen function runs after the body of the page is
loaded.  This makes sure that the AICC Java Applet, if
written to the page, has fully loaded
function finishOpen() {
     if mode was set to aicc earlier, use the Java applet to
     communicate with the server via the HACP method - STEP
     3
     if(mode=="aicc") {
          if (args.AICC_URL.indexOf(':',0,7) == -1) {
               args.AICC_URL = "http://" + args.AICC_URL;
          }
          body.document.aiccPoster.post(args.AICC_URL,"GetPa
ram", args.AICC_SID,"2.0","");
          Check to make sure that no AICC error was returned
- STEP 3
          var success = "" +
     body.document.aiccPoster.getErrorCode() + "";
          if(success == "0") {
               If no AICC error is returned, check user's
               lesson status value returned from GetParam -
               STEP 3
               status =
body.document.aiccPoster.getLessonStatus();
```

```
                status = status.toLowerCase();
                statusIndicator = status.substr(0,1);
                if (statusIndicator == "n") {
                     if user has not attempted course, change
                     status in LMS to "incomplete" as user is
                     now starting course, also clear any
                     residual values from cookie - STEP 3
                     // the student is now attempting the
lesson
                     setCookie(mainCookieName,"","","/");
                     var aicc_data = "[core]\r\nlesson_status
= incomplete\r\nlesson_location = 0\r\nscore = \r\ntime =
00:00:00";
                     aicc_data = escape(aicc_data);

     body.document.aiccPoster.post(args.AICC_URL,
"PutParam",args.AICC_SID,"2.0",aicc_data);
                }
                else {
                     if user has already started course, get
                     Cookie info from GetParam command
                     ([core_lesson] field) and put info into
                     main client-side cookie - STEP 4
                     LMSCookieInfo =
body.document.aiccPoster.getLessonData();
                     setCookie(mainCookieName, "",
LMSCookieInfo, "/");
                }
           }
           else {
                if an AICC error is returned, provide user
                with error information
                alert("The Learning Management System is
experiencing problems and is unable to retrieve your data.
Please contact your system administrator for support.  You
may continue on with the course but your data may not be
saved when you exit.");
           }
           Input AICC querystring values into main client
           side cookie - STEP 4
           setCookie(mainCookieName,"mode","aicc","/");

           setCookie(mainCookieName,"aicc_sid",args.AICC_SID,
     "/");

           setCookie(mainCookieName,"aicc_url",args.AICC_URL,
     "/");
           }

Set currentCompanyLink value of client-side cookie to
javascript companyLink variable set in jscript/includes.js
file
     setCookie(mainCookieName, "currentCompanyLink",
companyLink,"/");
Make sure student score has a numerical zero value if
previously empty.
     var studentScore =
getCookieCrumb(mainCookieName,"user_score");
     //zero out score
     if(studentScore=="void"){
          setCookie(mainCookieName,"user_score","0","/");
     }
```
Send user to home page for the course - STEP 6
```
     body.location.href="dr_home.html";
}
```

Function to set querystring values into an array
```
function getArgs() {
     var args = new Object();
     var query = parent.location.search.substring(1);
     var pairs = query.split("&");
     for(var i=0; i<pairs.length; i++) {
          var pos = pairs[i].indexOf('=');
          if(pos == -1) {
               continue;
          }
          var argname = pairs[i].substring(0,pos);
          argname = argname.toUpperCase();
          var value = pairs[i].substring(pos+1);
          args[argname] = unescape(value);
     }
     return args;
}
```

sets mode variable to current mode in cookie - STEP 2
```
var mode = getCookieCrumb(mainCookieName,"mode");
```
Get querystring arguments and put into args array cookie - STEP 3
```
var args = getArgs();
```
execute initial accessLMS function before page has finished loading to determine what method of datapass is being used. - STEP 2
```
accessLMS();
</script>
</head>

<frameset cols="155,585" rows="*" border="0"
frameborder="NO">
  <frame src="dr_rail.html" scrolling="NO" name="rail">
  <frameset rows="80,420" cols="*" border="0"
frameborder="NO">
    <frame src="dr_navtop.html" scrolling="NO"
name="navigation">
    <frame src="dr_home.html" scrolling="AUTO" name="body">
  </frameset>
</frameset>
<noframes>
</noframes>
</html>
```

startlms.html

startlms.html page is called in the frameset of the

index.html page, and is used to load the AICC Java Applet, if necessary - STEP 2

```
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Expires" content="-1">
<html>
<head>
     <title></title>
<script language="javascript">
```

writeApplet function used to output html code for Java Applet if AICC HACP method is used. Java Applet contains all functions necessary to communicate with the LMS via the AICC HACP method.

```
function writeApplet() {
     document.write("<applet code=\"AICCApplet.class\"
codebase=\"classes\" name=\"aiccPoster\" id=\"aiccPoster\"
archive=\"hbspaicc.jar\" width=\"1\" height=\"1\"
mayscript>");
     document.write("<param name=\"cabbase\"
value=\"hbspaicc.cab\">");
     document.write("</applet>");
}
</script>
</head>
```

body onload event calls finishOpen function on index.html page once entire startlms.html page is loaded. This ensures that if AICC Java Applet is needed, it is fully loaded before its functions are accessed.

```
<body onload="parent.finishOpen();">
<script language="javascript">
```

If mode variable on index.html page was set to "aicc", and the AICC querystring arguments are present, call the writeApplet function to create the instance of the AICC Java Applet. If the mode variable is set to "aicc", but no querystring arguments are present, variable is a residual value, and mode variable is set to "none" - STEP 2,3

```
     if(parent.mode == "aicc" && parent.args.AICC_SID) {
          writeApplet();
     }
     else if(parent.mode == "aicc") {

     parent.setCookie(parent.mainCookieName,"mode","none","/
");
          parent.mode = "none";
     }
</script>
</body>
</html>
```

Exit.html

```
<html>
<head>
<script language="javascript"
src="jscript/includes.js"></script>
```

```
<script language="javascript"
src="jscript/scripts.js"></script>
<script language="javascript"
src="jscript/APIWrapper.js"></script>
<script language="javascript"
src="jscript/close.js"></script>
<script language="javascript">writeTitle();</script>
</head>
```

When page has loaded, body onLoad event fires off, executing exitToLMS function in close.js file - STEP 8

```
<body bgcolor="#ffffff" onLoad="exitToLMS();">
```

If mode variable on close.js file was set to "aicc" (obtained from the cookie), call the writeApplet function to create the instance of the AICC Java Applet.

```
<script language="javascript">
     if(mode == "aicc") {
          writeApplet('');
     }
</script>
</body>
</html>
```

Close.js

Retrieve method of datapass from cookie and set into mode variable - STEP 9

```
var mode = getCookieCrumb(mainCookieName,"mode");

function exitToLMS(){
```

executes closeLMS function (below)

```
     closeLMS();
}
```

closeProgram function loads closer.html file which subsequently closes top level window - STEP 11

```
function closeProgram() {
     document.location.replace("closer.html");
}

function closeLMS() {
```

puts cookie info for score and course data into variables

```
     var courseScore =
getCookieCrumb(mainCookieName,"user_score");
     var cookieToPass = getCookieData(mainCookieName);
```

sets courseStatus variable based on user score

```
     if (courseScore >= 70) {
          var courseStatus = "completed";
     }
     else {
          var courseStatus = "incomplete";
     }
```

```
Check to see if mode is SCORM, if so, test to see if API
for SCORM/AICC exists, if so, send the cookie data to the
LMS - STEP 9, 10
     if (mode == "scorm") {
          if (LMSIsInitialized()) {
               doLMSSetValue("cmi.core.lesson_status",
          courseStatus);

               only set score.raw value if course status is
          completed
               if(courseStatus == "completed") {
                    doLMSSetValue("cmi.core.score.raw",
          courseScore);
               }

               Test for errors in connection with LMS while
          sending suspend_data.  If successful, clear
          cookie.  If there are errors, do not clear cookie
          and report error message to user.
               var success =
          doLMSSetValue("cmi.suspend_data", cookieToPass);
               if(success) {
                    setCookie(mainCookieName,"","","/");
               }
               else {
                    alert("The Learning Management System is
          experiencing problems and is unable to save your
          data.  Please contact your system administrator
          for support.  You may wish to print out any
          information you wish to save.");
                    return;
               }

               execute doLMSFinish function which closes out
          the API
               doLMSFinish();

               execute closeProgram function to close course
          window down. - STEP 11
               closeProgram();
          }
     }

check to see if mode is aicc, if so, run hacp commands to
post user completion data - STEP 9, 10
     else if(mode == "aicc") {
          //code to fire off aicc commands

          get aicc values from cookie
          aicc_url =
getCookieCrumb(mainCookieName,"aicc_url");
          aicc_sid =
getCookieCrumb(mainCookieName,"aicc_sid");

          Test status of course and set variable to hold
          aicc data to send to LMS, do not send score back
          unless course is complete
          if(courseStatus == "completed") {
               aicc_data = "[core]\r\nlesson_status = " +
          courseStatus + "\r\nlesson_location = 0\r\nscore =
          " + courseScore + "\r\ntime =
          00:00:00\r\n[core_lesson]\r\n" +
          unescape(cookieToPass) + "\r\n";
          }
          else {
               aicc_data = "[core]\r\nlesson_status = " +
          courseStatus + "\r\nlesson_location = 0\r\nscore =
          \r\ntime = 00:00:00\r\n[core_lesson]\r\n" +
          unescape(cookieToPass) + "\r\n";
          }

          aicc_data = escape(aicc_data);

          submit aicc data back to LMS via the Java Applet
          document.aiccPoster.post(aicc_url,
"PutParam",aicc_sid,"2.0",aicc_data);

          Test for errors in connection with LMS while
          sending aicc_data.  If successful, clear cookie.
          If there are errors, do not clear cookie and
          report error message to user.
          var success = "" +
          document.aiccPoster.getErrorCode() + "";
          if(success == "0") {
               setCookie(mainCookieName,"","","/");
          }
          else {
               alert("The Learning Management System is
experiencing problems and is unable to save your data.
Please contact your system administrator for support.  You
may wish to print out any information you wish to save.");
               return;
          }

          execute ExitAU function which closes out the HACP
          connection
          document.aiccPoster.post(aicc_url,"ExitAU",aicc_si
          d,"2.0","");
          execute closeProgram function to close course
          window down. - STEP 11
          closeProgram();
     }

     if the mode was php or asp, send the user to the
exit.php or exit.asp page with variables from cookie passed
in querystring - STEP 9
     else if((mode == "php") || (mode == "asp")) {
          var userName =
getCookieCrumb(mainCookieName,"username");
          var prevStatus =
escape(getCookieCrumb(mainCookieName,"startStatus"));
          document.location="exit." + mode + "?username=" +
userName + "&score=" + courseScore + "&status=" +
courseStatus +"&prevStatus=" + prevStatus;
```

```
          }
          else {
               closeProgram(); - STEP 11
          }
     }
```

scripts.js

Only code pertaining to the exit.html page is included here from the scripts.js file

function LMSIsInitialized tests to see if SCORM/AICC API is initialized

```
function LMSIsInitialized() {
     var initialized = getAPI();
     if (initialized == null) {
          return false;
     }
     else {
          return true;
     }
}
```

writeApplet function used to output html code for Java Applet if AICC HACP method is used. Java Applet contains all functions necessary to communicate wit the LMS via the AICC HACP method.

```
function writeApplet(addPath) {
     document.write("<applet code=\"AICCApplet.class\"
codebase=\"" + addPath + "classes\" name=\"aiccPoster\"
id=\"aiccPoster\" archive=\"hbspaicc.jar\" width=\"1\"
height=\"1\" mayscript>");
     document.write("<param name=\"cabbase\"
value=\"hbspaicc.cab\">");
     document.write("</applet>");
}
```

Exit.PHP

```
<?php
```

Error reporting turned on in order to test that content is connecting properly to database

```
error_reporting(0);
```

Properties of Database are set

```
//HBSP Connect Code
PutEnv("ORACLE_SID=[enter your SID]");
PutEnv("ORACLE_HOME=[enter your Oracle home directory]");
$db = "[enter your Oracle database address]";
$connection = OCILogon ("[enter your Oracle logon]","[enter
your Oracle password]", $db);
```

Connection to database is tested

```
if(!$connection)
```

```
{
     $error=OCIError();
}
```

Variables set to hold values sent in QueryString

```
$username = $HTTP_GET_VARS["username"];
$score = $HTTP_GET_VARS["score"];
$status = $HTTP_GET_VARS["status"];
$prevStatus = $HTTP_GET_VARS["prevStatus"];
$score=$score/100;
```

Variables set to hold values from cookies

```
//read cookie, put into db
$cookie =
preg_replace("/\r\n|\n\r|\n|\r/","%0A",$directReports);
```

Values from cookies are put into database table - STEP 10

```
if(!$error)
{
     //write to db
     $query = "UPDATE directreports_table SET
     lesson_status='".$status."', suspend_data='".$cookie."'
     ";
     $query = $query."WHERE student_id = '".$username."'";
     $cursor = OCIParse($connection, $query);
     $result = OCIExecute($cursor);
```

Test result of update statement to make sure no errors occurred

```
     if (!$result)
     {
          $error = OCIError($cursor);
     }
```

Score is updated in table if it is at least 70 (completion) - STEP 10

```
     if($score >= .70) {
          $query = "UPDATE directreports_table SET
     score_raw='".$score."' ";
          $query = $query."WHERE student_id =
     '".$username."'";
          $cursor = OCIParse($connection, $query);
          $result = OCIExecute($cursor);
```

Test result of update statement to make sure no errors occurred

```
          if (!$result)
          {
               $error = OCIError($cursor);
          }
     }
```

Completion date is set in db if course was just completed - STEP 10

```
     if($prevStatus != "completed" && $status ==
     "completed") {
          $endDate = date('Y-m-d');
          $query = "UPDATE directreports_table SET
     completed=to_date('$endDate','YYYY-MM-DD') ";
          $query = $query."WHERE student_id =
     '".$username."'";
          $cursor = OCIParse($connection, $query);
          $result = OCIExecute($cursor);
```

Test result of update statement to make sure no errors occurred

```
          if (!$result)
          {
               $error = OCIError($cursor);
          }
     }
}

//close up shop
OCIFreeStatement($cursor);
OCILogoff($connection);

?>
<html>
<head>
<script language="JavaScript"
src="jscript/scripts.js"></script>
<script language="javascript">
```

Test for errors in connection with database while sending course data. If successful, clear cookie. If there are errors, do not clear cookie and report error message to user.

```
var errTest = "" + <?php echo($error); ?> + "";
if(errTest != "0") {
     alert("The Learning Management System is experiencing
problems and is unable to save your data.  Please contact
your system administrator for support.  You may wish to
print out any information you wish to save.");
}
else {
     document.write("Data saved.  You must restart the
course from your LMS to have access to your data again.");
     setCookie(mainCookieName,"","","/");
```

Window Closes - STEP 11

```
     parent.close();
}
</script>
</head>

<body bgcolor="#ffffff">
</body>
</html>
```

Exit.ASP

```
<%
```

Error reporting turned on in order to test that content is connecting properly to database

```
      on Error Resume Next
      Dim errorVal

      Properties of Database are set
      set cnObj=Server.CreateObject("ADODB.Connection")
      cnObj.ConnectionString="Provider=SQLOLEDB.1;Password=[e
nter your SQL Server password];Persist Security
Info=False;User ID=[enter your SQL Server user ID];Initial
Catalog=[enter your SQL Server database name];Data
Source=[enter your SQL Server data source]"

      Connection to database is tested
      cnObj.Open
      if cnObj.Errors.Count > 0 then
           For Each objError in cnObj.Errors
                errorVal = objError.Number
           Next
      end if

      Variables set to hold values sent in QueryString
      username = Request.QueryString("username")
      score = Request.QueryString("score")
      status = Request.QueryString("status")
      prevStatus = Request.QueryString("prevStatus")
      score=score/100

      Variables set to hold value from cookie
      'read cookies
      cookie = Request.Cookies("directReports")
      set objRegExp = new RegExp
      objRegExp.Pattern = "\r\n|\n\r|\n|\r"
      objRegExp.Global = True
      cookie = objRegExp.Replace(cookie,"%0A")

      Values from cookies are put into database table - STEP
10
      'update table with cookie values (for lesson_status,
suspend_data)
      SQLstring = "UPDATE directreports_table SET
lesson_status='" & status & "', suspend_data='" & cookie &
"' "
      SQLstring = SQLstring & "WHERE student_id = '" &
username & "'"
      cnObj.Execute(SQLstring)

      Test result of update statement to make sure no errors
      occurred
      if cnObj.Errors.Count > 0 then
           For Each objError in cnObj.Errors
                errorVal = objError.Number
           Next
      end if


      Score is updated in table if it is at least 70
      (completion) - STEP 10
      if score >= .70 then
```

```
          'update table with score_raw
          SQLstring = "UPDATE directreports_table SET
score_raw='" & score & "' "
          SQLstring = SQLstring & "WHERE student_id = '" &
username & "'"
          cnObj.Execute(SQLstring)

          Test result of update statement to make sure no
          errors occurred
          if cnObj.Errors.Count > 0 then
               For Each objError in cnObj.Errors
                    errorVal = objError.Number
               Next
          end if
     end if

     Completion date is set in db if course was just
     completed - STEP 10
     if NOT prevStatus = "completed" AND status =
"completed" then
          'write to db
          SQLstring = "UPDATE directreports_table SET
completed='" & Date() & "' "
          SQLstring = SQLstring & "WHERE student_id = '" &
username & "'"
          set rs = cnObj.Execute(SQLstring)

          Test result of update statement to make sure no
          errors occurred
          if cnObj.Errors.Count > 0 then
               For Each objError in cnObj.Errors
                    errorVal = objError.Number
               Next
          end if
     end If

     %>
<html>
<head>
<script language="JavaScript"
src="jscript/scripts.js"></script>
<script language="javascript">
Test for errors in connection with database while sending
course data.  If successful, clear cookie.  If there are
errors, do not clear cookie and report error message to
user.
var errTest = "" + <%=errorVal %> + "";
if(errTest != "0") {
     alert("The Learning Management System is experiencing
problems and is unable to save your data.  Please contact
your system administrator for support.  You may wish to
print out any information you wish to save.");
}
else {
     setCookie(mainCookieName,"","","/");
     document.write("Data saved.  You must restart the
course from your LMS to have access to your data again.");
```

```
      Window Closes - STEP 11
      parent.close();
}
</script>
</head>

<body bgcolor="#ffffff">
</body>
</html>
```

02480/104WO 237626.1

What is claimed is:

1. A computer product for providing interactive content for use in a computer system that optionally includes a network of computers, the computer product comprising:

a digital storage medium, the storage medium encoded with computer readable files as follows:

a first set of files providing interactive content using a markup language format and a script format, the scripting language for controlling data communication between a user and a program source;

a second set of files providing compatibility of the interactive content for use with a first e-learning standard environment; and

a third set of files providing compatibility of the interactive content for use with a second e-learning standard environment.

2. A computer product according to claim 1, further comprising a fourth set of files, which may be included in any of the other sets of files, for identifying the nature of an e-learning environment in which interactive content has been placed, and invoking processes for providing data pass consistent with the thus identified e-learning environment, or providing no data pass at all.

3. A computer product according to claim 1, wherein the first e-learning standard environment may comply with AICC standards.

4. A computer product according to claim 1, wherein the second e-learning standard environment may comply with SCORM standards.

5. A computer product according to claim 1, wherein the scripting language is selected from the group consisting of JavaScript, ASP script, and PHP script.

6. A method of providing interactive content, for use in a computer system that optionally includes a network of computers, comprising:

providing interactive content using a markup language format;

providing a first set of files providing compatibility of the interactive content for use with a first e-learning standard environment and a second set of files providing compatibility of the interactive content for use with a second e-learning standard environment;

identifying the nature of an e-learning environment in which interactive content has been placed;

using a cookie, for each user experiencing the interactive content, to provide a record of the user's place in experiencing of the interactive content.

7. A method according to claim 6, wherein the cookie provides a record of the user's data in experiencing of the interactive content.

8. A method according to claim 6, wherein the interactive content is provided in an environment devoid of a network of computers.

02480/104WO 237626.1

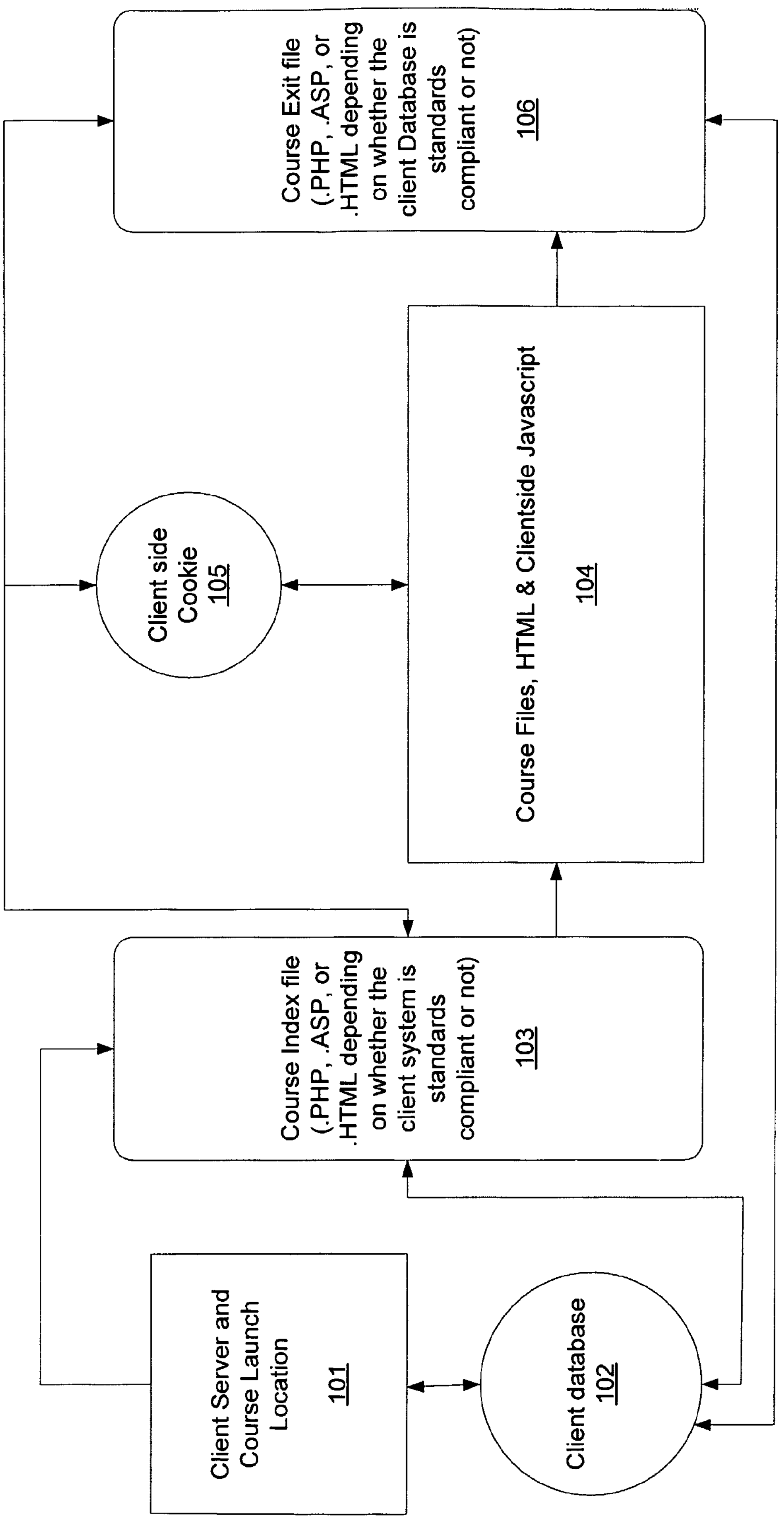

Client Server and Course Launch Location 101
Client database 102
Course Index file (.PHP, .ASP, or .HTML depending on whether the client system is standards compliant or not) 103
Course Files, HTML & Clientside Javascript 104
Client side Cookie 105
Course Exit file (.PHP, .ASP, or .HTML depending on whether the client Database is standards compliant or not) 106

Fig. 1

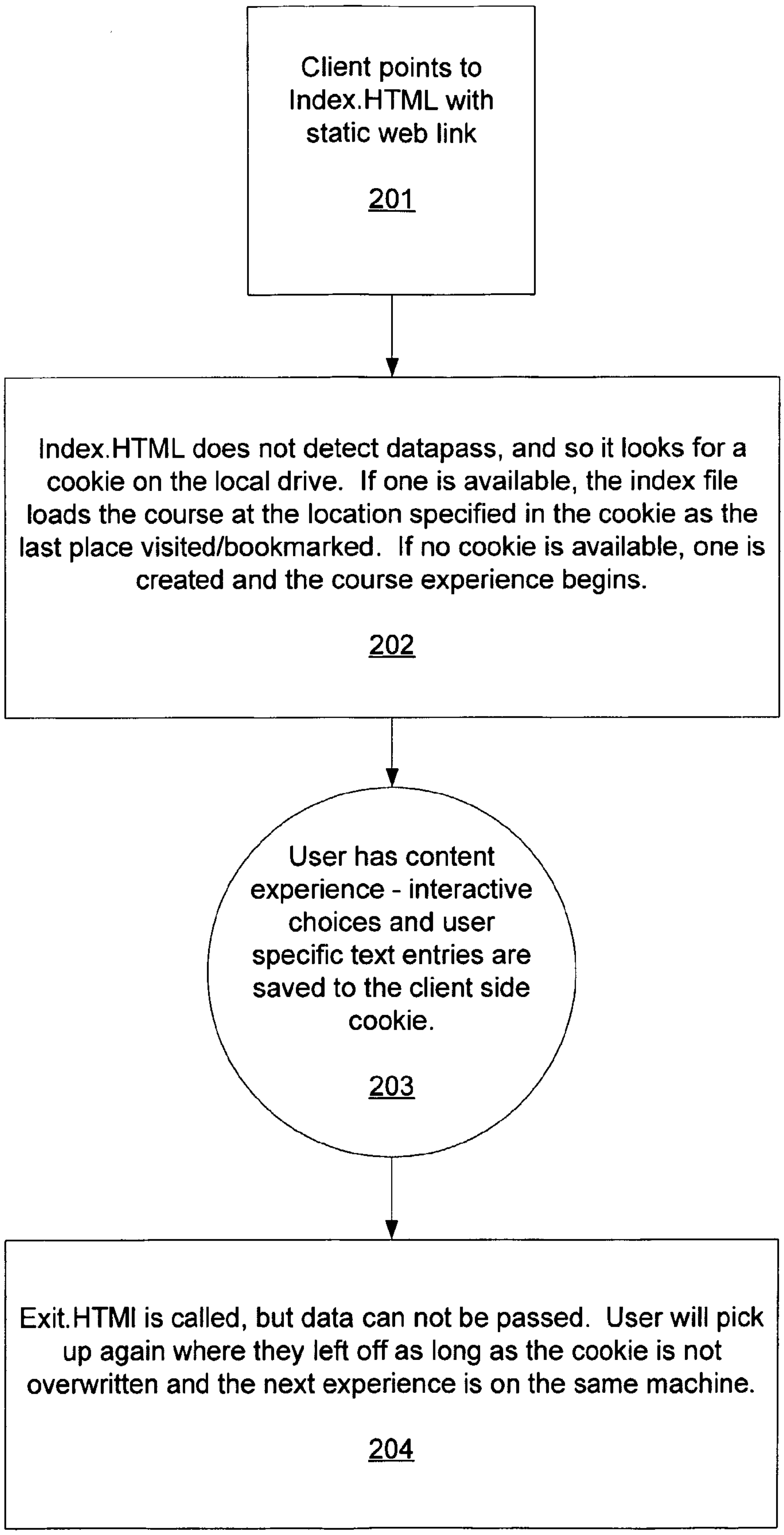

Client points to Index.HTML with static web link
201
Index.HTML does not detect datapass, and so it looks for a cookie on the local drive. If one is available, the index file loads the course at the location specified in the cookie as the last place visited/bookmarked. If no cookie is available, one is created and the course experience begins.
202
User has content experience - interactive choices and user specific text entries are saved to the client side cookie.
203
Exit.HTMl is called, but data can not be passed. User will pick up again where they left off as long as the cookie is not overwritten and the next experience is on the same machine.
204

Fig. 2

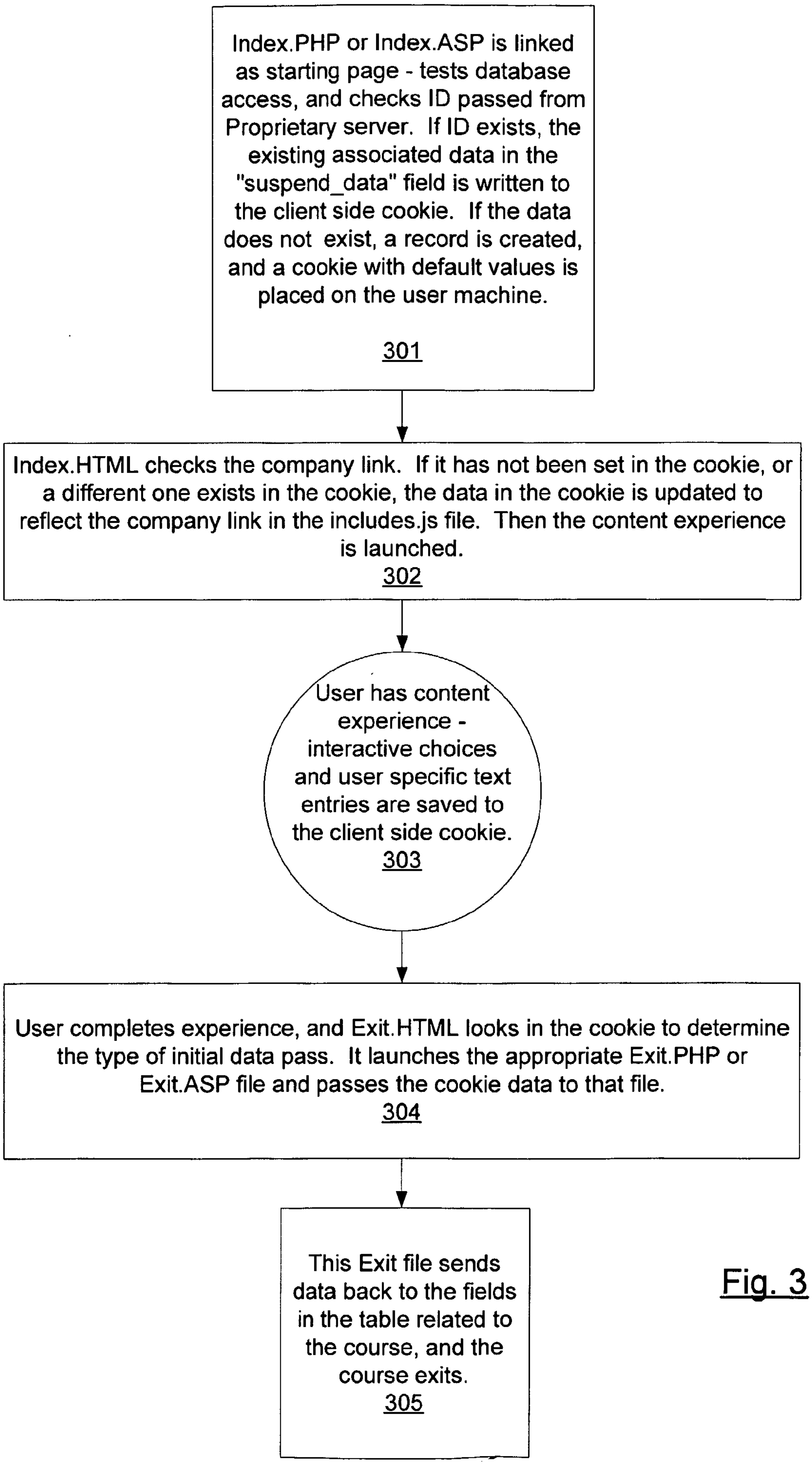

Index.PHP or Index.ASP is linked as starting page - tests database access, and checks ID passed from Proprietary server. If ID exists, the existing associated data in the "suspend_data" field is written to the client side cookie. If the data does not exist, a record is created, and a cookie with default values is placed on the user machine.
301
Index.HTML checks the company link. If it has not been set in the cookie, or a different one exists in the cookie, the data in the cookie is updated to reflect the company link in the includes.js file. Then the content experience is launched.
302
User has content experience - interactive choices and user specific text entries are saved to the client side cookie.
303
User completes experience, and Exit.HTML looks in the cookie to determine the type of initial data pass. It launches the appropriate Exit.PHP or Exit.ASP file and passes the cookie data to that file.
304
This Exit file sends data back to the fields in the table related to the course, and the course exits.
305

Fig. 3

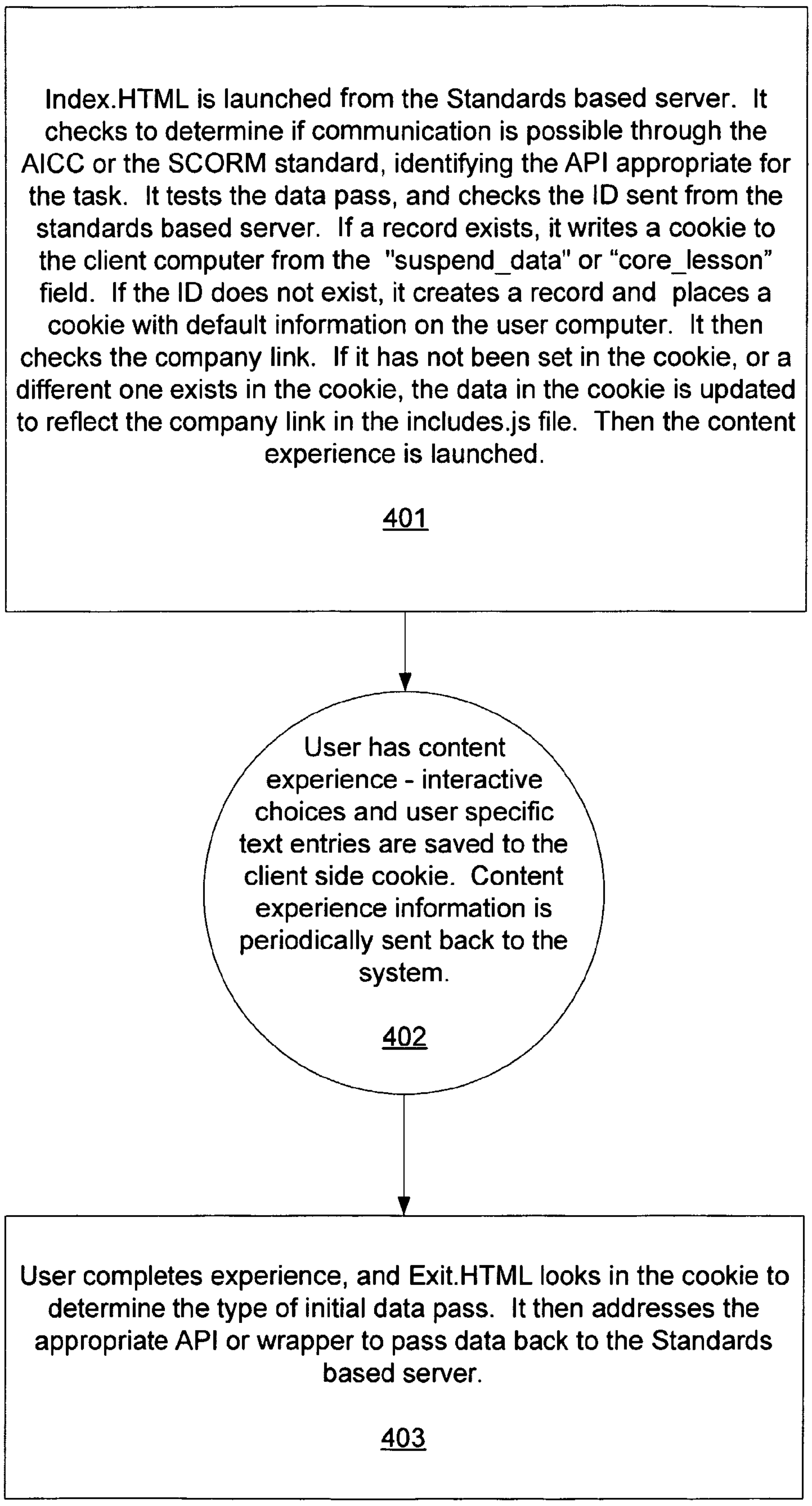
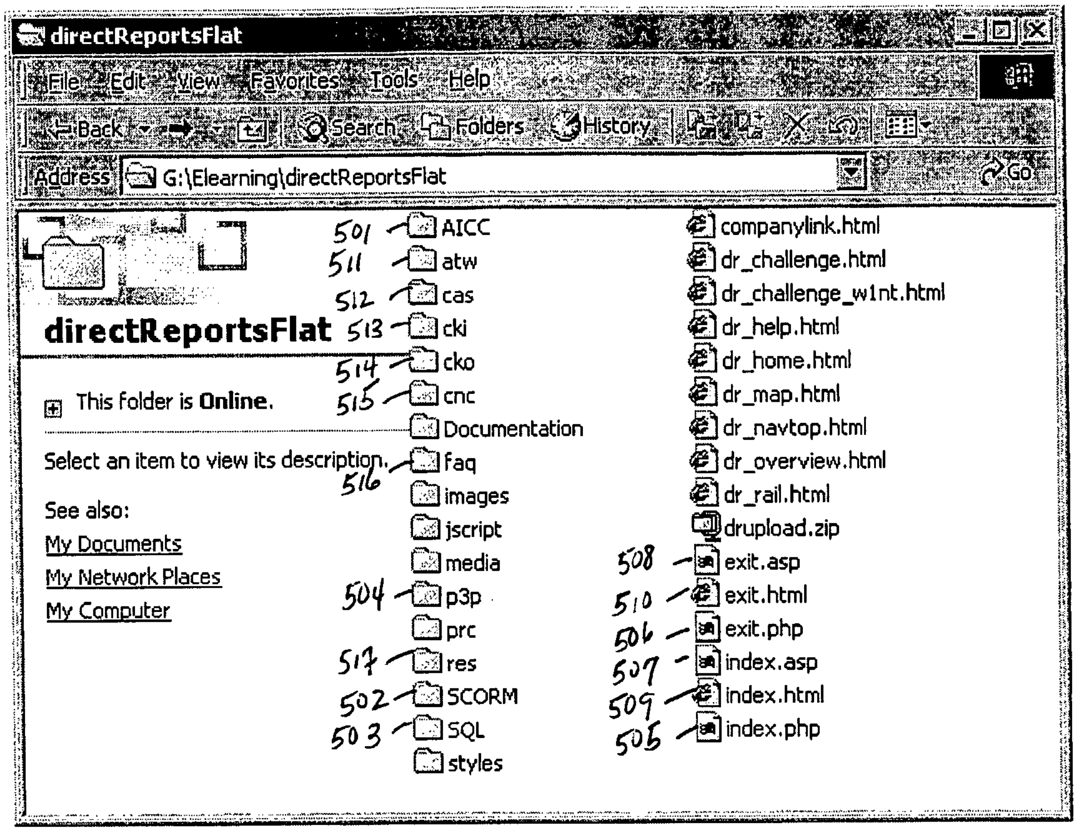

Index.HTML is launched from the Standards based server. It checks to determine if communication is possible through the AICC or the SCORM standard, identifying the API appropriate for the task. It tests the data pass, and checks the ID sent from the standards based server. If a record exists, it writes a cookie to the client computer from the "suspend_data" or "core_lesson" field. If the ID does not exist, it creates a record and places a cookie with default information on the user computer. It then checks the company link. If it has not been set in the cookie, or a different one exists in the cookie, the data in the cookie is updated to reflect the company link in the includes.js file. Then the content experience is launched.
401
User has content experience - interactive choices and user specific text entries are saved to the client side cookie. Content experience information is periodically sent back to the system.
402
User completes experience, and Exit.HTML looks in the cookie to determine the type of initial data pass. It then addresses the appropriate API or wrapper to pass data back to the Standards based server.
403

Fig. 4


directReportsFlat
File Edit View Favorites Tools Help
Back Search Folders History
Address G:\Elearning\directReportsFlat
Go

directReportsFlat

This folder is **Online**.

Select an item to view its description.

See also:
My Documents
My Network Places
My Computer

501 AICC
511 atw
512 cas
513 cki
514 cko
515 cnc
Documentation
516 faq
images
jscript
media
504 p3p
prc
517 res
502 SCORM
503 SQL
styles

companylink.html
dr_challenge.html
dr_challenge_w1nt.html
dr_help.html
dr_home.html
dr_map.html
dr_navtop.html
dr_overview.html
dr_rail.html
drupload.zip
508 exit.asp
510 exit.html
506 exit.php
507 index.asp
509 index.html
505 index.php

Fig. 5