Content Manager OnDemand for Multiplatforms
Version 8 Release 5

Web Enablement Kit Implementation Guide

SC19-2941-00

Content Manager OnDemand for Multiplatforms
Version 8 Release 5

Web Enablement Kit Implementation Guide

SC19-2941-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 173.

This edition applies to IBM Content Manager OnDemand for Multiplatforms version 8, release 5 (product number 5724-J33) and to all subsequent releases and modifications until otherwise indicated in new editions.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

Contents

About this publication

This publication provides information that you can use to plan for, install, configure, and use the IBM® Content Manager OnDemand for Multiplatforms (Content Manager OnDemand) Web Enablement Kit.

Who should use this publication

This publication is for administrators that need to configure ODWEK for an organization.

How this publication is organized

This publication provides the information that you need to install and configure the Content Manager OnDemand Web Enablement Kit (ODWEK) and to configure the Web server software so that users can access data from an OnDemand server with a Web browser. This publication contains the following sections:

- Chapter 1, "Product overview," on page 1 provides general information about the ODWEK system, programming interfaces, viewers, and functions and server and data security.
- Chapter 2, "Implementation overview," on page 11 provides important information about the installation and configuration process and contains an implementation checklist.
- Chapter 3, "Installation requirements," on page 13 lists the software requirements, disk space requirements, and transform requirements.
- Chapter 4, "Installing ODWEK," on page 15 describes how to install the ODWEK software on the system.
- Chapter 5, "Deploying and programming Java API," on page 21 describes how to set up the system environment and run Java on an ODWEK application. This section is also a guide to programming with the Java API and contains examples.
- Chapter 6, "Java API reference," on page 67 explains where to locate reference information for programming with the Java API.
- Chapter 7, "Configuring the sample applications," on page 69 explains how to customize the sample applications that are provided with ODWEK.
- Chapter 8, "Installing and configuring viewing software," on page 73 describes the viewers, lists the requirements for the viewers, and describes how to install, configure, and distribute the viewer software.
- Chapter 9, "Verifying the installation," on page 85 explains how to verify the installation, contains troubleshooting help, and provides a road map to information that you may need after you have finished installing ODWEK.
- Appendix A, "Deploying the CGI program," on page 89 contains reference information for programming with the CGI version of ODWEK.
- Appendix B, "Deploying the Java servlet," on page 149 describes how to deploy the Java servlet on the system.
- Appendix C, "No HTML output," on page 155 describes the delimited ASCII output that can be generated by ODWEK.
- Appendix D, "National language support," on page 159 provides information to help configure the system for DBCS languages.

- Appendix E, “Problem determination tools,” on page 163 lists the tools that you can use to gather information about the system and documents to aid in troubleshooting.

What you should already know

The documentation for ODWEK assumes the following:

- You understand the Internet, Web servers and browsers, Transmission Control Protocol/Internet Protocol (TCP/IP) networking, and OnDemand.
- You are familiar with Hypertext Markup Language (HTML), Common Gateway Interface (CGI) and Java programming.
- You provide content for Web pages.
- You know how to configure and operate an Hypertext Transfer Protocol (HTTP) server, a Java-enabled Web server, and a Java application server.
- You can administer an OnDemand server.

Additional resources

The following three articles provide additional information on using ODWEK:

- *Best practices for building Web Applications using IBM Content Manager OnDemand Web Enablement Kit*: http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101203
- *Best practices for AFP Resource caching in a ODWEK Java API application*: http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101247
- *Debugging a Custom ODWEK Java API Web Application*:http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101248

ibm.com and related resources

Product support and documentation are available from ibm.com®.

Support and assistance

Product support is available on the Web. Click Support from the product Web site at:

Content Manager OnDemand for Multiplatforms
www.ibm.com/software/data/ondemand/mp/support.html

Information Center

You can view the product documentation in an Eclipse-based information center that you can install when you install the product. By default, the information center runs in a Web server mode that other Web browsers can access. You can also run it locally on your workstation. See the information center at: www.ibm.com/software/data/ondemand/mp/support.html

PDF publications

You can view the PDF files online using the Adobe® Acrobat Reader for your operating system. If you do not have Acrobat Reader installed, you can download it from the Adobe Web site at www.adobe.com.

You can find PDF publications for IBM Content Manager OnDemand for Multiplatforms at: http://www.ibm.com/support/docview.wss?rs=129&uid=swg27012713

Accessibility information for OnDemand

For complete information about accessibility features that are supported by this product, see your Content Manager OnDemand *Administration Guide*.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this publication or other OnDemand documentation. Visit the IBM Data Management Online Reader's Comment Form (RCF) page at www.ibm.com/software/data/rcf.

Be sure to include the name of the product, the version number of the product, and the name of the book. If you are commenting on specific text, please include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

If you would like to help IBM make IBM Content Manager OnDemand for Multiplatforms easier to use, take the consumability survey at http://www.ibm.com/software/data/info/consumability-survey/.

What's new in Version 8.5

Lightweight Directory Access Protocol (LDAP) Secure Sockets Layer (SSL) support
You can now use LDAP with SSL.

Report Distribution enhancements
IBM Content Manager OnDemand Report Distribution for Multiplatforms has been enhanced to allow support for additional data type conversion engines, that is, from line data to PDF.

Enhanced reporting and analysis for managing the Content Manager OnDemand system
Content Manager OnDemand now provides ability to analyze system log data and generate daily activity reports.

Improved capability and productivity for indexing PDF data into Content Manager OnDemand
You can now index PDF documents using metadata values contained within the PDF document.

Expanded security capabilities to better comply with Federal Information Processing Standards (FIPS 140-2)
Content Manager OnDemand has added additional security to better comply with FIPS 140-2.

Additional language support
The Content Manager OnDemand server and ODWEK installation programs are translated into 9 additional languages. The Content Manager OnDemand client installation program is translated into 22 additional languages. The Content Manager OnDemand Configurator is translated into 9 additional languages.

HP-UX Itanium support for DB2 and Oracle databases
Content Manager OnDemand now supports HP-UX Itanium for DB2 and Oracle databases.

ARSXML system administration performance improvement
Performance improvements have been made to the ARSXML system administration command.

Install Anywhere replacing ISMP for servers
InstallAnywhere 2009 SP1 has replaced InstallShield Multiplatform (ISMP) as the installer engine for Content Manager OnDemand server and ODWEK on Windows and Unix platforms.

Chapter 1. Product overview

ODWEK allows users to access data that is stored in an IBM Content Manager OnDemand server by using a Web browser or a user-written program. For example, you can provide some users with the Uniform Resource Locator (URL) of a Web page that permits them to log on to an OnDemand server; you can provide other users with the URL of a Web page that permits them to search a specific folder. ODWEK verifies that the user information is valid on the Content Manager OnDemand server, such as permission to access the server and data stored in an application group. After the user submits a search, ODWEK displays a Web page that contains a list of the documents that match the query. The user selects a document to view and ODWEK sends the document to the browser.

Figure 1 shows a workstation with a Web browser that is being used to access data from an OnDemand server.

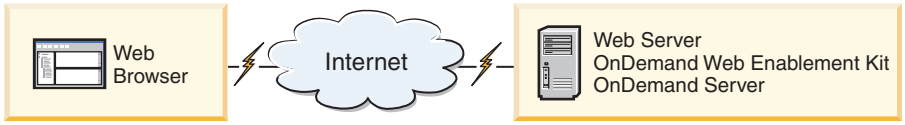

Figure 1. Accessing data stored in OnDemand using ODWEK.

ODWEK can search for and retrieve documents from Content Manager OnDemand servers that are running IBM Content Manager OnDemand for i Common Server Version 5, IBM Content Manager OnDemand for Multiplatforms Version 8 Release 5, and IBM Content Manager OnDemand for z/OS®, Version 8.

ODWEK contains several components:

- Content Manager OnDemand programming interface. The programming interface uses standard OnDemand interfaces and protocols to access data stored in an OnDemand server. No additional code is needed on the OnDemand server to support ODWEK. Use one of the following Content Manager OnDemand programming interfaces in your ODWEK application:
  - **Recommended:** Java Application Programming Interface (API). The Java API provides a way to access OnDemand data from a user-written program.
  - CGI program. The CGI program provides a way to access OnDemand data from a Web browser.
  - Java servlet. The servlet provides a way to access to OnDemand data from a Web browser. The servlet runs on a system that is running a Java-enabled HTTP server with a Java application server, such as the IBM WebSphere Application Server.

  Important: The CGI program and Java servlet continue to be provided and supported in IBM Content Manager OnDemand for Multiplatforms, Version 8.5. However, the code is stabilized and the strategic clients to use going forward are WEBi or a custom client that is designed by using the ODWEK Java API.
- IBM Content Manager OnDemand AFP Web Viewer. The AFP Web Viewer lets users search, retrieve, view, navigate, and print AFP documents from a Web browser.

- IBM Content Manager OnDemand Image Web Viewer. The Image Web Viewer lets users search, retrieve, view, navigate, and print BMP, GIF, JPEG, PCX, and TIFF documents from a Web browser.
- Java Line Data Viewer. The Java Line Data Viewer lets users view line data documents from a Web browser. IBM now provides two versions of the Java Line Data Viewer. See Chapter 8, "Installing and configuring viewing software," on page 73 and "Java line data viewer" on page 81 for information about the Java Line Data Viewer.
- Java AFP2HTML Viewer. The Java AFP2HTML Viewer lets users view the output generated by the IBM Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms service offering. The Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms converts AFP documents and resources into HTML files that can be displayed with the Java AFP2HTML Viewer. After installing and configuring the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms, an administrator enables the use of the Java AFP2HTML Viewer by configuring the ARSWWW.INI file.

Important: To view other types of documents stored in Content Manager OnDemand, you must obtain and install the appropriate viewer. For example, to view Adobe Portable Data Format (PDF) documents, IBM recommends that you obtain the Adobe Acrobat viewer for the browsers that are used in your organization.

About the programming interfaces

An *instance* of ODWEK (sometimes called an *application*) is ODWEK code that accesses data on a Content Manager OnDemand server. An instance controls what can be done to the data, and manages the system resources that are assigned to it. Each instance is a complete environment. An instance has its own ARSWWW.INI file and ODWEK programming interface, which other instances cannot access. There are three ODWEK programming interfaces:

- **Recommended:** Java API, a set of methods that may be used to access OnDemand data from a user-written program
- CGI program, an interface between a Web browser and an OnDemand server
- Java servlet, an interface between a Web browser and an OnDemand server

It is very important to understand that any single ODWEK instance can use only one programming interface. However, it is possible to run multiple instances of ODWEK on a single machine and let each instance use a different programming interface by running each separate ODWEK Instance within a separate Web Application or JVM.

The most common implementation of ODWEK is a single instance on a system. The single instance configuration is typically for developer or standalone production computing, which involves a single application server instance operating independently of any other applications.

Figure 2 on page 3 shows an example of a single instance using the Java API interface.

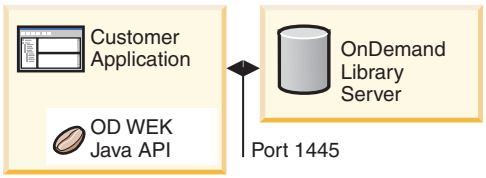

Figure 2. Single instance using Java API interface

Figure 3 shows an example of a single instance using the CGI interface.

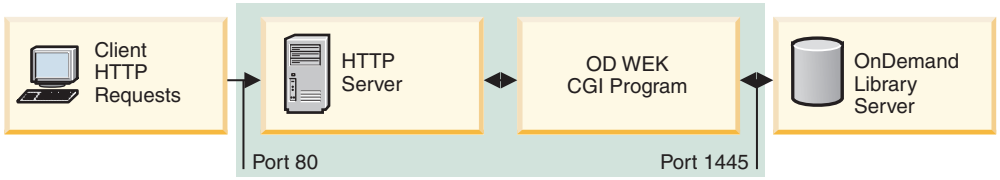

Figure 3. Single instance using CGI interface

Figure 4 shows an example of a single instance using the Java servlet interface.

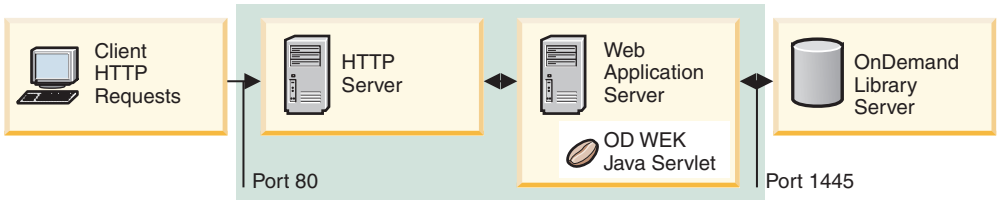

Figure 4. Single instance using Java interface

You can configure multiple instances of ODWEK on the same system. Each instance requires its own programming interface and ARSWWW.INI file, which specifies the unique port number over which communications between the programming interface and the Content Manager OnDemand server take place. Each instance also requires its own storage and security. The multiple instance configuration is typically for customers that need to run one or more developer, testing or production applications on the same system. The instances operate independently of each other.

Figure 5 on page 4 shows an example of the multiple instance topology.

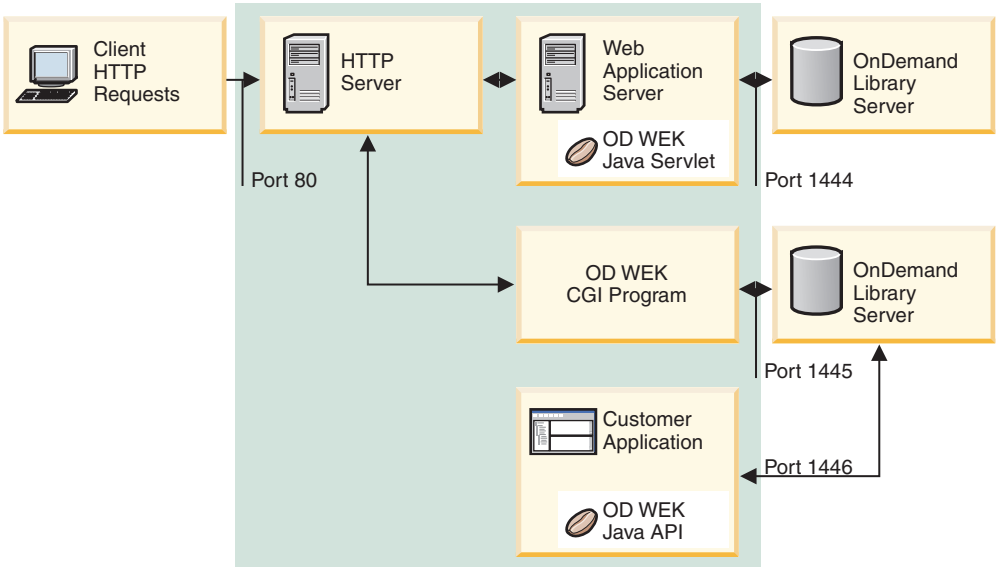

Figure 5. Multiple instance topology

Use cases of ODWEK Java APIs from a business perspective

From a business perspective, there are generally two different approaches to accessing the data that is stored in Content Manager OnDemand: Internet access and Intranet access.

In the Internet case, users (users that are external to the organization) are given access to a specific subset of information, For example, in a banking application, an internet user can log on and view his or her current statement or statements that cover the last twelve months. This is the only access that the user has to the Content Manager OnDemand archive, and the user is presented with a very limited set of menus to obtain this data. Typical internet usage involves tens of thousands of users accessing the Content Manager OnDemand archive concurrently. The Content Manager OnDemand architecture lets the system scale to the limits of the available hardware and network resources, and enables businesses to grow their system as their usage increases.

In the Intranet case, users (users that work for an organization) are given access to a wide variety of data based on their user ID access privileges in the Content Manager OnDemand archive. The user is presented with a selection criteria that allows him or her to access any of the stored data based on the security profile. The flexibility that is provided by the ODWEK Java APIs allows for the design of customized interfaces that are capable of meeting the specific requirements of an organization.

A common customer use case

A common use case across many different business segments involves allowing registered customers to select a range of documents to view. In the case of a banking industry, these documents can be bank statements that can be selected from a predefined number of months. For health insurance companies, the documents can be explanation of benefit statements. For utility companies, the documents can be bills or invoices.

The customer is usually a user who has been registered to the company's web site. The registration process gives entitlement back to the user in the form of credentials, for example, a user ID and password. To gain access to any of the company applications, the user submits the credentials from a web page such as a portal to be authenticated by the company's web application. After the user is authenticated, the application places constraints on what the user can perform when the user navigates the Web site.

Currently Content Manager OnDemand requires that a user ID be defined for any user that needs access. For the Internet users, it is not manageable to create a user ID for each registered Internet user. The user population might reach thousands or millions, therefore most applications perform a search on behalf of a user ID that has the permissions to search a specific folder and retrieve a document. The assumption is that if the user was not authenticated, and the constraints are placed on the search values to guarantee unique results, then one or more user IDs can be defined to have full permissions to the folder that is searched.

Regardless of the industry that has Content Manager OnDemand statements to present to customers, there are typically two kinds of transactions that a customer is allowed to perform:

- One transaction involves displaying a search results list of documents, usually within some predefined date range. The application performs the search on behalf of the user. As soon as the user is authenticated, the application chooses a unique key or key combination to ensure that the search results are for that user, and performs the search without any user interaction.
- Another transaction involves the user selecting a document from the search results list that was presented to the user. The data type of the document is taken into consideration when the content is displayed to the users. For example, if the document is stored in Content Manager OnDemand as an AFP data stream, either the user needs an AFP viewer program that is locally installed on the user's machine, or some data conversion needs to take place to provide a data stream that is more palatable for viewing. In many cases, transforms are performed by the application, such as AFP to PDF, to make it easier for the user to view. The PDF viewer is easily obtainable if it is not already installed on the user's machine. The AFP viewer, such as a plug-in might be more difficult to obtain as well as managed by the company provide the Web service.

The steps involved in this use case are:

1. User signs on to web site with their credentials. The web application authenticates the user, and retrieves a Content Manager OnDemand connection from a pool of connections.
2. The Web application selects a predefined folder to search.
3. The Web application assigns one or more Content Manager OnDemand folder field values to perform a search that is unique to the user.
4. The Web application performs a search, for example, by account number and date range, and returns a search result list to the user.
5. The Web application closes the folder and releases the connection back to the pool of connections.
6. The user selects a document to view from the search results list. Optional: the user is authenticated again before the user is allowed to retrieve the document.
7. The Web application retrieves a Content Manager OnDemand connection from a pool of connections.

8. The Web application retrieves the document and optionally performs a data transformation before it releases the data to the user.
9. The Web application releases the Content Manager OnDemand connection back to the pool of connections.

About the viewers

ODWEK provides the following viewers:

- AFP Web Viewer
- Image Web Viewer
- Java Line Data Viewer
- Java AFP2HTML Viewer

The AFP Web Viewer and the Image Web Viewer are software programs that extend the capabilities of a Web browser in a specific way. The AFP Web Viewer lets users view AFP documents. The Image Viewer lets users view BMP, GIF, JPEG, PCX, and TIFF documents. The viewers provide the capability to display documents in the browser window. Each viewer adds a toolbar to the top of the display window. The viewer toolbar may be in addition to the browser's toolbar. The viewer toolbar provides controls that can help users work with documents. Each user who plans to use the Web viewers to view documents must install them on their PC.

The Java Line Data Viewer is an applet that lets users view line data documents that are stored in Content Manager OnDemand. The Java Line Data Viewer displays line data documents in the browser window and adds a toolbar to the top of the display window. The Java Line Data Viewer toolbar provides controls that can help users work with documents. An administrator enables the use of the Java Line Data Viewer by configuring the ARSWWW.INI file.

The Java AFP2HTML Viewer is an applet that lets users view the output generated by the IBM Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms service offering. The Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms converts AFP documents and resources into HTML documents. After installing and configuring the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms, an administrator enables the use of the Java AFP2HTML Viewer by configuring the ARSWWW.INI file. The Java AFP2HTML Viewer provides a toolbar with controls that can help users work with documents, including controls for large objects.

One advantage of the applets is that users never have to install or upgrade software on the PC to use them, unlike the AFP Web Viewer and the Image Web Viewer, which must be installed on the PC. Also, if IBM provides a new version of the AFP Web Viewer or the Image Web Viewer, you must distribute the updated software to all users.

When using the applets and viewers that are provided by IBM, the documents that are retrieved from an OnDemand server remain compressed until reaching the viewer. The viewer uncompresses the documents and displays the pages in a Web browser window. If a document was stored in OnDemand as a large object, then the viewer retrieves and uncompresses segments of the document, as needed, when the user moves through pages of the document.

Using ODWEK

The most common method of using ODWEK is by customizing the sample HTML applications that are provided by IBM. The LOGON.HTM sample application supports users that are permitted access to several folders. To use the sample application, first modify the LOGON.HTM page with information about your OnDemand server. Then publish the URL of the LOGON.HTM file. Your users can then link to the URL and log on to the specified server. ODWEK automatically displays a series of Web pages for users to search for, retrieve, and display Content Manager OnDemand documents. The CREDIT.HTM sample application supports casual use of OnDemand by providing a Web page that contains search criteria for a specific folder. After you customize the sample, the user links to the URL, completes the search criteria, and presses the Submit button. ODWEK displays a Web page that lists the documents that match the query.

Important: ODWEK requires the ability to write cookie data on the PC. Users must configure their browsers to accept cookies.

Most customers define one OnDemand userid to access a server with ODWEK. This is common in environments with many casual users of OnDemand who will be accessing the same folder. You can also provide each user with their own OnDemand userid. Regardless of how you decide to access Content Manager OnDemand with ODWEK, you must manage the userids in OnDemand: you must add them to the server and set application group and folder permissions for the users.

Product functions

The following OnDemand functions are supported by ODWEK. You typically invoke the functions by creating Web pages that contain links to the ODWEK server program. Each link invokes a specific function. The output of one function is another Web page with links that lead the user to the next logical function. For example, the initial Web page may invoke the Logon function. The Logon function generates a Web page with a link to the Search Criteria function. Each function can be called with an API. See Appendix A, “Deploying the CGI program,” on page 89 for details.

Add Annotation

The Add Annotation function enables users to add an annotation to the specified document. To add an annotation, the user must be given the Add permission under Annotation in the Content Manager OnDemand application group. (A user is automatically given the Add permission when they are given permission to access an application group.)

Change Password

The Change Password function allows users to change their OnDemand passwords.

Document Hit List

The Document Hit List function builds the list of items that match the search criteria. The list is presented in an HTML table. Each item that matches the search is stored in a table cell and contains a link to the Retrieve Document function.

Logoff

The Logoff function allows users to log off an OnDemand server.

Logon

The Logon function allows the users to logon to an OnDemand server. If the Logon function is successful, the user is presented with a Web page that contains the list of folders that the user is authorized to open.

Retrieve Document

The Retrieve Document function retrieves a document from OnDemand. The data stream returned from the server includes the document, and depending on the data type, the resources required to view the document. The data stream must not be modified in any way. The browser, along with the viewer, interpret and decode the data stream and display the document. If the document is stored in OnDemand as a large object, then only the first segment of the document is returned. Subsequent segments of the document are retrieved and displayed as needed.

Search Criteria

After a successful logon, the user is presented with the list of folders that the user is authorized to open. The user selects a folder to open. Upon opening a folder, a Web page is displayed that contains the search fields for the folder. The user can accept the default search criteria or enter search criteria to search for specific documents. When the user presses the Submit button, the search request is sent to the OnDemand server.

Server Print Document

The Server Print Document function sends copies of documents to a Content Manager OnDemand server printer. To use server print, the user must be given the Print permission under Document in the Content Manager OnDemand application group. (A user is automatically given the Print permission when they are given permission to access an application group.) At least one server printer must be defined on the Content Manager OnDemand server.

Update Document

The Update Document function allows users to update the database. The Update Document function updates one or more database values for a specific document. To update a document, the user must be given the Update permission under Document in the Content Manager OnDemand application group.

View Annotations

The View Annotations function enables users to view the annotations attached to the specified document. To view annotations, the user must be given the View permission under Annotation in the Content Manager OnDemand each application group. (A user is automatically given the View permission when they are given permission to access an application group.)

Delete Annotations

The Delete Annotations function enables users to delete the annotations attached to a specified document.

Server and data security

There are two levels of security that you need to consider before you use ODWEK:

- Who can access the ODWEK programs and the Web pages
- Who can access data on the Content Manager OnDemand server

Any user that can access your Web server and the programs and Web pages that comprise the front-end to ODWEK can potentially access data stored in Content Manager OnDemand. IBM strongly encourages you to limit access to the programs and Web pages. There are many ways that you can limit access to programs and Web pages on your Web server. For example, many Web servers provide a system of security to sensitive Web pages by allowing you to restrict access to directories. You can also use a password file on the Web server, that requires users to enter a userid and password before accessing the Web pages. However, even though Web server userids and passwords are similar to operating system userids and passwords, there is no correspondence between them and operating system userids and passwords. There is also no correspondence between Web server userids and passwords and Content Manager OnDemand userids and passwords.

ODWEK provides access to Content Manager OnDemand servers and data using standard OnDemand APIs. The APIs verify that the Content Manager OnDemand user ID can access the server and the requested data. Someone in your organization must administer user and data security on the Content Manager OnDemand server.

There's one other security-related detail that you might want to consider: the method used to transfer form parameters and values between the browser and the server. The forms provided with ODWEK use the POST method to transfer parameters and values within the body of the HTTP request. With the POST method, the parameters and values do not appear in the Location field of the browser. For example, a typical function call appears as follows:

```
http://www.company.com/cgi-bin/arswww.cgi
```

However, if you do not specify a method when you create a form, then the default method is GET, which transfers parameters and values within the URL itself. With the GET method, a typical function call appears as follows:

```
http://www.company.com/cgi-bin/arswww.cgi?_function=logon
&_user=bob&_password=secret
```

The parameters and values appear as clear text in the Location field of the browser window. If you create your own forms, IBM strongly encourages you to use the POST method. To change the default method from GET to POST, you must code the METHOD attribute on the form tag.

IBM Web Interface for Content Management (WEBi)

IBM Web Interface for Content Management (WEBi) is an example of IBM's use of the ODWEK Java APIs. WEBi is a fully functional Web 2.0 technology based client that meets multiple customer needs. You can use WEBi either as your only browser or as a second browser implementation in addition to one or more of your own custom developed browser implementation(s). Like all other web implementations accessing the Content Manager OnDemand server, WEBi is built on top of the ODWEK Java APIs.

Chapter 2. Implementation overview

Implementation steps

Important: An *instance* of ODWEK (sometimes called an *application*) is ODWEK code that accesses data on an OnDemand server. An instance controls what can be done to the data, and manages the system resources that are assigned to it. Each instance is a complete environment. An instance has its own ARSWWW.INI file and ODWEK programming interface, which other instances cannot access. There are three ODWEK programming interfaces:

- Java API
- CGI program
- Java servlet

It is very important to understand that an instance may use only one programming interface. The programming interfaces are mutually exclusive. They cannot be used in the same instance at the same time. However, it is possible to run multiple instances of ODWEK on a single machine and have each instance use a different programming interface by configuring each instance to use a different port number. Additionally, each ODWEK servlet must run within a separate JVM, and must reference independent configuration files and directories.

To perform a basic installation of the ODWEK software:

1. Review the hardware and software requirements for ODWEK. See Chapter 3, “Installation requirements,” on page 13.
2. Install the ODWEK software on the system. See Chapter 4, “Installing ODWEK,” on page 15.

   Tip: After installing the ODWEK software, if you plan to use the Java API, see Chapter 5, “Deploying and programming Java API,” on page 21 for information about setting up the system environment and running ODWEK applications.
3. If you plan to use the CGI version of ODWEK, deploy the CGI program. See Appendix A, “Deploying the CGI program,” on page 89.
4. If you plan to use the Java servlet, deploy the servlet. See Appendix B, “Deploying the Java servlet,” on page 149.
5. Verify and configure the ODWEK configuration file (`ARSWWW.INI`). See “Specifying the ARSWWW.INI file” on page 89.
6. Modify the `logon.htm` sample application. See Chapter 7, “Configuring the sample applications,” on page 69.
7. Install the viewers. See Chapter 8, “Installing and configuring viewing software,” on page 73.
8. Verify the basic installation. See Chapter 9, “Verifying the installation,” on page 85.

Your next step

After you have reviewed the implementation steps, verified that the HTTP server and the Web application server are installed, configured and operational, and verified that any optional transforms that you may require are installed and configured, you can now install the ODWEK software on the system. For installation procedures, see the following sections:

- Chapter 3, “Installation requirements,” on page 13.
- Chapter 4, “Installing ODWEK,” on page 15.

Chapter 3. Installation requirements

Before you install the ODWEK software, you should make sure that your system meets ODWEK's hardware and software requirements.

For server, client, disk space, and transform requirements, see http://www.ibm.com/support/docview.wss?rs=129&uid=swg27016455 or search for 7016455 at: www.ibm.com.

Your next step

After you have determined that your system meets all hardware and software requirements, and after you have reviewed the implementation overview, you can now install the ODWEK software. For installation procedures, see Chapter 4, “Installing ODWEK,” on page 15.

Chapter 4. Installing ODWEK

This section explains how to install the ODWEK software on the system.

Before you begin

Before you begin the installation, make sure that your system meets all of the hardware and software requirements to install the ODWEK product. For more information, see Chapter 3, “Installation requirements,” on page 13.

You must install ODWEK for it to work properly. If you copy the Content Manager OnDemand Web Enablement Kit (ODWEK) files from the CD-ROM, and configure ODWEK, it does not work. When you try to execute the default log in Web page, you will receive the following error message: *”ICONV conversion failed. rc=7”*.

Your next step

To install ODWEK, go to the appropriate chapter:

- “Installing on AIX.”
- “Installing on HP-UX” on page 16.
- “Installing on Linux and Linux on zSeries” on page 17.
- “Installing on Solaris” on page 18.
- “Installing on Windows servers” on page 19.

Installing on AIX

Important: Native installations are no longer supported: SMIT, swinstall, pkgadd, and RPM.

To install the ODWEK software:

1. Insert the OnDemand for AIX server CD-ROM into the drive. The steps that follow assume that the CD-ROM is mounted on directory `/cdrom`.
2. Log in as the root user, and go to the `/cdrom/wek/aix` directory.
3. Enter the following command: `./odwekaix`.
4. In the IBM OnDemand Web Enablement Kit Setup window, read the Welcome screen and then click **Next**.
5. In the License Agreement window, Select **"I accept the terms in the license agreement to accept the license agreement."** Click **Next**.
6. Accept the default directory name. Click **Next**.
7. When the process completes, you see this question

   ```
   "Would you like to display the product ReadMe file?"
   ```

   The location of the product readme file is also displayed. On AIX, the readme file is located in the `/usr/lpp/ars/www or the /opt/ondemand/www` directory.
8. If you want to view the readme file now, click **Yes**. Otherwise, click **No**. Click **Next**.
9. Read the information in the window, and click **Next**.
10. Click **Finish**.

11. After you install the software from the CD-ROM, apply the latest service update for OnDemand. You can obtain the latest service update from IBM service at `ftp://service.software.ibm.com/software/ondemand/fixes/`.
12. After the installation completes successfully, eject the CD-ROM from the drive.

Optionally, you can install the software in the character-based console mode. To install the OnDemand for AIX server in the console mode, enter the following command from the directory which contains the installer:

```
./odwekaix -console
```

Follow the instructions on the installation panels.

Your next step

If you plan to use the Java API, go to Chapter 5, "Deploying and programming Java API," on page 21 for information about setting up the system environment and running ODWEK applications.

If you plan to use the CGI program, go to Appendix A, "Deploying the CGI program," on page 89.

If you plan to use the Java servlet, go to Appendix B, "Deploying the Java servlet," on page 149.

Installing on HP-UX

Important: Native installations are no longer supported: SMIT, swinstall, pkgadd, and RPM.

To install the OnDemand ODWEK product files on an HP-UX workstation:

1. Insert the OnDemand for HP-UX server CD-ROM into the drive. The steps that follow assume that the CD-ROM is mounted on directory `/cdrom`.
2. Log in as the root user, and go to the `/cdrom/server/hpux` directory.
3. Enter the following command: `./odwekhp`.
4. In the IBM OnDemand Web Enablement Kit Setup window, Read the Welcome screen and then click **Next**.
5. In the License Agreement window, select **"I accept the terms in the license agreement to accept the license agreement."** Click **Next**.
6. Accept the default directory name. Click **Next**.
7. When the process completes, you see the question

   ```
   "Would you like to display the product ReadMe file?"
   ```

   The location of the product readme file is also displayed. On HP-UX, the readme file is located in the `/usr/lpp/ars/www` or the `/opt/ondemand/www` directory.
8. If you want to view the readme file now, click **Yes**. Otherwise, click **No**. Click **Next**.
9. Read the information in the window, and click **Next**.
10. Click **Finish**.
11. After installing the software from the CD-ROM, apply the latest service update for OnDemand. You can obtain the latest service update from IBM service at `ftp://service.software.ibm.com/software/ondemand/fixes/`.

12. After the installation completes successfully, eject the CD-ROM from the drive.

Optionally, you can perform the installation in the character-based console mode. To install the OnDemand for HP-UX server in the console mode, enter the following command from the directory that contains the installer:

```
./odwekhp -console
```

Follow the instructions on the installation panels.

Your next step

If you plan to use the Java API, go to Chapter 5, "Deploying and programming Java API," on page 21 for information about setting up the system environment and running ODWEK applications.

If you plan to use the CGI program, go to Appendix A, "Deploying the CGI program," on page 89.

If you plan to use the Java servlet, go to Appendix B, "Deploying the Java servlet," on page 149.

Installing on Linux and Linux on zSeries

Important: Native installations are no longer supported: SMIT, swinstall, pkgadd, and RPM.

To install the OnDemand ODWEK product files on a Linux or Linux on zSeries workstation:

1. Insert the OnDemand for Linux server CD-ROM into the drive. The steps that follow assume that the CD-ROM is mounted on directory /cdrom.
2. Log in as the root user.
3. For installation on a Linux workstation:
   a. Go to the `/cdrom/wek/linux` directory.
   b. Enter this command: `./odweklinux`
4. For installation on a Linux on zSeries workstation:
   a. Go to the `/cdrom/wek/linux390` directory.
   b. Enter this command: `./odweklinux390`
5. Read the Welcome screen and then click **Next**.
6. In the License Agreement window, select **"I accept the terms in the license agreement to accept the license agreement."** Click **Next**.
7. Accept the default directory name. Click **Next**.
8. When the process completes, you see a question

   ```
   "Would you like to display the product ReadMe file?"
   ```

   The location of the product readme file is also displayed. On Linux and Linux on zSeries, the readme file is located in the `/usr/lpp/ars/www` or the `/opt/ondemand/www` directory.
9. If you want to view the readme file now, click **Yes**. Otherwise, click **No**. Click **Next**.
10. Read the information in the window, and click **Next**.
11. Click **Finish**.

12. After you install the software from the CD-ROM, apply the latest service update for OnDemand. You can obtain the latest service update from IBM service at `ftp://service.software.ibm.com/software/ondemand/fixes/`.
13. After the installation completes successfully, eject the CD-ROM from the drive.

Optionally, you can perform the installation in the character-based console mode. To install the OnDemand for Linux (or Linux on zSeries) server in the console mode, enter the following command from the directory which contains the installer: For Linux:

```
./odweklinux -console
```

For Linux on zSeries:

```
./odweklinux390 -console
```

Follow the instructions on the installation panels.

Your next step

If you plan to use the Java API, go to Chapter 5, "Deploying and programming Java API," on page 21 for information about setting up the system environment and running ODWEK applications.

If you plan to use the CGI program, go to Appendix A, "Deploying the CGI program," on page 89.

If you plan to use the Java servlet, go to Appendix B, "Deploying the Java servlet," on page 149.

Installing on Solaris

Important: Native installations are no longer supported: SMIT, swinstall, pkgadd, and RPM.

To install the OnDemand ODWEK product files on a Solaris workstation:

1. Insert the OnDemand for Solaris server CD-ROM into the drive. The steps that follow assume that the CD-ROM is mounted on directory `/cdrom`.
2. Log in as the root user.
3. Go to the `/cdrom/wek/sun` directory.
4. Enter this command: `./odweksun`
5. In the IBM OnDemand Web Enablement Kit Setup window, read the Welcome screen and then click **Next**.
6. In the License Agreement window, select **"I accept the terms in the license agreement to accept the license agreement."**, and click **Next**.
7. Accept the default directory name. Click **Next**.
8. When the process completes, you see the question

   ```
   "Would you like to display the product ReadMe file?"
   ```

   The location of the product readme file is also displayed. On Solaris, the readme file is located in the `/usr/lpp/ars/www` or the `/opt/ondemand/www` directory.
9. If you want to view the readme file now, click **Yes**. Otherwise, click **No**. Click **Next**.
10. Read the information in the window, and click **Next**.

11. Click **Finish**.
12. After installing the software from the CD-ROM, apply the latest service update for OnDemand. You can obtain the latest service update from IBM service at `ftp://service.software.ibm.com/software/ondemand/fixes/`.
13. After the installation completes successfully, eject the CD-ROM from the drive.

Optionally, the install can be performed in the character-based console mode. To install the OnDemand for Solaris server in the console mode, enter the following command from the directory which contains the installer:

```
./odweksun -console
```

Follow the instructions on the installation panels.

Your next step

If you plan to use the Java API, go to Chapter 5, "Deploying and programming Java API," on page 21 for information about setting up the system environment and running ODWEK applications.

If you plan to use the CGI program, go to Appendix A, "Deploying the CGI program," on page 89.

If you plan to use the Java servlet, go to Appendix B, "Deploying the Java servlet," on page 149.

Installing on Windows servers

Important: Native installs are no longer supported: SMIT, swinstall, pkgadd, and RPM.

To install the OnDemand ODWEK product files on a Windows workstation:

1. Log on with the OnDemand system administrator account (see "OnDemand system administrator account" on page 297 of the *DB2 CM OnDemand for Multiplatforms Installation and Configuration Guide* for details).
2. Insert the OnDemand for Windows Servers CD-ROM into the CD-ROM drive.
3. Choose Run from the Start menu. In the open box, type `x:\server\windows\odwin`, where x is the letter of the CDROM drive.
4. Read the Welcome screen and then click Next. The License Agreement window appears.
5. Select Yes to accept the license agreement. Click Next. The Destination window appears.
6. Accept the default directory name. Click Next. The Start Copying Files window appears.
7. Click Next. The progress window appears.
8. When the process completes, the Setup Complete windows appears.
9. If you want to view the README file now, click Finish. Otherwise, clear the View README checkbox and then click Finish to complete the installation and restart the computer.
10. After installing the software from the CD-ROM, apply the latest service update for OnDemand. You can obtain the latest service update from IBM service at ftp://service.software.ibm.com/software/ondemand/fixes/.

Your next step

If you plan to use the Java servlet, go to Appendix B, “Deploying the Java servlet,” on page 149.

If you plan to use the Java API, go to Chapter 5, “Deploying and programming Java API,” on page 21 for information about setting up the system environment and running ODWEK applications.

If you plan to use the CGI program, go to Appendix A, “Deploying the CGI program,” on page 89.

Chapter 5. Deploying and programming Java API

The Java application programming interfaces (APIs) are a set of classes that access and manipulate data on an OnDemand server. This section describes the Java APIs, the Java implementation of document functions, and Internet connectivity.

The Java APIs support:

- A common object model for data access
- Search and update across OnDemand servers.
- Client/server implementation for Java application users

Client/server architecture

The APIs provide a convenient programming interface for application users. APIs can reside on both the OnDemand server and the client (both provide the same interface), and the applications can be located locally or remotely. The client API communicates with the server to access data through the network. Communication between the client and the server is performed by classes; it is not necessary to add any additional programs.

The API classes consist of one package: `com.ibm.edms.od` .

Packaging for the Java environment

The API classes are contained in one package: `com.ibm.edms.od` . The classes are:

com.ibm.edms.od.ODApplication
This class represents a Content Manager OnDemand application. An instance of the ODApplication object provides an application developer access to information that is specified for a Content Manager OnDemand application.

com.ibm.edms.od.ODApplicationGroup
This class represents a Content Manager OnDemand application group. An instance of the ODApplicationGroup object provides an application developer access to information that is specified for a Content Manager OnDemand application group. An instance of the ODApplictionGroup object is generated through an instance of the ODServer object using the `getApplicationGroup()` method.

com.ibm.edms.od.ODApplicationGroupField
This class represents a Content Manager OnDemand application group field. It contains application group field information.

Important: You can access all ODWEK objects only in a single threaded environment.

com.ibm.edms.od.ODCabinet
This class represents a collection of Content Manager OnDemand folders. The administrator defines and manages cabinets on the Content Manager OnDemand server.

com.ibm.edms.od.ODCallback
This class is used with all methods in which the server operation returns data while processing.

com.ibm.edms.od.ODConfig
The ODConfig Java object is the preferred method to configure the system parameters. For more information, see "Configuring system parameters" on page 29.

com.ibm.edms.od.ODConstant
ODConstant is the public interface.

com.ibm.edms.od.ODCriteria
A class that represents the search criteria from an OnDemand folder. The criteria class contains methods to set a search operator and search values.

com.ibm.edms.od.ODException
This class represents the exceptions which might occur when using the APIs.

com.ibm.edms.od.ODFolder
A class that represents an OnDemand folder. This object is returned from a successful call to `ODServer.openFolder()`. This class contains folder criteria information. These criteria objects are what need to be modified in order to narrow the query on the server.

com.ibm.edms.od.ODHit
This class represents an OnDemand document.

com.ibm.edms.od.ODHitProperties
Use this class to obtain the Content Manager OnDemand internal property values for a hit.

com.ibm.edms.od.ODHold
This class represents an Content Manager OnDemand hold definition. This object is returned from a successful call to `(ODServer.getHolds())`.

Important: You must access all ODWEK objects in a single threaded environment.

com.ibm.edms.od.ODLogicalView
This class represents a Content Manager OnDemand logical view.

com.ibm.edms.od.ODNamedQuery
This class represents a Content Manager OnDemand named query. This class contains the details of a named query, and enables the system to retrieve existing named queries and save new named queries to the Content Manager OnDemand server.

com.ibm.edms.od.ODNamedQueryCriteria
This class represents the criteria of a Content Manager OnDemand named query. This class contains search criteria details that are stored in a named query.

com.ibm.edms.od.ODNote
This class represents an OnDemand annotation.

com.ibm.edms.od.ODServer
This class represents a connection to an OnDemand server. From this class you can logon, logoff and change the password. After a successful logon, this object will contain a list of all folders that the session has access to.

Access to this server object should be done in a single threaded environment. The only exception is when cancelling a server operation.

com.ibm.edms.od.ODUser
This class represents a Content Manager OnDemand user. From this class, you can gather user information such as address and phone number that is stored in the Content Manager OnDemand server.

Programming tips

You must import the `com.ibm.edms.od` package into your ODWEK application.

You do not need an HTTP server or a Web application server to run ODWEK applications that use the Java API. You can run the Java interpreter on ODWEK applications. Specify all Java Strings that are passed into the ODWEK APIs in thier proper Unicode encoding.

To run the Java interpreter on an ODWEK application:

1. Copy the shared library to your runtime directory or your Java.Library.Path:

Table 1. Shared Library Filename

Operating System	Shared Library
AIX	libars3wapi.a
HP-UX	libars3wapi.sl
Linux	libars3wapi.so
Solaris	libars3wapi.so
Windows	arswwwsl.dll

2. For Windows systems, copy the shared library to your runtime directory or your Java.Library.Path:

```
ARSSCKNT.DLL
ARSCT32.DLL
```

Setting up the system environment

When you set up your AIX, HP-UX, Linux, Solaris, or Windows environment, you must establish the following settings:

package
Import for all ODWEK applications.

- com.ibm.edms.od

Library files

Shared objects for AIX, HP-UX, Linux, and Solaris

DLLs for Windows

Setting environment variables

When developing an ODWEK application, you must set up your environment. Follow these instructions to specify environment variables for your platform or specify the library path via the Java Library Path parameter in the JVM instead of LIBPATH, SHLIB_PATH, LD_LIBRARY_PATH, or PATH environment variables.

AIX

In the AIX environment, you must set the following environment variables to set up your development environment for developing ODWEK applications.

LIBPATH Make sure your LIBPATH contains /usr/lpp/ars/www

CLASSPATH Make sure your CLASSPATH contains /usr/lpp/ars/www/api/ODApi.jar , which is the class library.

HP-UX

In the HP-UX environment, you must set the following environment variables to set up your development environment for developing ODWEK applications.

SHLIB_PATH Make sure your SHLIB_PATH contains /opt/ondemand/www

CLASSPATH Make sure your CLASSPATH contains /opt/ondemand/www/api/ODApi.jar , which is the class library.

Linux

In the Linux environment, you must set the following environment variables to set up your development environment for developing ODWEK applications.

LD_LIBRARY_PATH
Make sure your LD_LIBRARY_PATH contains /opt/ondemand/www

CLASSPATH Make sure your CLASSPATH contains /opt/ondemand/www/api/ODApi.jar , which is the class library.

Solaris

In the Solaris environment, you must set the following environment variables to set up your development environment for developing ODWEK applications.

LD_LIBRARY_PATH
Make sure your LD_LIBRARY_PATH contains /opt/ondemand/www

CLASSPATH Make sure your CLASSPATH contains /opt/ondemand/www/api/ODApi.jar , which is the class library.

Windows

In the Windows environment, you must set the following environment variables to set up your development environment for developing ODWEK applications.

PATH Make sure your PATH contains x :\yyyyyyyy \DLL; where x is the drive on which you installed ODWEK and yyyyyyyy is the installation directory for the ODWEK software.

CLASSPATH Make sure your CLASSPATH contains x :\yyyyyyyy \WWW \API\ODApi.jar, where x is the drive on which you installed ODWEK and yyyyyyyy is the installation directory for the class library.

Tracing and diagnostic information

To handle problems that arise in your Java API applications, you can use tracing and exception handling.

Tracing

ODWEK tracing writes trace statements to an arswww.trace file.

ODWEK tracing is intended to assist in problem determination. As with any form of tracing, there is a decline in the performance when you enable ODWEK tracing.

You must manually clean the trace file periodically. The trace file is not circular, and requires sufficient file space.

To enable the ODWEK tracing:

1. Modify the ODConfig default constructor. Follow instructions in "Configuring system parameters" on page 29.
2. Modify the debug stanza to reflect the following:

   ```
   [debug]
   ;Trace:None=0, Error=1, Error+Warn=2, Err+Warn+Info=3,All=4
   Trace=4
   TraceDir=/ars/tmp
   ```

   Tracing can be set to many different levels by the Trace parameter. When you troubleshoot an ODWEK issue, set the trace level to the highest level, unless otherwise specified by IBM support.

   If you set the trace at lower levels, for example `Trace=1`, that creates minimal overhead and alerts you of only error conditions. The lower trace levels are ideal for monitoring an ODWEK application that is in a steady state, and higher levels are typically used for troubleshooting a current ODWEK issue.

   If you use an `arswww.ini` from a previous release of ODWEK, delete the old debug section. Having multiple debug sections might inhibit ODWEK tracing.
3. The `arswww.trace` trace file is created in the directory referenced by the `TraceDir` parameter.
4. You must restart your ODWEK application for the `arswww.ini` file changes to take affect.
5. After tracing is enabled, recreate the issue and send the `arswww.trace` file to IBM support.

See Appendix E, "Problem determination tools," on page 163 for information about other tools that you can use to gather information about the system and documents.

Exception handling

When the Java APIs encounter a problem, they throw a Java exception. Throwing an exception creates an exception object of `ODException` class or one of its subclasses.

When an `ODException` is created, the API logs diagnostic information into a log file, assuming that logging is enabled. See "Tracing" on page 24 for more information about the log file used by the Java APIs.

When an `ODException` is caught, it allows you to see any error messages, error codes, and error states that occurred while running. When an error is caught, an error message is issued along with the location of where the exception was thrown. The error ID and exception ID are also given. The code below shows an example of the throw and catch process:

```
try
 {
   odServer = new ODServer(new ODConfig);
   odServer.initialize("TcUpdate.java");
   System.out.println( "Logging on to " + argv[0] + "..." );
   odServer.logon( argv[0], argv[1], argv[2] );
   odServer.logoff( );
   odServer.terminate( );
 }
```

```
catch ( ODException e )
 {
   System.out.println( "ODException: " + e );
   System.out.println( "   id = " + e.getErrorId( ) );
   System.out.println( "  msg = " + e.getErrorMsg( ) );
   e.printStackTrace( );
 }
```

Constants

The constants provided for use with the Java APIs are described in the JavaDoc that is shipped with ODWEK.

Running an ODWEK application

You can use the Java interpreter to run an ODWEK application. Please keep the following points in mind when creating, compiling and running an ODWEK application:

1. Create your ODWEK application by using the methods that are available to you in the Java API. Import the Java API package in your ODWEK application file. For example:

   ```
   //*********************************************************
   import java.util.*;
   import java.io.*;
   import com.ibm.edms.od.*;

   public class Logon
   {
     public static void main ( String argv[] )
     {
       .
       .
       .
     }
   }
   ```

2. Compile your ODWEK application file (`.java`) with `javac` to produce the `.class` file. See your Java reference publication for instructions on compiling Java applications.
3. Run the Java interpreter on your application (`.class` file). For example:

   ```
   java Logon server userid passwd /tmp/ondemand/www
   ```

   Where `Logon` is the name of the `.class` file, `server`, `userid`, and `passwd` are parameters for the application. **Note:** This example assumes that you have specified the path to the ODWEK class and servlet libraries by using system environment variables (see "Setting up the system environment" on page 23).

Connecting to an OnDemand server

Important: You must access an ODWEK instance, which consists of an ODServer object and all ODWEK objects that are created from that ODServer object such as ODFolder and ODHit, only by a single thread. If you attempt to access a single ODWEK instance through multiple threads, that attempt is not supported by IBM, and might cause undesired delays and behavior.

An object of the class `ODServer` represents and manages a connection to an OnDemand server, provides transaction support, and runs server commands.

Chapter 6, “Java API reference,” on page 67 explains where to find the online reference of methods and their descriptions.

When connecting to an OnDemand server, you must be aware of the requirements for the server; for example, the password for OnDemand can be no more than eight characters in length.

Establishing a connection

To establish a connection to OnDemand, complete the following steps:

1. Instantiate an ODConfig object with correct property values.
2. Instantiate an ODServer object with the ODConfig object.
3. Initiate the ODServer object.
4. Perform a logon with a valid OnDemand ID.

The `ODServer` class provides methods for connecting to an OnDemand server and disconnecting from the server. The following example uses an OnDemand library server named `LIBSRVR1` , the userid `ADMIN` and password `PASSWD` . The example creates an ODServer object for the OnDemand server, connects to it, works with it (not specified in the example), and then disconnects from it.

```
odServer = new ODServer( new ODConfig() );
System.out.println( "Logging on to " + "LIBSRVR1" + "..." );
odServer.logon( "LIBSRVR1", "ADMIN", "PASSWD" );
  .
  .
  .
odServer.logoff( );
odServer.terminate( );
```

See “Working with an OnDemand server” on page 30 for the complete sample application from which this example was taken.

The ODServerConnection class controls the API calls. The following example demonstrates how the getConnection method establishes the connection and returns a new ODServer object.

```
ODServer odServer = null;
ODConfig odConfig = null;
// get the custom afp2pdf transform properties
String transformPropsName = myProps.getProperty("TransformProperties");
Properties transformProps = new Properties();
transformProps.load(new FileInputStream(transformPropsName));
// Build the relative URL for the LineDataViewer Applet directory
// This would be something as '/ITSOWEK/applets'
String appletDir = new StringBuffer()
.append("/")
.append(contextRoot)
.append("/applets").toString();
// set the configuration values from my custom properties
odConfig = new ODConfig(myProps.getProperty("AfpViewer"), // AfpViewer
myProps.getProperty("LineViewer"), // LineViewer
myProps.getProperty("MetaViewer"), // MetaViewer
Long.parseLong(myProps.getProperty("MaxHits")), // MaxHits
appletDir, // AppletDir
myProps.getProperty("Language"), // Language
myProps.getProperty("TempDir"), // TempDir
myProps.getProperty("TraceDir"), // TraceDir
Integer.parseInt(myProps.getProperty("TraceLevel")), //
trace
// level
transformProps); // properties
```

```
odServer = new ODServer(new ODConfig);
odServer.initialize(new StringBuffer()
.append("/")
.append(contextRoot)
.append("/ODPassthru").toString());
// logon to the ODServer
odServer.logon(server, userid, password);
// print out the configuration values
// odConfig.printConfig();
return odServer;
```

The transForm properties parameter on the ODConfig constructor is optional, and is required only if an AFP2PDF transformation is performed by ODWEK.

The parameters of the ODServer.initialize method take on different meanings depending on the requirements of the application. If you use the ODWEK line data viewer applet for viewing line data content, specify the name of a servlet that accepts callback calls from the applet. Otherwise you can use any desired string value for the parameter.

A counter is incremented for each successful initialize() call and is decremented for each terminate() call. The ars3wapi.dll is loaded during the first ODServer.initialize() call, and is unloaded when the last ODServer is terminated. During the loading process, several internal environments are also initialized such as the Messaging and Trace engines. This is the only time when the Language= and TraceDir/TraceLevel settings are read from the ODConfig object.

Important: The proper way to end a user's connection is always to call the ODServer.logoff() and ODServer.terminate() methods.

After the servlet receives the new ODServer object, the reference is saved in the session object:

```
session.setAttribute("odServer", odServer);
```

If no exceptions are thrown, the servlet forwards the HTTP request and response references to either another servlet or JSP. In the case of a successful OnDemand connection, the servlet forwards the HTTP request on to the ODListFolders servlet as shown below.

```
request.getRequestDispatcher("/ODListFolders").forward(requ
est, response);
```

Setting and getting passwords

You can access or set a user's password on an OnDemand server by using the methods in ODServer. The following example shows how to set and get a user's password on an OnDemand library server.

```
odServer = new ODServer( new ODConfig() );
odServer.initialize( "AppName" );
odServer.setServerName( "LIBSRVR1" );
odServer.setUserId( "ADMIN" );
odServer.setPassword( "PASSWD" );

System.out.println( "Logging on to " + "LIBSRVR1" + "..." );

odServer.logon( odServer.getServerName( ),
                odServer.getUserId( ),
                        odServer.getPassword( ),
                        ODConstant.CONNECT_TYPE_LOCAL,
                        0 );
```

See “Working with an OnDemand server” on page 30 for the complete sample application from which this example was taken.

Configuring system parameters

The ODConfig Java object is the preferred method to configure the system parameters.

There are eight system parameters needed for a working ODServer instance. Use the ODConfig default constructor to set these parameters to default values:

```
<pre>
   try{
        ODConfig cfg = new ODConfig();
        ODServer srvr = new ODServer(cfg);
        srvr.initialize(null, "MyCustomApp");
        cfg.printConfig();
    }
    catch(ODException e){
        System.out.println("Exception " + e);
    }
 </pre>
```

This sample code configures the following default parameters:

```
<pre>
 AfpViewOpt   PLUGIN
 LineViewOpt  APPLET
 MaxHits      200
 MetaViewOpt  NATVIE
 AppletDir    /applets
 Language     ENU
 TempDir      The temp path as defined by the Java System.getProperty("java.io.tmpdir") method.
 TraceDir     The temp path as defined by the Java System.getProperty("java.io.tmpdir") method.
 TraceLevel   0
</pre>
```

For information on these parameters, see “Specifying the ARSWWW.INI file” on page 89.

You can also explicitly set these parameters by using the following sample code. This sample code uses a different ODConfig constructor:

```
<pre>
   try{
        ODConfig cfg = new ODConfig(ODConstant.PLUGIN, //AfpViewer
                                    ODConstant.APPLET, //LineViewer
                                    null,              //MetaViewer
                                    500,               //MaxHits
                                    "c:\\applets",     //AppletDir
                                    "ENU",             //Language
                                    "c:\\temp",        //TempDir
                                    "c:\\temp\\trace", //TraceDir
                                    1);                //TraceLevel

       ODServer srvr = new ODServer(cfg);
       srvr.initialize(null, "MyCustomApp");
       cfg.printConfig();
    }
    catch(ODException e){
        System.out.println("Exception " + e);
    }
 </pre>
```

This constructor sets parameters with zero or null values to the defaults mentioned above.

Important: This object has no methods to set parameters except during construction. The object cannot be modified after it has been constructed.

Working with an OnDemand server

An object of the class `ODServer` represents and manages a connection to an OnDemand server, provides transaction support, and runs server commands.

The following example uses `ODServer` methods to prepare for logon, set the application name, (optionally) display the local directory, display the server name, userid and password, display and set the connection type, display and set the port, and disconnect from the server.

This example demonstrates these `ODServer` methods:
- initialize
- logon
- logoff
- terminate
- getConnectType
- getPassword
- getPort
- getServerName
- getUserId
- setConnectType
- setPassword
- setPort
- setServerName
- setUserId

This example uses these run-time parameters:
- Server name
- User Id
- Password

Example of working with an OnDemand server:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;
public class TcServerMisc
{
   public static void main ( String argv[] )
   {
       ODServer odServer;
       ODConfig odConfig;
       String str;
       int j;
       //    ----------
       //     If too few parameters, display syntax and get out
       //    ----------
       if ( argv.length < 3 )
       {
           System.out.println( "usage: java TcServerMisc <server> <userid> <password> " );
           return;
       }
       try
       {
           //    ----------
           //     Set the stage
           //    ----------
           System.out.println( "This testcase should:" );
           System.out.println( " Use ODServer methods setServerName, setUserId, and setPassword" );
           System.out.println( " to prepare for logon" );
           System.out.println( " Set the application name" );
           System.out.println( " Server name" );
           System.out.println( " User Id" );
           System.out.println( " Password" );
           System.out.println( " Connect Type" );
           System.out.println( " Set and display the port" );
           System.out.println( " Set the connect type" );
           System.out.println( " Logoff" );
```

```
            System.out.println( "" );
            System.out.println( "Ensure that all information is correct." );
            System.out.println( "" );
            System.out.println( "-----------------------------------------------" );
            System.out.println( "" );
            //  ----------
            //Setup ODConfig object using defaults and initialize ODServer.
            //  ----------
            odConfig = new ODConfig();
            odServer = new ODServer(new ODConfig);
            odServer.initialize( "TcServerMisc.java" );

            odServer.setUserId( argv[1] );
            odServer.setPassword( argv[2] );
            System.out.println( "Logging on to " + argv[0] + "..." );
            //  ----------
            //  Logon to specified server
            // ----------
            odServer.logon( );

            //    ----------
            //     Test miscellaneous methods
            //    ----------
            System.out.println( "Setting application name to TcServerMisc.java..." );
            System.out.println( "Server Name: " + odServer.getServerName( ) );
            System.out.println( "User Id: " + odServer.getUserId( ) );
            System.out.println( "Password: " + odServer.getPassword( ) );
            j = odServer.getPort( );
            System.out.println( "Setting port to " + j + "..." );
            odServer.setPort( j );
            System.out.println( "Port: " + j );

            //    ----------
            //     Cleanup
            //    ----------
            System.out.println( "Logging off..." );
            odServer.logoff( );
            odServer.terminate( );
            System.out.println( "" );
            System.out.println( "-----------------------------------------------------" );
            System.out.println( "" );
            System.out.println( "Testcase completed - analyze if required" );
            System.out.println( "" );
        }
        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( " id = " + e.getErrorId( ) );
            System.out.println( " msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }
        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

Connecting to a non-default port using the Java APIs

In some instances, you may have the need to access a non-default port using ODWEK Java APIs. For example, you may have two instances on the OD Server. One instance uses the default port, and another uses a different port. Unless you configure your system properly, when you run your Java program, you'll receive the following error: "A connection cannot be established to the instance2 server".

To implement this arrangement, use the ODServer.setPort() API call just before the logon within your Java source.

Listing application groups in a folder

An object of the class `ODFolder` represents an OnDemand folder.

The following example uses `ODFolder` methods to display the number of application groups that can be searched from the folder and display the name of each application group.

This example demonstrates these `ODFolder` methods:
- getNumApplGroups
- close

This example also uses ODServer methods to prepare for logon, open the specified folder, and log off. This example demonstrates these ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example uses these run-time parameters:

- Server name
- User Id
- Password
- Folder name

Example of listing the application groups in a folder:

```
import java.util.Enumeration;

import com.ibm.edms.od.*;

public class TcApplGrp
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODFolder odFolder;
        ODConfig odConfig;
        Object[] appl_grps;
        ODApplicationGroupField agf_obj;
        int j;
        long agid = 0;

        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 4 )
        {
            System.out.println( "usage: java TcApplGrp <server> <userid> <password> <folder> " );
            return;
        }
        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "This testcase should:" );
            System.out.println( "  Logon to the specified server" );
            System.out.println( "  Open the specified folder" );
            System.out.println( "  Display the folder name" );
            System.out.println( "  Display the number of application groups" );
            System.out.println( "  Display the name and miscellaneous information of each application group" );
            System.out.println( "  Display the name and miscellaneous information of each application " );
            System.out.println( "" );
            System.out.println( "-----------------------------------------------------" );
            System.out.println( "" );

            //----------
            //Setup ODConfig object using defaults and initialize ODServer.
            //----------
            odConfig = new ODConfig();
            odServer = new ODServer(odConfig );
            odServer.initialize( "TcApplGrp.java" );

            //----------
            // Logon to the specified server
            //----------

            System.out.println( "Logging on to " + argv[0] + "..." );
            odServer.logon( argv[0], argv[1], argv[2] );

            //----------
            // Open the specified folder
            //----------
            System.out.println( "Opening " + argv[3] + " folder..." );
            odFolder = odServer.openFolder( argv[3] );
            //----------
            // Display number and names of application groups
            //----------
            System.out.println( "There is(are) " + odFolder.getNumApplGroups( ) + " application group(s) in the folder:" );
            appl_grps = odFolder.getApplGroupNames( );
            for ( j = 0; j < appl_grps.length; j++ )
                System.out.println( "  " + appl_grps[j].toString( ) );
            for (int i = 0; i < appl_grps.length; i++)
            {
                String appl_grp =  appl_grps[i].toString();
                System.out.println("Get the Application Group Object for " + appl_grp);
                ODApplicationGroup ag_obj = odServer.getApplicationGroup(appl_grp);
        System.out.println("AGName = " + ag_obj.getName());
        System.out.println("AGDescription = " + ag_obj.getDescription());
        System.out.println("AGid = " + ag_obj.getId());
        agid = ag_obj.getId();

                //---------
                // Check user permissions to add Docs to this AG
                //---------
```

```
                System.out.println("Has Perms to add documents = " +
                        ((ag_obj.getDocPerms() & ODConstant.OD_PERM_ADD) != 0 ? true : false));

                System.out.println("Has Perms to access logical views = " +
                        ((ag_obj.getIdPerms() & ODConstant.OD_APPLGRP_ID_LOGICAL_VIEW) != 0 ? true : false));

                //---------
                // Open Application
                //---------
                String[] appls = ag_obj.getApplicationNames();
                System.out.println("There are " + appls.length + " applications are defined for " + ag_obj.getName());
                for (int a = 0; a < appls.length; a++)
                {
                    String appl_name = appls[a];
                    System.out.println("  Open ODApplication" + (a + 1) + "  for " + appl_name);
                    ODApplication appl = ag_obj.getApplication(appl_name);

                    //---------
                    // Get the View Extension for the first Application in this AG.
                    //---------
                    System.out.println("   Application Name        = " + appl.getName());
                    System.out.println("    Application Description = " + appl.getDescription());
                    System.out.println("    Application Doc Type    = " + appl.getDocumentType());

                    String ext = appl.getExtension();
                    if(ext.length() > 0)
                        System.out.println("    Application View Ext    = " + ext);
                    else
                        System.out.println("    The Application is not DataType = UserDefined. There isn't any Extension defined.");


                }
            }

            //----------
            // Cleanup
            //----------
            odFolder.close( );
            odServer.logoff( );
            odServer.terminate( );
            System.out.println( "" );
            System.out.println( "-----------------------------------------------------" );
            System.out.println( "" );
            System.out.println( "Testcase completed - analyze results if required" );
            System.out.println( "" );
        }
        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

Searching a folder

An object of the class `ODFolder` represents an OnDemand folder. An object of the class `ODCriteria` represents the search criteria for an OnDemand folder. An object of the class `ODHit` represents an OnDemand document.

The following example uses `ODFolder` methods to open the specified folder, display the folder name, description, display order and search criteria, search the folder, and close the folder. This example uses `ODCriteria` methods to set the current search operand and search values. This example uses `ODHit` methods to get the display values for the document, get the document type, get a persistent identifier for the document, get the document location, and get the MIME content type for the document.

This example demonstrates these `ODFolder` methods:

- getName
- getDescription
- getDisplayOrder
- getCriteria
- search
- getSearchMessage
- close

This example demonstrates these ODCriteria methods:
- getName
- setOperator
- setSearchValue
- setSearchValues

This example demonstrates these ODHit methods:
- getDisplayValue
- getDisplayValues
- getDocType
- getMimeType
- getDocLocation
- getDocId

If you extract and reload the same document, the system generates a new DocID for the document, and the old DocID does not reference the new document.

This example also uses ODServer methods to prepare for logon, open the specified folder, and log off. This example demonstrates these ODServer methods:
- initialize
- logon
- openFolder
- terminate

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Criteria name
- Operator (must be one of eq, ne, lt, le, gt, ge, in, ni, li, nl, be, nb)
- Search value 1
- (optional) Search value 2

Important: The number of hits might be restricted by the MAXHITS specified in the ODConfig or passed to the search.

Example of searching a folder:

```
import java.util.Enumeration;
import java.util.Vector;

import com.ibm.edms.od.*;

public class TcSearch
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODConfig odConfig;
    ODFolder odFolder;
    ODCriteria odCrit;
    ODHit odHit;
    Enumeration values_enum;
    Vector hits;
    String[] display_crit;
    String server, userid, password, folder, directory;
    String crit = "", operator = "", value1 = "", value2 = "";
    String header, line1, line2, hit_value, useable_value;
    boolean mismatch_detected, use_default_values;
    int j, k, opr;
    int maxHits;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length <  9 && argv.length != 5 )
    {
      System.out.println( "usage: java TcSearch <server> <userid> <password> <folder> <MaxHits (-1 to use default)> <criteria> <opr> <value1> <value2> " );
      System.out.println( "         or, to use deafult search criteria" );
      System.out.println( "       java TcSearch <server> <userid> <password> <folder> <MaxHits (-1 to use default)> " );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
```

```
        System.out.println( "This testcase should:" );
        System.out.println( "  Logon to the specified server" );
        System.out.println( "  Open the specified folder" );
        System.out.println( "  Display the folder name and description" );
        System.out.println( "  If specified:" );
        System.out.println( "    Get the specified criteria" );
        System.out.println( "    Set the operator" );
        System.out.println( "    Set the operand(s)" );
        System.out.println( "  Search the folder" );
        System.out.println( "  Display search message (if any)" );
        System.out.println( "  Display the number of hits" );
        System.out.println( "  Display the hitlist with each hit using 3 lines:" );
        System.out.println( "    1. The hit values returned by the ODHit.getDisplayValue method" );
        System.out.println( "    2. The hit values returned by the ODHit.getDisplayValues method" );
        System.out.println( "    3. The doc type, mime type, doc location, and doc id values" );
        System.out.println( "" );
        System.out.println( "Ensure that lines 1 and 2 of the hitlist are the same and that the" );
        System.out.println( "hitlist values are the same as those displayed using the Windows Client." );
        System.out.println( "" );
        System.out.println( "-----------------------------------------------------" );
        System.out.println( "" );

        //----------
        // Logon to specified server
        //----------
        use_default_values = argv.length == 6;
        server   = argv[0];
        userid   = argv[1];
        password = argv[2];
        folder   = argv[3];
        maxHits  = Integer.parseInt(argv[4]);
        if ( use_default_values )
          directory = argv[5];
        else
        {
          crit      = argv[5];
          operator  = argv[6];
          value1    = argv[7];
          value2    = argv[8];
          directory = argv[9];
        }

//      ----------
        //Setup ODConfig object using defaults and initialize ODServer.
        //  ----------
        odConfig = new ODConfig();
        odServer = new ODServer(new ODConfig);
        odServer.initialize( "TcSearch.java" );
        System.out.println( "Logging on to " + server + "..." );
        odServer.logon( server, userid, password );

        //----------
        // Open the specified folder
        //----------
        System.out.println( "Opening " + folder + " folder..." );
        odFolder = odServer.openFolder( folder );

        System.out.println( "Name='" + odFolder.getName( ) + "' Desc='" + odFolder.getDescription( ) + "'" );

        //----------
        // If we are not using the default search values:
        //----------
        if ( !use_default_values )
        {
          //----------
          // Find the requested criteria
          //----------
          System.out.println( "Getting " + crit + " criteria..." );
          odCrit = odFolder.getCriteria( crit );
          if ( odCrit == null )
            System.out.println( "  *** " + crit + " criteria does not exist - NullPointerException will be reported" );

          //----------
          // Convert the operator parameter to the internal operator value and set
          // the criteria operator
          //----------
          System.out.println( "Setting operator to " + operator + "..." );
          if ( operator.equals( "eq" ) )
            opr = ODConstant.OPEqual;
          else if ( operator.equals( "ne" ) )
            opr = ODConstant.OPNotEqual;
          else if ( operator.equals( "lt" ) )
            opr = ODConstant.OPLessThan;
          else if ( operator.equals( "le" ) )
            opr = ODConstant.OPLessThanEqual;
          else if ( operator.equals( "gt" ) )
            opr = ODConstant.OPGreaterThan;
          else if ( operator.equals( "ge" ) )
            opr = ODConstant.OPGreaterThanEqual;
          else if ( operator.equals( "in" ) )
            opr = ODConstant.OPIn;
          else if ( operator.equals( "ni" ) )
            opr = ODConstant.OPNotIn;
          else if ( operator.equals( "li" ) )
            opr = ODConstant.OPLike;
          else if ( operator.equals( "nl" ) )
            opr = ODConstant.OPNotLike;
          else if ( operator.equals( "be" ) )
            opr = ODConstant.OPBetween;
          else if ( operator.equals( "nb" ) )
            opr = ODConstant.OPNotBetween;
          else
            opr = -1;

          System.out.println( "Setting operand(s)..." );
          odCrit.setOperator( opr );

          //----------
          // Set the search values
          //----------
          if ( opr == ODConstant.OPBetween || opr == ODConstant.OPNotBetween )
          {
            odCrit.setSearchValues( value1, value2 );
            System.out.println( "  " + odCrit.getName( ) + " " + getOperatorName( opr ) + " " + value1 + " and " + value2 );
          }
          else
          {
            odCrit.setSearchValue( value1 );
            System.out.println( "  " + odCrit.getName( ) + " " + getOperatorName( opr ) + " " + value1 );
          }
        }

        //----------
        // Set Max Hits limit if specified
        //----------
        if (maxHits != -1)
        {
            System.out.println("Setting MaxHITS to " + maxHits + ".");
```

```
        odFolder.setMaxHits(maxHits);
    }
    else
       System.out.println("No MaxHits passed in. Will use the MaxHits in ODConfig, or the Folder defined max, which ever is less.");

    //----------
    // Search the folder
    //----------
    System.out.println( "  Searching " + folder + ( use_default_values ? " using default values" : "" ) + "..." );

    long startTime  = System.currentTimeMillis();
    hits = odFolder.search( );
    long estimatedTime = System.currentTimeMillis() - startTime;
    System.out.println( "Elapsed Search Time: " +  estimatedTime + " ms");
    System.out.println( "    Search message: " + odFolder.getSearchMessage( ) );
    System.out.println( "    Number of hits: " + hits.size( ) );

    //----------
    // Display the hits
    //----------
    mismatch_detected = false;
    if ( hits != null && hits.size( ) > 0 )
    {
      display_crit = odFolder.getDisplayOrder( );
      header = "      ";
      for( j = 0; j < display_crit.length; j++ )
        header = header + display_crit[j] + "--";
      System.out.println( "      ----------------------------------------------------" );
      System.out.println( header + " (from ODHit.getDisplayValue method)" );
      System.out.println( header + " (from ODHit.getDisplayValues method)" );
      System.out.println( "        DocType--MimeType--DocLocation--DocId" );
      System.out.println( "      ----------------------------------------------------" );
      for ( j = 0; j < hits.size( ); j++ )
      {
        odHit = (ODHit)hits.elementAt( j );
        line1 = "      ";
        for ( k = 0; k < display_crit.length; k++ )
        {
          hit_value = odHit.getDisplayValue( display_crit[k] );
          useable_value = ( hit_value.equals( "" ) ) ? " " : hit_value;
          line1 = line1 + useable_value + "--";
        }
        System.out.println( line1 );
        line2 = "      ";
        for ( values_enum = odHit.getDisplayValues( ); values_enum.hasMoreElements( ); )
        {
          hit_value = (String)values_enum.nextElement( );
          useable_value = ( hit_value.equals( "" ) ) ? " " : hit_value;
          line2 = line2 + useable_value + "--";
        }
        System.out.println( line2 );
        System.out.println( "        " + getDocTypeString( odHit.getDocType( ) ) +
                            "--" + odHit.getMimeType( ) +
                            "--" + getLocationString( odHit.getDocLocation( ) ) +
                            "--" + odHit.getDocId( ) );
        if ( !line1.equals( line2 ) )
          mismatch_detected = true;
      }
    }

    //----------
    // Cleanup
    //----------
    odFolder.close( );
    odServer.logoff( );
    odServer.terminate( );
    System.out.println( "" );
    System.out.println( "----------------------------------------------------" );
    System.out.println( "" );
    System.out.println( "Testcase completed - analyze if required" );
    System.out.println( "" );
    if ( mismatch_detected )
    {
      System.out.println( "*** At least one mismatch was found between" );
      System.out.println( "***   lines 1 and 2 of a hit" );
      System.out.println( "" );
    }
  }

  catch ( ODException e )
  {
    System.out.println( "ODException: " + e );
    System.out.println( "   id = " + e.getErrorId( ) );
    System.out.println( "  msg = " + e.getErrorMsg( ) );
    e.printStackTrace( );
  }

  catch ( Exception e2 )
  {
    System.out.println( "exception: " + e2 );
    e2.printStackTrace( );
  }
}

static String getOperatorName( int oper )
{
  String str;

  switch( oper )
  {
    case ODConstant.OPEqual:
      str = "Equals";
      break;
    case ODConstant.OPNotEqual:
      str = "Not Equal";
      break;
    case ODConstant.OPLessThan:
      str = "Less Than";
      break;
    case ODConstant.OPLessThanEqual:
      str = "Less Than or Equal";
      break;
    case ODConstant.OPGreaterThan:
      str = "Greater Than";
      break;
    case ODConstant.OPGreaterThanEqual:
      str = "Greather Than or Equal";
      break;
    case ODConstant.OPIn:
      str = "In";
      break;
    case ODConstant.OPNotIn:
      str = "Not In";
      break;
    case ODConstant.OPLike:
      str = "Like";
      break;
    case ODConstant.OPNotLike:
```

```
      str = "Not Like";
      break;
    case ODConstant.OPBetween:
      str = "Between";
      break;
    case ODConstant.OPNotBetween:
      str = "Not Between";
      break;
    default:
      str = "Operator unknown";
      break;
  }

  return str;
}

static String getDocTypeString( char type )
{
  String str;

  switch( type )
  {
    case ODConstant.FileTypeAFP:
      str = "AFP";
      break;
    case ODConstant.FileTypeBMP:
      str = "BMP";
      break;
    case ODConstant.FileTypeEMAIL:
      str = "EMAIL";
      break;
    case ODConstant.FileTypeGIF:
      str = "GIF";
      break;
    case ODConstant.FileTypeJFIF:
      str = "JFIF";
      break;
    case ODConstant.FileTypeLINE:
      str = "LINE";
      break;
    case ODConstant.FileTypeMETA:
      str = "META";
      break;
    case ODConstant.FileTypeNONE:
      str = "NONE";
      break;
    case ODConstant.FileTypePCX:
      str = "PCX";
      break;
    case ODConstant.FileTypePDF:
      str = "PDF";
      break;
    case ODConstant.FileTypePNG:
      str = "PNG";
      break;
    case ODConstant.FileTypeTIFF:
      str = "TIFF";
      break;
    case ODConstant.FileTypeUSRDEF:
      str = "USRDEF";
      break;
    default:
      str = "*** Invalid Doc Type ***";
      break;
  }

  return str;
}

static String getLocationString( int loc )
{
  String str;

  switch( loc )
  {
    case ODConstant.DocLocationCache:
      str = "Cache";
      break;
    case ODConstant.DocLocationArchive:
      str = "Archive";
      break;
    case ODConstant.DocLocationExternal:
      str = "External";
      break;
    case ODConstant.DocLocationUnknown:
      str = "Unknown";
      break;
    default:
      str = "*** Invalid Doc Location ***";
      break;
  }

  return str;
 }
}
```

Searching a folder using an SQL string

The following example uses `ODFolder` methods to open the specified folder, search the folder with the specified SQL string, and close the folder. This example uses `ODHit` methods to display the number of items that match the query and to display the document list.

This example demonstrates these `ODFolder` methods:

- setApplGroupForSearchWithSQL
- search
- getDisplayOrder
- close

This example demonstrates these ODHit methods:
- getDisplayValue

This example also uses ODServer methods to prepare for logon, open the specified folder, and log off. This example demonstrates these ODServer methods:
- initialize
- logon
- openFolder
- logoff
- terminate

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Application group name
- SQL string

Example of searching a folder using an SQL string:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;
public class TcSearchWithSQL
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODConfig odConfig;
        ODFolder odFolder;
        ODHit odHit;
        Vector hits;
        String[] display_crit;
        String server, userid, password, folder, directory;
        String sql, appl_group;
        String header, line, hit_value, useable_value;
        int j, k;
        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 6 )
        {
            System.out.println( "usage: java TcSearchWithSQL <server> <userid> <password> <folder> <appl group> <sql string> " );
            return;
        }
        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "This testcase should:" );
            System.out.println( " Logon to the specified server" );
            System.out.println( " Open the specified folder" );
            System.out.println( " Search the folder with the specified SQL string" );
            System.out.println( " Display the number of hits" );
            System.out.println( " Display the hitlist" );
            System.out.println( "" );
            System.out.println( "-------------------------------------------------" );
            System.out.println( "" );
            //----------
            // Logon to specified server
            //----------
            server = argv[0];
            userid = argv[1];
            password = argv[2];
            folder = argv[3];
            appl_group = argv[4];
            sql = argv[5];
            //----------
            //Setup ODConfig object using defaults and initialize ODServer.
            //----------
            odConfig = new ODConfig();
            odServer = new ODServer(new ODConfig);
            odServer.initialize( "TcSearchWithSQL.java" );
            System.out.println( "Logging on to " + server + "..." );
            odServer.logon( server, userid, password );
            //----------
            // Open the specified folder
            //----------
            System.out.println( "Opening " + folder + " folder..." );
            odFolder = odServer.openFolder( folder );
            //----------
            // Search the folder
            //----------
            if ( appl_group.length( ) > 0 )
            {
                System.out.println( "Setting Appl Group to search: " + appl_group );
                odFolder.setApplGroupForSearchWithSQL( appl_group );
            }
```

```
            //----------
            // Search the folder
            //----------
            System.out.println( " Searching " + folder + "..." );
            hits = odFolder.search( sql );
            System.out.println( " Number of hits: " + hits.size( ) );
            //----------
            // Display the hits
            //----------
            if ( hits != null && hits.size( ) > 0 )
            {
                display_crit = odFolder.getDisplayOrder( );
                header = " ";
                for( j = 0; j < display_crit.length; j++ )
                    header = header + display_crit[j] + "--";
                System.out.println( " ------------------------------------------------" );
                System.out.println( header );
                System.out.println( " ------------------------------------------------" );
                for ( j = 0; j < hits.size( ); j++ )
                {
                    odHit = (ODHit)hits.elementAt( j );
                    line = " ";
                    for ( k = 0; k < display_crit.length; k++ )
                    {
                        hit_value = odHit.getDisplayValue( display_crit[k] );
                        useable_value = ( hit_value.equals( "" ) ) ? " " : hit_value;
                        line = line + useable_value + "--";
                    }
                    System.out.println( line );
                }
            }
            //----------
            // Cleanup
            //----------
            odFolder.close( );
            odServer.logoff( );
            odServer.terminate( );
            System.out.println( "" );
            System.out.println( "------------------------------------------------" );
            System.out.println( "" );
            System.out.println( "Testcase completed - analyze if required" );
            System.out.println( "" );
        }
        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( " id = " + e.getErrorId( ) );
            System.out.println( " msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }
        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

Cancelling a search

The following example uses the `ODServer.cancel` method to cancel a search in progress.

This example uses `ODServer` , `ODFolder` , and `ODCriteria` methods to logon to a server, open a folder, and set the Date criteria to 1970-2001. A second thread is then initiated to perform a search. When second thread completes, the number of hits is displayed. A second thread is again initiated, to perform a search. The process is put to sleep for .5 seconds and then the search is cancelled. When second thread completes, the number of hits is displayed.

This example demonstrates these `ODServer` methods:
- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates these `ODFolder` methods:
- getCriteria
- search
- close

This example demonstrates these `ODCriteria` methods:
- setSearchValues

This example uses these run-time parameters:

- Server name
- User Id
- Password
- Folder name

Example of cancelling a search:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;
class TestThread extends Thread
{
  ODFolder odFolder;

  TestThread( ODFolder fld )
  {
    odFolder = fld;
  }

  public void run( )
  {
    Vector hits;

    try
    {
      System.out.println( "  Second thread Searching..." );
      hits = odFolder.search( );
      System.out.println( "  Search completed - Number of hits: " + hits.size( ) );
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}

public class TcCancelSearch
{
  public static void main ( String argv[] )
  {
    ODServer odServer;    ODConfig odConfig;
    ODFolder odFolder;
    ODCriteria odCrit;
    TestThread search_thread;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 4 )
    {
      System.out.println( "usage: java TcCancelSearch <server> <userid> <password> <folder> " );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Set the Date criteria to 1970-2001" );
      System.out.println( "  Initiate a second thread to perform the search" );
      System.out.println( "  When second thread completes, display the number of hits" );
      System.out.println( "  Initiate a second thread to perform the search" );
      System.out.println( "  Sleep for .5 seconds" );
      System.out.println( "  Cancel the search" );
      System.out.println( "  When second thread completes, display the number of hits" );
      System.out.println( "" );
      System.out.println( "Ensure that a folder is chosen that includes a criteria named Date." );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );
      //----------
      //Setup ODConfig object using defaults and initialize ODServer.
      //----------
      odConfig = new ODConfig();
      odServer = new ODServer(new ODConfig);
      odServer.initialize( "TcCancelSearch.java" );

      //----------
```

```
      // Logon to specified server
      //----------
      System.out.println( "Logging on to " + argv[0] + "..." );
      odServer.logon( argv[0], argv[1], argv[2] );

      //----------
      // Open the specified folder and display its name and description
      //----------
      System.out.println( "Opening " + argv[3] + "..." );
      odFolder = odServer.openFolder( argv[3] );
      odCrit = odFolder.getCriteria( "Date" );
      odCrit.setOperator( ODConstant.OPBetween );
      odCrit.setSearchValues( "01/01/70", "01/01/01" );

      //----------
      // Start a search on a different thread, sleep briefly, awake and cancel search
      //----------
      System.out.println( "Main thread initiating search (will not attempt to cancel)..." );
      search_thread = new TestThread( odFolder );
      search_thread.start( );
      search_thread.join( );

      System.out.println( "Main thread initiating search (will attempt to cancel)..." );
      search_thread = new TestThread( odFolder );
      search_thread.start( );
      System.out.println( "Main thread sleeping for .5 seconds..." );
      ( Thread.currentThread( ) ).sleep( 500 );
      System.out.println( "Main thread attempting to cancel search..." );
      odServer.cancel( );
      System.out.println( "Main thread returned from attempt to cancel" );
      search_thread.join( );

      //----------
      // Cleanup
      //----------
      odFolder.close( );
      odServer.logoff( );
      odServer.terminate( );
      System.out.println( "" );
      System.out.println( "------------------------------------------------------" );
      System.out.println( "" );
      System.out.println( "Testcase completed - Ensure that the second search," );
      System.out.println( "  which was cancelled, yielded fewer hits than the first" );
      System.out.println( "" );
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}
```

Listing search criteria

The following example demonstrates how to use `ODCriteria` methods to list the search criteria for a given folder. For each search field, this example lists the name of the search field, the default operator, the operators that are valid for the field, the field type, and any default search values. The default values are listed by the `ODCriteria.getSearchValues` method. Fixed search values are listed for any search fields that are defined as `FixedChoice` or `Segment` .

This example demonstrates these `ODCriteria` methods:
- setOperator
- getType
- setSearchValues
- getFixedValues

This example demonstrates these `ODServer` methods:
- initialize
- logon
- openFolder
- logoff

- terminate

This example demonstrates these ODFolder methods:
- getCriteria
- close

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name

Example of accessing search criteria:

```
//*********************************************************************
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcListCriteria
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODCriteria odCrit;
    Enumeration crit_enum;
    Vector value_vec;
    String[] search_values, fixed_values;
    int[] valid_oprs;
    int j, opr;
    char field_type;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcListCriteria <server> <userid> <password> <folder> <config dir> [<local server dir>]" );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Display the folder name and description" );
      System.out.println( "  Display the number of folder criteria" );
      System.out.println( "  For each criteria, display the" );
      System.out.println( "    Name" );
      System.out.println( "    Default operator" );
      System.out.println( "    Valid operators" );
      System.out.println( "    Field Type" );
      System.out.println( "    Default values (by ODCrit.getSearchValues method)" );
      System.out.println( "    Fixed values (only for FixedChoice and Segment criteria)" );
      System.out.println( "" );
      System.out.println( "Ensure that none of the operators indicates 'Unknown operator'," );
      System.out.println( "that none of the field types indicates 'Unknown type',  that the" );
      System.out.println( "default values are the same for each method, and that all" );
      System.out.println( "information is the same as that displayed using the Windows Client." );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to the specified server
      //----------
      odServer = new ODServer(new ODConfig);
      odServer.initialize( argv[4], "TcListCriteria.java" );

      System.out.println( "Logging on to " + argv[0] + "..." );
      if ( argv.length == 5 )
        odServer.logon( argv[0], argv[1], argv[2] );
      else
        if ( argv.length == 6 )
          odServer.logon( argv[0], argv[1], argv[2], ODConstant.CONNECT_TYPE_LOCAL, 0, argv[5] );

      //----------
      // Open the specified folder and display its name and description
      //----------
      System.out.println( "Opening " + argv[3] + " folder..." );
      odFolder = odServer.openFolder( argv[3] );
      System.out.println( "Name='" + odFolder.getName( ) + "' Desc='" + odFolder.getDescription( ) + "'" );
      System.out.println( "There are " + odFolder.getNumCriteria( ) + " criteria:" );

      //----------
      // For each folder criteria,
      //----------
      for ( crit_enum = odFolder.getCriteria( ); crit_enum.hasMoreElements( ); )
      {
```

```
      //----------
      // Display criteria name
      //----------
      System.out.println( "" );
      odCrit = (ODCriteria)crit_enum.nextElement( );
      System.out.println( odCrit.getName( ) );

      //----------
      // Display default operator
      //----------
      opr = odCrit.getOperator( );
      System.out.println( "  Default operator: " );
      System.out.println( "    " + getOperatorName( opr ) );

      //----------
      // Display valid operators
      //----------
      valid_oprs = odCrit.getValidOperators( );
      System.out.println( "  Valid operators:" );
      for ( j = 0; j < valid_oprs.length; j++ )
        System.out.println( "    " + getOperatorName( valid_oprs[j] ) );

      //----------
      // Display field type
      //----------
      field_type = odCrit.getType( );
      System.out.println( "  Type:" );
      System.out.println( "    " + getTypeName( field_type ) );

      //----------
      // Display default value(s) using ODCrit.getSearchValues( )
      //----------
      value_vec = odCrit.getSearchValues( );
      System.out.println("  Default Value(s) (ODCrit.getSearchValues method):");
      System.out.println( "    '" + value_vec.elementAt( 0 ) + "'" );
      System.out.println( "    '" + value_vec.elementAt( 1 ) + "'" );


      //----------
      // Display fixed choices
      //----------
      switch ( field_type )
      {
        case ODConstant.InputTypeChoice:
        case ODConstant.InputTypeSegment:
          fixed_values = odCrit.getFixedValues( );
          System.out.println("  Fixed Values (only for field types FixedChoice and Segment):");
          for ( j = 0; j < fixed_values.length; j++ )
            System.out.println("    '" + fixed_values[j] + "'" );
          break;
      }
    }

    //----------
    // Cleanup
    //----------
    odFolder.close( );
    odServer.logoff( );
    odServer.terminate( );
    System.out.println( "" );
    System.out.println( "-----------------------------------------------------" );
    System.out.println( "" );
    System.out.println( "Testcase completed - analyze and compare results to" );
    System.out.println( "                     Windows Client if required" );
    System.out.println( "" );
  }

  catch ( ODException e )
  {
    System.out.println( "ODException: " + e );
    System.out.println( "   id = " + e.getErrorId( ) );
    System.out.println( "  msg = " + e.getErrorMsg( ) );
    e.printStackTrace( );
  }

  catch ( Exception e2 )
  {
    System.out.println( "exception: " + e2 );
    e2.printStackTrace( );
  }
}

static String getOperatorName( int oper )
{
  String str;

  switch( oper )
  {
    case ODConstant.OPEqual:
      str = "Equal";
      break;
    case ODConstant.OPNotEqual:
      str = "Not Equal";
      break;
    case ODConstant.OPLessThan:
      str = "Less Than";
      break;
    case ODConstant.OPLessThanEqual:
      str = "Less Than or Equal";
      break;
    case ODConstant.OPGreaterThan:
      str = "Greater Than";
      break;
```

```
      case ODConstant.OPGreaterThanEqual:
        str = "Greather Than or Equal";
        break;
      case ODConstant.OPIn:
        str = "In";
        break;
      case ODConstant.OPNotIn:
        str = "Not In";
        break;
      case ODConstant.OPLike:
        str = "Like";
        break;
      case ODConstant.OPNotLike:
        str = "Not Like";
        break;
      case ODConstant.OPBetween:
        str = "Between";
        break;
      case ODConstant.OPNotBetween:
        str = "Not Between";
        break;
      default:
        str = "*** Unknown operator";
        break;
    }

    return str;
  }

  static String getTypeName( char type )
  {
    String str;

    switch( type )
    {
      case ODConstant.InputTypeNormal:
        str = "Normal";
        break;
      case ODConstant.InputTypeTextSearch:
        str = "TextSearch";
        break;
      case ODConstant.InputTypeNoteTextSearch:
        str = "NoteTextSearch";
        break;
      case ODConstant.InputTypeNoteColor:
        str = "NoteColor";
        break;
      case ODConstant.InputTypeChoice:
        str = "FixedChoice";
        break;
      case ODConstant.InputTypeSegment:
        str = "Segment";
        break;
      default:
        str = "*** Unknown type";
        break;
    }

    return str;
  }
}
```

Listing folders and folder information

The following example uses `ODServer` methods to print a line showing the number of folders on the specified server that may be searched by the specified userid. The example prints one line for each folder, showing the folder name and description.

This example demonstrates these `ODServer` methods:

- initialize
- logon
- getNumFolders
- getFolderNames
- getFolderDescription
- logoff
- terminate

This example uses these run-time parameters:

- Server name
- User Id
- Password

Example of listing folders and folder information:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;
public class TcListFolders
{
  public static void main ( String argv[] )
  {
    ODServer    odServer;
    ODConfig odConfig;
    Enumeration folders_enum;
    String folder_name;
    String folder_desc;
    int num_folders;
   //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 4 )
   {
      System.out.println( "usage: java TcListFolders <server> <userid> <password> <Folder Criteria>" );
      return;
    }
    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Display a line showing number of folders on the server available to the userid" );
      System.out.println( "  Display one line for each folder, showing name and description" );
      System.out.println( "" );
      System.out.println( "The information should be the same as that displayed using the Windows Client" );
   System.out.println( "(with the 'All' button checked if available), but the sequence of the folders" );
      System.out.println( "may be different depending on the server specified" );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );

      //----------
      //Setup ODConfig object using defaults and initialize ODServer.
      //----------
      odConfig = new ODConfig();
      odServer = new ODServer(new ODConfig);
      odServer.initialize( "TcListFolders.java" );
      //----------
      // Logon to specified server
      //----------
      System.out.println( "Logging on to " + argv[0] + "..." );
      odServer.logon( argv[0], argv[1], argv[2] );


      //----------
      // Display the number of folders available.
      //----------
      num_folders = odServer.getNumFolders( );
      System.out.println( "" );
      System.out.println( "There are " + num_folders + " folders available to " + argv[1] + " on " + argv[0] + ":" );

      //----------
      // Display the folder names and descriptions
      //----------
      for ( folders_enum = odServer.getFolderNames( ); folders_enum.hasMoreElements( ); )
      {
        folder_name = (String)folders_enum.nextElement( );
        folder_desc = odServer.getFolderDescription( folder_name );
        System.out.println( "  " + folder_name + "  ---  " + folder_desc );
      }

      //----------
      // get a limited folder list based on search criteria
      //----------
      System.out.println("\n*************************************\n");
      System.out.println("List folders found using search criteria");
      Enumeration folders_enum2 = odServer.getFolderNames(argv[3]);
      if (folders_enum2 != null)
         while ( folders_enum2.hasMoreElements( ) )
         {
      folder_name = (String)folders_enum2.nextElement( );
      folder_desc = odServer.getFolderDescription( folder_name );
      System.out.println( "  " + folder_name + "  ---  " + folder_desc );
         }
      else
         System.out.println("No Folders match specified Search Criteria.");
      //----------
      // Cleanup
      //----------
      odServer.logoff( );
      odServer.terminate( );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );
      System.out.println( "Testcase completed - compare results to Windows Client if required" );
      System.out.println( "" );
    }
    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }
    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}
```

Displaying folder criteria information

The following example uses `ODServer`, `ODFolder`, and `ODCrit` methods to open a folder on a specified server, displays the folder name and description, and prints the folder criteria information. Folder criteria includes:

- Name
- Default operator
- Valid operators
- Field Type
- Default values (by `ODCrit.getSearchValues` method)
- Fixed values (only for FixedChoice and Segment criteria)

To display folder criteria information, ensure the following:

- None of the operators indicates `'Unknown operator'`.
- None of the field types indicates `'Unknown type'`.
- For each method, the default values are the same.
- The folder criteria information that is displayed by ODWEK API matches what is displayed by the Content Manager OnDemand client.

This example demonstrates these `ODServer` methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates these `ODFolder` methods:

- getCriteria
- close

This example demonstrates these `ODCrit` methods:

- getType
- getSearchValues
- getFixedValues

This example uses these run-time parameters:

- Server name
- User ID
- Password
- Folder name
- Configuration directory (contains `arswww.ini`)
- Optional - Local server directory

Example of listing folders and folder information:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcListCritOrders
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODCriteria odCrit;
    Enumeration crit_enum;
    String[] value_vec;
    String[] search_values, fixed_values;
    int[] valid_oprs;
    int j, opr;
    char field_type;
    Object[] applGroup;
```

```
String [] dbFieldName;

//----------
// If too few parameters, display syntax and get out
//----------
if ( argv.length < 5 )
{
  System.out.println( "usage: java TcListCritOrders <server> <userid> <password> <folder> <config dir> [<local server dir>]" );
  return;
}

try
{
  //----------
  // Set the stage
  //----------
  System.out.println( "This testcase should:" );
  System.out.println( "  Logon to the specified server" );
  System.out.println( "  Open the specified folder" );
  System.out.println( "  Display the folder name and description" );
  System.out.println( "  Display the number of folder criteria" );
  System.out.println( "  For each criteria, display the" );
  System.out.println( "    Name" );
  System.out.println( "    Default operator" );
  System.out.println( "    Valid operators" );
  System.out.println( "    Field Type" );
  System.out.println( "    Default values (by ODCrit.getSearchValues method)" );
  System.out.println( "    Fixed values (only for FixedChoice and Segment criteria)" );
  System.out.println( "" );
  System.out.println( "Ensure that none of the operators indicates 'Unknown operator'," );
  System.out.println( "that none of the field types indicates 'Unknown type',  that the" );
  System.out.println( "default values are the same for each method, and that all" );
  System.out.println( "information is the same as that displayed using the Windows Client." );
  System.out.println( "" );
  System.out.println( "-----------------------------------------------------" );
  System.out.println( "" );

  //----------
  // Logon to the specified server
  //----------
  ODConfig odConfig = BuildODConfig.build(argv[4] + "/arswww.props");
  if (odConfig == null)
    System.out.println("BuildODConfig Failed.");

  else
  {
    odServer = new ODServer(new ODConfig);
    odServer.initialize(  "TcListCritOrders.java" );
    System.out.println( "Logging on to " + argv[0] + "..." );
    if ( argv.length == 5 )
      odServer.logon( argv[0], argv[1], argv[2] );
    else
      if ( argv.length == 6 )
      odServer.logon( argv[0], argv[1], argv[2], ODConstant.CONNECT_TYPE_LOCAL, 0, argv[5] );

    //----------
    // Open the specified folder and display its name and description
    //----------
    System.out.println( "Opening " + argv[3] + " folder..." );
    odFolder = odServer.openFolder( argv[3] );
    System.out.println( "Name='" + odFolder.getName( ) + "' Desc='" + odFolder.getDescription( ) + "'" );
    System.out.println( "There are " + odFolder.getNumCriteria( ) + " criteria:" );

    //----------
    // For each folder criteria,
    //----------
    System.out.println( "" );
    for ( crit_enum = odFolder.getCriteria( ); crit_enum.hasMoreElements( ); )
    {
      //----------
      // Display criteria name
      //----------
      odCrit = (ODCriteria)crit_enum.nextElement( );
      System.out.println( odCrit.getName( ) );
    }

    ArrayList sortedCrits = odFolder.getCriteriaSortOrder();
    System.out.println( "\nNumber of sorted critiera = " + sortedCrits.size() );
    for (int i=0; i < sortedCrits.size(); i++)
    {
      //----------
      // Sort criteria
      //----------
      System.out.println( "----------------------------" );
      odCrit = (ODCriteria)sortedCrits.get(i);
      System.out.println( i+1 + ". " +odCrit.getName( ) );
      displayCriteriaValues(odCrit);
    }

    ArrayList displayCrits = odFolder.getCriteriaDisplayOrder();
    System.out.println( "\nNumber of display critiera = " + displayCrits.size() );
    for (int i=0; i < displayCrits.size(); i++)
    {
      //----------
      // Display criteria
      //----------
      System.out.println( "----------------------------" );
      odCrit = (ODCriteria)displayCrits.get(i);
      System.out.println( i+1 + ". " +odCrit.getName( ) );
      displayCriteriaValues(odCrit);
    }

    ArrayList queryCrits = odFolder.getCriteriaQueryOrder();
    System.out.println( "\nNumber of query critiera = " + queryCrits.size() );
    for (int i=0; i < queryCrits.size(); i++)
    {
      //----------
      // Query criteria
      //----------
```

```
        System.out.println( "----------------------------" );
        odCrit = (ODCriteria)queryCrits.get(i);
        System.out.println( i+1 + ". " +odCrit.getName( ) );
        displayCriteriaValues(odCrit);
      }


      //----------
      // Cleanup
      //----------
      odFolder.close( );
      odServer.logoff( );
      odServer.terminate( );
      System.out.println( "" );
      System.out.println( "------------------------------------------------------" );
      System.out.println( "" );
      System.out.println( "Testcase completed - analyze and compare results to" );
      System.out.println( "                     Windows Client if required" );
      System.out.println( "" );
    }
  }

  catch ( ODException e )
  {
    System.out.println( "ODException: " + e );
    System.out.println( "   id = " + e.getErrorId( ) );
    System.out.println( "  msg = " + e.getErrorMsg( ) );
    e.printStackTrace( );
  }

  catch ( Exception e2 )
  {
    System.out.println( "exception: " + e2 );
    e2.printStackTrace( );
  }
}

static void displayCriteriaValues( ODCriteria odCrit )
{
  String[] value_vec;
  String[] search_values, fixed_values;
  int[] valid_oprs;
  int j, opr;
  char field_type;

  //----------
  // Display default operator
  //----------
  opr = odCrit.getOperator( );
  System.out.println( "  Default operator: " );
  System.out.println( "    " + getOperatorName( opr ) );

  //----------
  // Display valid operators
  //----------
  valid_oprs = odCrit.getValidOperators( );
  System.out.println( "  Valid operators:" );
  for ( j = 0; j < valid_oprs.length; j++ )
    System.out.println( "    " + getOperatorName( valid_oprs[j] ) );

  //----------
  // Display field type
  //----------
  field_type = odCrit.getType( );
  System.out.println( "  Type:" );
  System.out.println( "    " + getTypeName( field_type ) );

  //----------
  // Display default value(s) using ODCrit.getSearchValues( )
  //----------
  value_vec = odCrit.getSearchValues( );
  System.out.println("  Default Value(s) (ODCrit.getSearchValues method):");
  System.out.println( "    '" + value_vec[ 0 ] + "'" );
  System.out.println( "    '" + value_vec[1] + "'" );


  //----------
  // Display fixed choices
  //----------
  switch ( field_type )
  {
    case ODConstant.InputTypeChoice:
    case ODConstant.InputTypeSegment:
      fixed_values = odCrit.getFixedValues( );
      System.out.println("  Fixed Values (only for field types FixedChoice and Segment):");
      for ( j = 0; j < fixed_values.length; j++ )
        System.out.println("    '" + fixed_values[j] + "'" );
      break;
  }
}

static String getOperatorName( int oper )
{
  String str;

  switch ( oper )
  {
    case ODConstant.OPEqual:
      str = "Equal";
      break;
    case ODConstant.OPNotEqual:
      str = "Not Equal";
      break;
    case ODConstant.OPLessThan:
      str = "Less Than";
      break;
    case ODConstant.OPLessThanEqual:
      str = "Less Than or Equal";
      break;
```

```
      case ODConstant.OPGreaterThan:
        str = "Greater Than";
        break;
      case ODConstant.OPGreaterThanEqual:
        str = "Greather Than or Equal";
        break;
      case ODConstant.OPIn:
        str = "In";
        break;
      case ODConstant.OPNotIn:
        str = "Not In";
        break;
      case ODConstant.OPLike:
        str = "Like";
        break;
      case ODConstant.OPNotLike:
        str = "Not Like";
        break;
      case ODConstant.OPBetween:
        str = "Between";
        break;
      case ODConstant.OPNotBetween:
        str = "Not Between";
        break;
      default:
        str = "*** Unknown operator";
        break;
    }

    return str;
  }

  static String getTypeName( char type )
  {
    String str;

    switch ( type )
    {
      case ODConstant.InputTypeNormal:
        str = "Normal";
        break;
      case ODConstant.InputTypeTextSearch:
        str = "TextSearch";
        break;
      case ODConstant.InputTypeNoteTextSearch:
        str = "NoteTextSearch";
        break;
      case ODConstant.InputTypeNoteColor:
        str = "NoteColor";
        break;
      case ODConstant.InputTypeChoice:
        str = "FixedChoice";
        break;
      case ODConstant.InputTypeSegment:
        str = "Segment";
        break;
      default:
        str = "*** Unknown type";
        break;
    }

    return str;
  }
}
```

Displaying a list of documents

The following example uses `ODFolder` and `ODHit` methods to search a folder using the default search criteria, print the number of documents that matched the query, and lists the documents that matched the query.

This example demonstrates these `ODFolder` methods:

- getName
- getDisplayOrder
- search
- close

This example demonstrates these `ODHit` methods:

- getDisplayValue

This example also demonstrates these `ODServer` methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example uses these run-time parameters:

- Server name
- User Id
- Password
- Folder name

Example of displaying a list of documents:

```
//*********************************************************************
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcSortedHitlist
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    Vector hits;
    String[] display_crit;
    String server, userid, password, folder, value;
    int j, k;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcSortedHitlist <server> <userid> <password> <folder> <config dir>" );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Search the folder using the default criteria" );
      System.out.println( "  Display search message (if any)" );
      System.out.println( "  Display the number of hits" );
      System.out.println( "  Display the hitlist" );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to the server
      //----------
      server = argv[0];
      userid = argv[1];
      password = argv[2];
      folder = argv[3];
      odServer = new ODServer(new ODConfig);
      odServer.initialize( argv[4], "TcSortedHitlist.java" );
      System.out.println( "Logging on to " + server + " as " + userid + "/" + password + "..." );
      odServer.logon( server, userid, password );

      //----------
      // Open and search the folder
      //----------
      System.out.println( "Opening " + folder + "..." );
      odFolder = odServer.openFolder( folder );
      System.out.println( "Searching folder with default criteria..." );
      hits = odFolder.search( );
      System.out.println( "  Number of hits: " + hits.size( ) );

      //----------
      // Display the hits
      //----------
      if ( hits != null && hits.size( ) > 0 )
      {
        display_crit = odFolder.getDisplayOrder( );
        value = "    ";
        for( j = 0; j < display_crit.length; j++ )
          value = value + display_crit[j] + " ";
        System.out.println( value );
        for ( j = 0; j < hits.size( ); j++ )
        {
          odHit = (ODHit)hits.elementAt( j );
```

```
          value = "    ";
          for ( k = 0; k < display_crit.length; k++ )
            value = value + odHit.getDisplayValue( display_crit[k] ) + " ";
          System.out.println( value );
        }
      }

      //----------
      // Cleanup
      //----------
      odFolder.close( );
      odServer.logoff( );
      odServer.terminate( );
      System.out.println( "" );
      System.out.println( "--------------------------------------------------" );
      System.out.println( "" );
      System.out.println( "Testcase completed - Ensure that the order of the hits" );
      System.out.println( "  is the same as shown by the Windows Client" );
      System.out.println( "" );
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}
```

Retrieving a document

The following example completes these tasks:

- Logging on to the specified server
- Opening the specified folder
- Searching the folder by using the default criteria
- Displaying the number of hits
- Retrieving the data for the first hit by using `ODHit.getDocument (filename, true)`, which causes ODWEK to write all segments of the document to the specified file
- Retrieving the document again by using `ODHit.getDocument()`, which outputs the document data to byte array
- Verifying whether the document is AFP, if so, calling `ODHit.getResource()` to retrieve the AFP resource file

This example demonstrates these `ODServer` methods:

- initialize
- logon
- openFolder
- retrieve
- logoff
- terminate

This example demonstrates these `ODFolder` methods:

- search
- retrieve
- close

This example demonstrates these `ODHit` methods:

- getDocId
- retrieve

Important: If you extract and reload the same document, the system generates a new DocId for the document, and the old DocId does not reference the new document.

This example uses these run-time parameters:

- Server name
- User Id
- Password
- Folder name

Example of retrieving a document:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcRetrieve {
    public static void main(String argv[]) {
        ODServer odServer;
        ODFolder odFolder;
        ODConfig odConfig;
        ODHit odHit;
        Vector hits;

        //----------
        // If too few parameters, display syntax and get out
        //----------
        if (argv.length < 4)
        {
            System.out
            .println("usage: java TcRetrieve <server> <userid> <password> <folder> ");
            return;
        }
        try {
            //----------
            // Set the stage
            //----------
            System.out.println("This testcase should:");
            System.out.println("  Logon to the specified server");
            System.out.println("  Open the specified folder");
            System.out.println("  Search the folder using the default criteria");
            System.out.println("  Retrieve LO document data via ODHit.getDocument()");
            System.out.println("  Retrieve LO document data via ODHit.getDocument(Filename)");
            System.out.println("");
            System.out
            .println("-----------------------------------------------------");
            System.out.println("");

            //----------
            //Setup ODConfig object using defaults and initialize ODServer.
            //----------
            odConfig = new ODConfig();
            odServer = new ODServer(new ODConfig);
            odServer.initialize( "TcRetrieve.java" );
            //----------
            // Logon to specified server
            //----------
            System.out.println("Logging on to " + argv[0] + "...");
            odServer.logon(argv[0], argv[1], argv[2]);
            //----------
            // Open the specified folder and search with the default criteria
            //----------

            System.out.println("Opening " + argv[3] + " folder...");
            odFolder = odServer.openFolder(argv[3]);
            System.out.println("Searching with default criteria...");
            hits = odFolder.search();
            System.out.println("Number of hits: " + hits.size());
            if (hits.size() > 0)
            {
                odHit = (ODHit) hits.elementAt(0);

                System.out.println("\nGet the Document and write it directly to file");
                String ext = odHit.getFileExt();
                odHit.getDocument("TcRetrieve1." + ext ,true);
                System.out.println("Doc data written to " + "TcRetrieve1." + ext);

                System.out.println("\nGet the Document ByteArray, then write array to file.");
                System.out.println("NOTE This will only return the 1st segment by design.");
                byte[] docdata = odHit.getDocument();
                FileOutputStream fos1 = new FileOutputStream("TcRetrieve2." + ext);
                fos1.write(docdata);
                fos1.close();
                System.out.println("Data written to TcRetrieve." + ext);

                //---------
                //If document is AFP, go get the resources
                //---------
                if(odHit.getDocType() == ODConstant.FileTypeAFP)
                {
                    System.out.println("\nDocument is AFP, so we will now get the resource file.");
                    String rid = odHit.getResourceID();
                    System.out.println("Resource id is " + rid);
                    System.out.println("Call GetResource with filename");
                    odHit.getResources(rid +".res");
                    System.out.println("AFP Resources written to " + rid + ".res");
                }
            }
            // Call Java implementation of AFP2xxx, Xenos or other transforms with the byte array
```

```
            // to convert the data, if required. Also, if using a 3rd party AFP viewer, the Resources and the Document
            // data can be concatenated for viewing.


            //----------
            // Cleanup
            //----------
            odFolder.close();
            odServer.logoff();
            odServer.terminate();
            System.out.println("");
            System.out
            .println("-------------------------------------------------");
        }

        catch (ODException e) {
            System.out.println("ODException: " + e);
            System.out.println("   id = " + e.getErrorId());
            System.out.println("  msg = " + e.getErrorMsg());
            e.printStackTrace();
        }
        catch (Exception e2) {
            System.out.println("exception: " + e2);
            e2.printStackTrace();
        }
    }
}
```

The following example uses `ODCallback` methods for bulk retrieval of document data.

```
//*************************************************************************************************
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcCallback extends ODCallback
{
  byte[] data_from_folder;
  boolean init = true;

  TcCallback( )
  {
  }

  public void HitHandleCallback( int hit, int off, int len )
  {
  }

  public boolean HitCallback( String docid, char type, String[] values )
                  throws Exception
  {
    return true;
  }

  public boolean DataCallback( byte[] data )
  {
    byte[] temp;
    int j, k;

    //----------
    // If first data block received, initialize container; otherwise,
    // append new data to that previously received.
    //----------
    if ( init )
    {
      data_from_folder = data;
      init = false;
    }
    else
    {
      temp = new byte[ data_from_folder.length + data.length ];
      for ( j = 0; j < data_from_folder.length; j++ )
        temp[j] = data_from_folder[j];
      k = data_from_folder.length;
      for ( j = 0; j < data.length; j++ )
        temp[k++] = data[j];
      data_from_folder = temp;
    }

    return true;
  }

  public byte[] getData( )
```

```
  {
    return data_from_folder;
  }
}
```

Printing a document

The following example uses ODServer and ODFolder methods to list the printers that are available on the server and to print a document to the specified server printer. This example also uses ODServer methods to prepare for logon, open the specified folder, and log off.

This example demonstrates these ODServer methods:
- initialize
- logon
- openFolder
- getServerPrinters
- logoff
- terminate

This example demonstrates these ODFolder methods:
- search
- close

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Printer name

Example of printing a document:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcPrintHit
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    Vector hits;
    Vector hit_to_print;
 String [] printers;
 String printer_name;
 boolean match;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcPrintHit <server> <userid> <password> <folder> <printer> " );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Display the list of printers available on the server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Search the folder using the default criteria" );
      System.out.println( "  Display the number of hits" );
      System.out.println( "  Print the first hit to the specified server printer" );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );
      //----------
      //Setup ODConfig object using defaults and initialize ODServer.
```

```
    //----------
    ODConfig odConfig = new ODConfig();
    odServer = new ODServer(new ODConfig);
    odServer.initialize( "TcPrintHit.java" );

    //----------
    // Logon to specified server
    //----------
    System.out.println( "Logging on to " + argv[0] + "..." );
    odServer.logon( argv[0], argv[1], argv[2] );

    //----------
    // If any server printers are available on the server
    //----------
    System.out.println( "Retrieving list of server printers..." );
    printer_name = argv[4];
    printers = odServer.getServerPrinters( );
    if ( printers.length > 0 )
    {
      //----------
      // List the available server printers
      //----------
      System.out.println( "There are " + printers.length + " printers available on the server:" );
      match = false;
for( j = 0; j < printers.length; j++ )
{
       System.out.println( "  " + printers[j] );
  if ( printers[j].equals( printer_name ) )
    match = true;
}

      if ( match )
{
        //----------
        // Open the specified folder and search with the default criteria
        //----------
        System.out.println( "Opening " + argv[3] + " folder..." );
        odFolder = odServer.openFolder( argv[3] );
        System.out.println( "Searching with default criteria..." );
        hits = odFolder.search( );
        System.out.println( "  Number of hits: " + hits.size( ) );

        //----------
        // Print the first hit to the specified server printer
        //----------
        if ( hits.size( ) > 0 )
        {
    hit_to_print = new Vector( );
          odHit = (ODHit)hits.elementAt( 0 );
          hit_to_print.addElement( odHit );
          System.out.println( "Printing first hit to " + printer_name + "..." );
          odFolder.printDocuments( hit_to_print, printer_name,1 );
        }
  else
          System.out.println( "There is no document to print" );

        odFolder.close( );
}
else
        System.out.println( "The specified printer (" + printer_name + ") is not avilable on this server" );
 }
 else
      System.out.println( "No printers are avilable on this server" );

    //----------
    // Cleanup
    //----------
    odServer.logoff( );
    odServer.terminate( );
    System.out.println( "" );
    System.out.println( "----------------------------------------------------" );
    System.out.println( "" );
    System.out.println( "Testcase completed - Analyze the results" );
    System.out.println( "" );
  }

  catch ( ODException e )
  {
    System.out.println( "ODException: " + e );
    System.out.println( "   id = " + e.getErrorId( ) );
    System.out.println( "  msg = " + e.getErrorMsg( ) );
    e.printStackTrace( );
  }

  catch ( Exception e2 )
  {
    System.out.println( "exception: " + e2 );
    e2.printStackTrace( );
  }
 }
}
```

Listing information about notes

The following example uses ODNote methods to list detailed information about a note. This example logs on to the specified server, opens the specified folder, searches the folder using the default criteria, displays the number of hits, displays the number of notes associated with the first document, and displays detailed information for each note that is attached to the document. The information includes the position of the note on the page of the document, the background color, the date and time that the note was attached to the document, the userid that created the note and other attributes.

This example demonstrates these ODNote methods:
- getColor
- getDateTime
- getGroupName
- getOffsetX
- getOffsetY
- getPageNum
- getText
- getUserid
- isOkToCopy
- isPublic

This example also demonstrates these ODServer methods:
- initialize
- logon
- openFolder
- logoff
- terminate

This example also demonstrates these ODFolder methods:
- search
- close

This example also demonstrates these ODHit methods:
- getNotes

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name

Example of listing information about notes:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcListNotes
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    ODNote odNote;
    Vector hits, notes = null;
    int j, exist;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 4 )
    {
```

```
  System.out.println( "usage: java TcListNotes <server> <userid> <password> <folder> " );
  return;
}

try
{
  //----------
  // Set the stage
  //----------
  System.out.println( "This testcase should:" );
  System.out.println( "  Logon to the specified server" );
  System.out.println( "  Open the specified folder" );
  System.out.println( "  Search the folder using the default criteria" );
  System.out.println( "  Display the number of hits" );
  System.out.println( "  Display the note status for the first hit" );
  System.out.println( "  Display the number of notes associated with the first hit" );
  System.out.println( "  Display info for each note" );
  System.out.println( "" );
  System.out.println( "-------------------------------------------------" );
  System.out.println( "" );

  //----------
  //Setup ODConfig object using defaults and initialize ODServer.
  //----------
  ODConfig odConfig = new ODConfig();
  odServer = new ODServer(new ODConfig);
  odServer.initialize( "TcListNotes.java" );

  //----------
  // Logon to specified server
  //----------
   System.out.println( "Logging on to " + argv[0] + "..." );
   odServer.logon( argv[0], argv[1], argv[2] );

  //----------
  // Open the specified folder and search with the default criteria
  //----------
  System.out.println( "Opening " + argv[3] + " folder..." );
  odFolder = odServer.openFolder( argv[3] );
  System.out.println( "Searching with default criteria..." );
  hits = odFolder.search( );
  System.out.println( "  Number of hits: " + hits.size( ) );

  //----------
  // List info for each note for the first hit
  //----------
  if ( hits.size( ) > 0 )
  {
    odHit = (ODHit)hits.elementAt( 0 );
    System.out.println("  For the first hit:" );
    exist = odHit.getNoteStatus();
    System.out.println("    Note status allowing UNKNOWN is: " + getExistenceName( exist ) );
    notes = odHit.getNotes( );
    if(notes != null)
       System.out.println("    There is(are) " + notes.size( ) + " note(s)" );
    for ( j = 0; j < notes.size( ); j++ )
    {
      odNote = (ODNote)notes.elementAt( j );
      System.out.println("      " + (j+1) + ". Text='" + odNote.getText( ) + "'" );
      System.out.println("         UserId=" + odNote.getUserId( ) );
      System.out.println("         Page=" + odNote.getPageNum( ) );
      System.out.println("         Color=" + odNote.getColor( ) );
      System.out.println("         Date=" + odNote.getDateTime( ) );
      System.out.println("         Group=" + odNote.getGroupName( ) );
      System.out.println("         Offset=(" + odNote.getOffsetX( ) + "," + odNote.getOffsetY( ) + ")" );
      System.out.println("         OkToCopy=" + odNote.isOkToCopy( ) );
      System.out.println("         Public=" + odNote.isPublic( ) );
      System.out.println("         Text=" + odNote.getText( ) );

    }
  }
  else
    System.out.println( "There is no document - cannot list notes" );

  //----------
  // Cleanup
  //----------
  odFolder.close( );
  odServer.logoff( );
  odServer.terminate( );
  System.out.println( "" );
  System.out.println( "-------------------------------------------------" );
  System.out.println( "" );
  System.out.println( "Testcase completed - Ensure that the information" );
  System.out.println( "  is the same as shown by the Windows Client" );
  System.out.println( "" );
}
```

```
    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }

  static String getExistenceName( int code )
  {
    String str;

    switch( code )
    {
      case ODConstant.NoteStatusUnknown:
        str = "UNKNOWN";
        break;
      case ODConstant.NoteStatusYes:
        str = "YES";
        break;
      case ODConstant.NoteStatusNo:
        str = "NO";
        break;
      case ODConstant.NoteStatusError:
        str = "ERROR";
        break;
      default:
        str = "UNDEFINED VALUE";
    }

    return str;
  }
}
```

Adding a note

An object of the class `ODHit` represents an OnDemand document. The following example uses `ODHit` methods to display the number of notes associated with the document and add a new note with the these attributes:

- The specified note text
- OkToCopy=false
- Public=false (that is, a private note)
- An empty group name

This example demonstrates these `ODHit` methods:

- getNotes
- addNote

This example also uses `ODServer` methods to prepare for logon, open the specified folder, and log off and uses the `ODFolder` methods to search the folder, get the number of hits that matched the query, and close the folder. This example demonstrates these `ODServer` methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates these `ODFolder` methods:

- search
- getHits
- close

This example uses these run-time parameters:

- Server name
- User Id
- Password
- Folder name
- Text of note

Example of adding an annotation:

```
import java.util.*;import java.io.*;import com.ibm.edms.od.*;

public class TcAddNote
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    ODNote odNote;
    Vector hits;
    Vector notes;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcAddNote <server> <userid> <password> <folder> <note text> " );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Search the folder using the default criteria" );
      System.out.println( "  Display the number of hits" );
      System.out.println( "  Display the number of notes associated with the first hit" );
      System.out.println( "  Add a new note with the these attributes" );
      System.out.println( "    The specified note text" );
      System.out.println( "    OkToCopy=false" );
      System.out.println( "    Public=false (i.e. a private note)" );
      System.out.println( "    An empty group name" );
      System.out.println( "" );
      System.out.println( "--------------------------------------------------" );
      System.out.println( "" );

      //----------
      //Setup ODConfig object using defaults and initialize ODServer.
      //----------
      ODConfig odConfig = new ODConfig();
      odServer = new ODServer(new ODConfig);
      odServer.initialize( "TcAddNote.java" );
      //----------
      // Logon to specified server
      //----------
      System.out.println( "Logging on to " + argv[0] + "..." );
      odServer.logon( argv[0], argv[1], argv[2] );

      //----------
      // Open the specified folder and search with the default criteria
      //----------
      System.out.println( "Opening " + argv[3] + " folder..." );
      odFolder = odServer.openFolder( argv[3] );
      System.out.println( "Searching with default criteria..." );
      odFolder.search( );
      hits = odFolder.getHits( );
      System.out.println( "  Number of hits: " + hits.size( ) );

      //----------
      // Add a new note
      //----------
      if ( hits.size( ) > 0 )
      {
        odHit = (ODHit)hits.elementAt( 0 );
        notes = odHit.getNotes( );
        if(notes.size() > 0)
           System.out.println("  There are " + notes.size( ) + " notes for the first hit" );
```

```
        odNote = new ODNote( );
    odNote.setText( argv[4] );
        odNote.setGroupName( "" );
        odNote.setOkToCopy( false );
        odNote.setPublic( false );

        System.out.println("  Adding a new note with:" );
        System.out.println("    Text='" + odNote.getText( ) + "'" );
        System.out.println("    OkToCopy=" + odNote.isOkToCopy( ) );
        System.out.println("    Public=" + odNote.isPublic( ) );
        System.out.println("    Group=" + odNote.getGroupName( ) );

        odHit.addNote( odNote );
     }
     else
        System.out.println( "No document - cannot list notes" );

       //----------
       // Cleanup
       //----------
       odFolder.close( );
       odServer.logoff( );
       odServer.terminate( );
       System.out.println( "" );
       System.out.println( "--------------------------------------------------" );
       System.out.println( "" );
       System.out.println( "Testcase completed - Ensure that the new note was correctly" );
       System.out.println( "  added by displaying it with the Windows Client" );
       System.out.println( "" );
     }

     catch ( ODException e )
     {
       System.out.println( "ODException: " + e );
       System.out.println( "   id = " + e.getErrorId( ) );
       System.out.println( "  msg = " + e.getErrorMsg( ) );
       e.printStackTrace( );
     }

     catch ( Exception e2 )
     {
       System.out.println( "exception: " + e2 );
       e2.printStackTrace( );
     }
   }
}
```

Deleting a note

This example completes the following functions:

1. Logging on to the specified server
2. Opening the specified folder
3. Searching the folder by using the default criteria
4. Displaying the number of hits
5. Displaying the number of notes that are associated with the first hit
6. Deleting the first note

This example demonstrates these `ODServer` methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates these `ODFolder` methods:

- search
- getHits
- close

This example demonstrates these `ODHit` methods:

- getNotes
- deleteNote

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Configuration directory (contains arswww.ini)
- (Optional) Local server directory

Example of deleting an annotation:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcDeleteNote
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    ODNote odNote;
    Vector hits;
    Vector notes;
    int j, numNotes1 = 0, numNotes2 = 0;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcDeleteNote <server> <userid> <password> <folder> <config dir> " );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Search the folder using the default criteria" );
      System.out.println( "  Display the number of hits" );
      System.out.println( "  Display the number of notes associated with the first hit" );
      System.out.println( "  Delete the first note" );
      System.out.println( "" );
      System.out.println( "-------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer(new ODConfig);

      ODConfig odConfig = BuildODConfig.build(argv[4] + "/arswww.props");

      if (odConfig != null)
      {
        odServer = new ODServer(new ODConfig);
        odServer.initialize(  "TcDeleteNote.java" );
        System.out.println( "Logging on to " + argv[0] + "..." );
        odServer.logon( argv[0], argv[1], argv[2] );
        //----------
        // Open the specified folder and search with the default criteria
        //----------
        System.out.println( "Opening " + argv[3] + " folder..." );
        odFolder = odServer.openFolder( argv[3] );
        System.out.println( "Searching with default criteria..." );
        odFolder.search( );
        hits = odFolder.getHits( );
        System.out.println( "  Number of hits: " + hits.size( ) );

        //----------
        // Retrieve notes and delete the first one
        //----------
```

```
        if ( hits.size( ) > 0 )
        {
          odHit = (ODHit)hits.elementAt( 0 );
          System.out.println("Working with DocID " + odHit.getDocId());
          notes = odHit.getNotes( );
          numNotes1 = notes.size();
          System.out.println("  There are " + numNotes1 + " notes for the first hit" );

          if (numNotes1 > 0)
          {
            odNote = (ODNote)notes.elementAt( 0 );

            System.out.println("  Deleting the first note..." );

            odHit.deleteNote( odNote );

            // destroy hits
            hits.removeAllElements();

            // search again
            System.out.println( "Searching second time with default criteria..." );
            odFolder.search( );
            hits = odFolder.getHits( );
            System.out.println( "  Number of hits: " + hits.size( ) );

            // Retrieve notes again
            if ( hits.size( ) > 0 )
            {
              odHit = (ODHit)hits.elementAt( 0 );
              System.out.println("Working with DocID " + odHit.getDocId());
              notes = odHit.getNotes( );
              numNotes2 = notes.size();
              System.out.println("  There are " + numNotes2 + " notes for the first hit" );
            }
            else
              System.out.println( "No document - cannot list notes" );
          }
        }
        else
          System.out.println( "No document - cannot list notes" );

        if ((numNotes1 - numNotes2) == 1)
          System.out.println("\nSuccess!");
        else if (numNotes1 == 0)
          System.out.println("\nSuccess!");
        else
          System.out.println("\nFailed!");

        //----------
        // Cleanup
        //----------
        odFolder.close( );
        odServer.logoff( );
        odServer.terminate( );
        System.out.println( "" );
        System.out.println( "-------------------------------------------------" );
        System.out.println( "" );
        System.out.println( "Testcase completed - Ensure that the first note was deleted " );
        System.out.println( "by verifying with the Windows Client" );
        System.out.println( "" );
      }
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}
```

Updating a document

The following example demonstrates how to update a document.

This example uses ODServer , ODFolder , and ODCriteria methods to connect to a server using the specified userid and password, open the specified folder, set the search values for two search fields, set the Date search field to null, and search the folder. For the document that matches the query, ODHit methods are then used to update one or more database values.

This example demonstrates these ODServer methods:
- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates these ODFolder methods:
- getName
- getDisplayOrder
- getCriteria
- search
- closeinitialize

This example demonstrates these ODCriteria methods:
- setOperator
- setSearchValue

This example demonstrates these ODHit methods:
- getDisplayValue
- update

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Criteria name 1
- Search value 1
- Criteria name 2
- Search value 2
- New search value to replace search value 2

Example of updating a document:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcUpdate
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODCriteria odCrit;
    ODHit odHit;
    Hashtable hash;
    Vector hits;
    String[] display_crit;
    String line;
    String crit1;
    String crit2;
    String value1;
    String value2;
    String new_value;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 10 )
    {
      System.out.println( "usage: java TcUpdate <server> <userid> <password> <folder> <criteria1> <value1> <criteria2> <value2> <new value2>" );
      return;
    }

    try
    {
```

```
        System.out.println( "This testcase should:" );
        System.out.println( "  Logon to the specified server" );
        System.out.println( "  Open the specified folder" );
        System.out.println( "  Set the search values" );
        System.out.println( "  Search the folder" );
        System.out.println( "  For the first hit, change the value of 2nd specified criteria" );
        System.out.println( "    to the new value" );
        System.out.println( "" );
        System.out.println( "Using the Windows Client, ensure that the value has been changed." );
        System.out.println( "" );
        System.out.println( "------------------------------------------------------" );
        System.out.println( "" );


        //----------
        //Setup ODConfig object using defaults and initialize ODServer.
        //----------
        ODConfig odConfig = new ODConfig();
        odServer = new ODServer(new ODConfig);
        odServer.initialize( "TcUpdate.java" );

        //----------
        // Logon to specified server
        //----------
        System.out.println( "Logging on to " + argv[0] + "..." );
        odServer.logon( argv[0], argv[1], argv[2] );

        //----------
        // Open the specified folder and set the requested criteria
        //----------
        crit1 = argv[4];
        crit2 = argv[6];
        value1 = argv[5];
        value2 = argv[7];
        new_value = argv[8];
        System.out.println( "Opening " + argv[3] + " folder..." );
        odFolder = odServer.openFolder( argv[3] );
/*
        odCrit = odFolder.getCriteria( "Date" );
        odCrit.setOperator( ODConstant.OPEqual );
        odCrit.setSearchValue( "" );
*/
        odCrit = odFolder.getCriteria( crit1 );
        odCrit.setOperator( ODConstant.OPEqual );
        odCrit.setSearchValue( value1 );
        odCrit = odFolder.getCriteria( crit2 );
        odCrit.setOperator( ODConstant.OPEqual );
        odCrit.setSearchValue( value2 );

        //----------
        // Search the folder
        //----------
        System.out.println( "  Searching for " + crit1 + " = " + value1 + " and " + crit2 + " = " + value2 + "..." );
        hits = odFolder.search( );

        //----------
        // If there was at least one hit
        //----------
        if ( hits != null && hits.size( ) > 0 )
        {
          //----------
          // Display the values for the first hit
          //----------
          System.out.println( "    For first hit:" );
          odHit = (ODHit)hits.elementAt( 0 );
          System.out.println( "      DOCID = "+ odHit.getDocId() );
          line = "      ";
          display_crit = odFolder.getDisplayOrder( );
          for( j = 0; j < display_crit.length; j++ )
            line = line + display_crit[j] + " ";
          System.out.println( line );
          line = "      ";
          for ( j = 0; j < display_crit.length; j++ )
            line = line + odHit.getDisplayValue( display_crit[j] ) + " ";
          System.out.println( line );

          //----------
          // Create a hash table of existing critera/value pairs, except for critera 2
    // which will be set to the new value. Update the hit values
          //----------
          System.out.println( "    Replacing " + crit2 + " = " + value2 + " with " + crit2 + " = " + new_value );
          hash = new Hashtable( );
          for ( j = 0; j < display_crit.length; j++ )
    {
            if ( display_crit[j].equals( crit2 ) )
        hash.put( display_crit[j], new_value );
      else
        hash.put( display_crit[j], odHit.getDisplayValue( display_crit[j] ) );
    }
          odHit.updateValuesForHit( hash );
          System.out.println("New display values as set in ODHit");
          odHit = (ODHit)hits.elementAt( 0 );
          line = "      ";
          System.out.println( "      DOCID = "+ odHit.getDocId() );
          display_crit = odFolder.getDisplayOrder( );
          for( j = 0; j < display_crit.length; j++ )
            line = line + display_crit[j] + " ";
          System.out.println( line );

          line = "      ";



          for ( j = 0; j < display_crit.length; j++ )

            line = line + odHit.getDisplayValue( display_crit[j] ) + " ";

          System.out.println( line );
    }
     else
          System.out.println( "There were no hits" );

        //----------
        // Cleanup
        //----------
        odFolder.close( );
        odServer.logoff( );
        odServer.terminate( );
        System.out.println( "" );
        System.out.println( "------------------------------------------------------" );
```

```
      System.out.println( "" );
      System.out.println( "Testcase completed - Using the Windows Client," );
      System.out.println( "  ensure that the value has been changed." );
      System.out.println( "" );
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}
```

Changing a password

The following example uses the `ODServer` method `changePassword` to change the specified user's password to a new password. This example also uses `ODServer` methods to prepare for logon and log off.

This example demonstrates these `ODServer` methods:
- initialize
- logon
- changePassword
- logoff
- terminate

This example uses these run-time parameters:
- Server name
- User Id
- Password
- New password

Example of changing a password:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcChangePassword
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
 String server, userid, original_password, new_password;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcChangePassword <server> <userid> <password> <new password> <logon first> " );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the server using the specified password" );
      System.out.println( "  Change the password to the new password" );

      System.out.println( "  Logoff" );
      System.out.println( "  Logon to the server using the new password" );
      System.out.println( "  Change the password back to the original password" );
      System.out.println( "  Logoff" );
      System.out.println( "" );
      System.out.println( "If the testcase executes without exception, no further analysis" );
      System.out.println( "is required." );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Create the specified server
      //----------
   server = argv[0];
   userid = argv[1];
   original_password = argv[2];
   new_password = argv[3];
         int logon_first= Integer.parseInt(argv[4]);
```

```
      //----------
      //Setup ODConfig object using defaults and initialize ODServer.
      //----------
      ODConfig odConfig = new ODConfig();
      odServer = new ODServer(new ODConfig);
      odServer.initialize( "TcChangePassword.java" );

      if(logon_first == 1)
      {

         //----------
         // Logon to the server using the original password
         //----------
         System.out.println( "Logging on to " + server + " using original password..." );
         odServer.logon( server, userid, original_password );
      }
      else
      {
         System.out.println("Do NOT logon, but set the pre-req fields of UID,PWD,and Server");
         odServer.setUserId(userid);
         odServer.setPassword(original_password);
         odServer.setServerName(server);
      }
      //----------
      // Change to the new password and logoff
      //----------
      System.out.println( "Changing to new password..." );
      odServer.changePassword( new_password );
      System.out.println( "Current ODServer.password is " + odServer.getPassword() );
      System.out.println( "Logging off..." );
      odServer.logoff( );
      //----------
      // Logon to the server using the new password
      //----------
      System.out.println( "Logging on to " + server + " using new password..." );
      odServer.logon( server, userid, new_password );

      //----------
      // Change back to the original password and logoff
      //----------
      System.out.println( "Changing back to original password..." );
      odServer.changePassword( original_password );
      System.out.println( "Current ODServer.password is " + odServer.getPassword() );
      System.out.println( "Logging off..." );
      odServer.logoff( );

      //----------
      // Cleanup
      //----------
      odServer.terminate( );
      System.out.println( "" );
      System.out.println( "--------------------------------------------------" );
      System.out.println( "" );
      System.out.println( "Testcase completed successfully" );
      System.out.println( "" );
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}
```

Chapter 6. Java API reference

The documentation for the Java API is provided in standard Javadoc format with the ODWEK software.

Before you can view the documentation, you must install ODWEK software on the system and then extract the documentation files from the `ODApiDoc.zip` file in the `install/api` directory. (Where `install` is the installation directory for ODWEK software.) Use an extraction method that preserves the directory structure of the files in the archive.

To view the documentation, after extracting the files, open the `index.html` file with a Web browser.

Chapter 7. Configuring the sample applications

Important: If you are using the Java API to control ODWEK, see Chapter 5, "Deploying and programming Java API," on page 21 for information about setting up the system environment and running ODWEK applications.

This chapter explains how to customize the sample applications that are provided with ODWEK for the CGI program and the Java servlet.

- LOGON.HTM. This application supports users that are permitted to access several folders. Each user is defined to the OnDemand library server. After logging on to the server, ODWEK shows the user the list of folders that the user is permitted to open. "LOGON.HTM" contains instructions for customizing this application.
- CREDIT.HTM. This application supports casual use of OnDemand. The user is presented with search criteria for a specific folder. The OnDemand server name, userid and password, folder name, and folder fields are coded in the application. "CREDIT.HTM" on page 70 contains instructions for customizing this application.
- FCREDIT.HTM. A version of the CREDIT application that demonstrates the use of HTML frames.

After you modify the sample applications, publish the URL of each file so that users can link to them and access OnDemand. Each sample requires a different level of customization. There are complete instructions for customizing the CREDIT.HTM sample application. Use the instructions as a guide for customizing other applications that you may need.

Tip: In addition to modifying the sample applications, IBM recommends that you customize the TEMPLATE.HTM file for your organization. The TEMPLATE.HTM file contains user-defined content that ODWEK uses to display Web pages. See "TEMPLATE.HTM" on page 71 for important information about modifying this file.

LOGON.HTM

1. Copy the `logon.htm` file from the installation directory to the document root directory of the HTTP server. Table 2 lists the installation directories.

Table 2. Installation directory that contains the sample logon.htm file

Operating system	Installation Directory
AIX	/usr/lpp/ars/www/samples
HP-UX	/opt/ondemand/www/samples
Linux	/opt/ondemand/www/samples
Solaris	/opt/ondemand/www/samples
Windows	`X:\installation directory\samples`, where `X` is the installation drive and `installation directory` is the directory in which you installed the ODWEK software

2. For the CGI program, verify that the `logon.htm` file contains the following lines:

```
<h4>Please enter your logon information:</h4>
<FORM METHOD=POST ACTION="/arswww.cgi">
```

3. For the servlet, verify that the `logon.htm` file contains the following line:

```
<FORM METHOD=POST ACTION="/ArsWWWServlet">
```

CREDIT.HTM

Customize the CREDIT.HTM sample application by making a copy of the file for each folder that you want users to access. The name of the file should be the same as the name of the folder.

1. Edit the CREDIT.HTM file. (By default, this file is located in the `/usr/lpp/ars/www/samples` directory.)
2. Change or delete the background image specified in the <body> statement (line 11).
3. Optionally change the background color specified in the <body> statement (line 11).
4. Change or delete the product image specified in the <img> statement (line 12).
5. Replace the folder name specified in the <h1> statement (line 15).
6. Replace the text specified in the <p> statements (lines 17 through 25). Enter general instructions to the user.
7. Replace the CGI-BIN directory name specified in the <FORM> statement (line 29). Enter the name of the CGI-BIN directory that contains the ODWEK programs and files on the Web server.
8. Replace the value specified in the <input> statement (line 30). This is a comma separated string that contains the names of the folder display fields.
9. Replace the value specified in the <input> statement (line 31). This is the name of the folder.
10. Replace the value specified in the <input> statement (line 33). This is the maximum number of items displayed in the document list, regardless of the number of items that match the query.
11. Replace the server name specified in the <input> statement (line 35). This is the name of the OnDemand server with which ODWEK is to communicate. The supplied server name is `gunnar`.
12. If you want to sort items in the document list, verify the value specified in the <input> statement (line 36). Otherwise, delete line 36.
13. If you want to sort items in the document list, verify the value specified in the <input> statement (line 37). Otherwise, delete line 37.
14. Replace the value specified in the <input> statement (line 38). This is the OnDemand userid. The userid that you specify must have permission to open the folder and access application group data.
15. Optionally change the name of the template file specified in the <input> statement (line 39). OnDemand uses the template file to generate subsequent Web pages. The supplied template name is `template.htm`.
16. Modify lines 40 through 43 for the first folder search field.
    a. Type a name for the folder field in the <font> statement.
    b. Replace the value specified in the name field of the <input> statement with the actual folder field name.
    c. Replace the value specified in the value field of the <input> statement with the default search value.
17. Copy lines 40 through 43 and repeat step 16 for each additional folder search field.

18. Save your changes and close the text editor.

TEMPLATE.HTM

The TEMPLATE.HTM file is the default template file used by ODWEK to generate Web pages in response to the various product functions (such as Logon). You should replace this file with one that contains user-defined content. However, the template file must contain the following HTML comment line:

```
<!- - - AOI# Marker - - ->
```

The location of comment line determines where ODWEK program places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK.

By default, the template file is located in the directory named by the TEMPLATEDIR parameter in the ARSWWW.INI file. See "TEMPLATEDIR" on page 99 for details.

Your next step

After you have configured the sample applications, go to Chapter 8, "Installing and configuring viewing software," on page 73

Chapter 8. Installing and configuring viewing software

Overview

IBM provides viewers for the standard types of documents that can be retrieved from Content Manager OnDemand. The installation requirements vary, depending on the viewers that users need to use.

- To view line data documents, IBM recommends that you use the Java Line Data Viewer. The Java Line Data Viewer is an applet that is stored on the Web server. After an administrator enables the use of the Java Line Data Viewer, it is automatically loaded into memory on the workstation when the user selects to view a line data document. Verify that the LINEVIEWING parameter in the ARSWWW.INI file specifies the viewer that your users will be using.
- To view AFP documents, you can use the IBM Content Manager OnDemand AFP Web Viewer, the Adobe Acrobat viewer, or the Java AFP2HTML Viewer.
  - To view AFP documents with the IBM Content Manager OnDemand AFP Web Viewer, users must install it on the PC.
  - To view AFP documents with the Adobe Acrobat viewer, an administrator must install and configure either the Xenos transform or the AFP2PDF transform on the system and configure the ARSWWW.INI file. After an administrator enables the use of the transform, by default, the browser will attempt to start the Adobe Acrobat viewer when the user selects to view an AFP document. The user must obtain and install the Adobe Acrobat viewer on the PC.
  - To view AFP documents with the Java AFP2HTML Viewer, an administrator must install and configure the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms on the system and configure the ARSWWW.INI file. The Java AFP2HTML Viewer is stored on the Web server. After an administrator enables the use of the AFP2HTML applet, it is automatically loaded into memory on the PC when the user selects to view an AFP document.

  Verify that the AFPVIEWING parameter in the ARSWWW.INI file specifies the viewer that your users will be using.
- To view BMP, GIF, JPEG, PCX, and TIFF documents, IBM recommends that your users install the IBM Content Manager OnDemand Image Web Viewer on their PCs; otherwise, they should use some other viewer that handles these types of documents. (For example, most browsers have built-in viewers capable of viewing GIF and JPEG.) If your users decide to use some other viewer, make sure that an administrator changes the default MIME content type for these types of documents. Verify that the parameters in the MIMETYPES section of the ARSWWW.INI file specify the viewers that your users will be using.

To view other types of data, you may need to install other viewers. For example, to view PDF documents that are retrieved from the Content Manager OnDemand server, IBM recommends that you obtain and install the Adobe Acrobat viewer for the browsers used in your organization.

If you want to define one user ID in OnDemand for multiple users to log on to OnDemand—and you want each user to access only their own information—you must configure the system as follows:

1. provide a logon validation process as part of a Web application.

2. the logon validation process would take place before the query is sent to OnDemand.
3. use the results of a successful logon to provide the account number to OnDemand.
4. use the ODWEK APIs to send an SQL query to the OnDemand server. The SQL query would contain a specific account number.

Example: The user opens a "welcome to your company" home page. To access account information, the user must enter a user ID and PIN. These values are verified by your company's Web application, not OnDemand. After a successful logon, the Web application then presents the user with the account summary page. That page contains a link for viewing account statements. When the user clicks "view account statement", the Web application invokes the ODWEK APIs, including an SQL query that contains the account number that was derived from the logon process. The APIs log on to the OnDemand server using the user ID and password that you created for ODWEK connections to the server, search for and retrieve the user's statement, and send the document back to the browser. The browser launches the viewer that is appropriate for the type of data that is contained in the statement.

Requirements

For viewer requirements, see: http://www.ibm.com/support/docview.wss?rs=129 &uid=swg27016455 or search for 7016455 at http://www.ibm.com.

Installation

Tip: If you plan to distribute user-defined files with the AFP Web Viewer, then you should configure the AFP Web Viewer installation file to hold the user-defined files before your users begin installing the AFP Web Viewer. See “Distributing user-defined files” on page 75 for more information.

The viewers that are provided by IBM are installed using self extracting files. These files should be downloaded to the user's Windows workstation and run to install the appropriate viewer. If the user is running a browser while the installation is in progress, then the user must stop and restart the browser before the viewer can be used. The following viewer files can be found in the plugins subdirectory where ODWEK was installed::

- afpplgin.exe - IBM Content Manager OnDemand AFP Web Viewer - All languages including DBCS support
- afpplgin.zip - IBM Content Manager OnDemand AFP Web Viewer - Zip format for all languages include DBCS support
- imgplgin.exe - IBM Content Manager OnDemand Image Web Viewer - All languages

The installation process copies the viewer and its associated files to directories of the user's choice. The AFP Web Viewer requires approximately 3 MB of space on the PC. The Image Web Viewer requires approximately 2 MB of space on the PC. Remind your users to restart their browser if it is active during the installation process.

AFP Web Viewer

The following settings may be applied from logical views on the server to the AFP Web Viewer.

- Background Color. The following colors are supported. No other colors are supported.

  Green Bar (displayed with a white background)

  Green

  Red

  Yellow

  Black

  White

  Grey
- Image Color. The following colors are supported. No other colors are supported.

  Yellow

  Blue

  Red

  Magenta

  Green

  Cyan

  Default (should display as black)
- Zoom.

Selected Area Color is not applied to the AFP Web Viewer. The selected area is always displayed with white text and a black background.

Distributing user-defined files

You can distribute user-defined files with the IBM OnDemand AFP Web Viewer software that is supplied by IBM. For example, suppose that someone in your organization creates AFP font files for documents that are stored in OnDemand. You can distribute the font files with the AFP Web Viewer software. That way, when a user views an AFP document, the document will be displayed with the correct fonts.

To distribute user-defined files with the AFP Web Viewer, you must package the files into an installation file and store the installation file in a shared location. When a user runs the installation file, the Setup program automatically installs the AFP Web Viewer and the user-defined files on the user's workstation.

You can distribute the following types of user-defined files with the AFP Web Viewer:

- AFP font files. These files are copied to the FONT subdirectory of the AFP Web Viewer destination directory on the workstation.
- Adobe Type 1 font files. These files are copied to a directory specified by the user and installed in ATM by the Setup program.
- TrueType font files. These files are copied to the Windows FONTS directory and installed in Windows by the Setup program.
- Miscellaneous user-defined files. These files are copied to the AFP Web Viewer destination directory on the user's workstation.

Tip: The Setup program copies user-defined files to the workstation after the AFP Web Viewer files that are supplied by IBM. If you name a user-defined file the same as one of the files supplied by IBM, then the user-defined file will replace the file supplied by IBM. You can take advantage of this feature, for example, to distribute an updated FTDPORT2.INI file or to distribute IBM AFP font files that your organization has modified.

The following topics contain more information about configuring and distributing the AFP Web Viewer:

- Install the AFP Web Viewer files supplied by IBM
- Add subdirectories to hold user-defined files
- Store user-defined files in subdirectories
- Configure font files
- Build the AFP Web Viewer installation file
- Install the AFP Web Viewer on a user's workstation

Installing the AFP Web Viewer files

Most customers use one of two ways to distribute the viewer files from a server, depending on whether they plan to distribute user-defined files with the AFP Web Viewer:

- Standard Install. Use to distribute the AFP Web Viewer files supplied by IBM and to prepare for distributing user-defined files with the AFP Web Viewer. When an administrator installs the ODWEK software on the Web server, the installation files for the viewers are stored in a directory on the server. There should be an installation file (EXE) for each viewer and a ZIP archive file for the AFP Web Viewer. The administrator typically moves the installation files to the public directory on the server and creates a web page with the links to the files. A user installs a viewer by loading the web page into their browser and activating the link to the appropriate installation file.
- Custom Install for the AFP Web Viewer. Use to distribute user-defined files with the AFP Web Viewer.
  1. Set up the server for a Standard Install.
  2. Before any users actually install the viewer, obtain a copy of the AFP Web Viewer ZIP archive file.
  3. Extract the files from the ZIP archive file to an empty work directory.
  4. Add subdirectories to the work directory and store user-defined files in the directories. See “Adding subdirectories” on page 77 and “Storing user-defined files” on page 77 for details.
  5. If distributing user-defined Adobe Type 1 font files, then create a font configuration file. See “Configuring font files” on page 77 for details.
  6. After all of the directories and files have been configured, create a self-extracting EXE file for distribution. See “Building the AFP Web Viewer installation file” on page 78 for details.
  7. Replace the EXE file provided by IBM for a Standard Install with the self-extracting EXE file that you created.
  8. After an administrator completes steps 1 through 7, users can install the AFP Web Viewer and the user-defined files by loading the web page into their browsers and activating the link to the updated installation file.

Adding subdirectories

The user-defined files that you plan to distribute must be stored in the CUSTOM subdirectory tree under the main viewer installation directory. For example, you could name the main viewer installation directory \ONDEMAND\AFP32.

To configure the main viewer installation directory to hold user-defined files:

1. Create a CUSTOM directory under the main viewer installation directory. For example:

   ```
   \ondemand\afp32\custom
   ```

   Tip: The CUSTOM directory can hold other[1] user-defined files that you want to distribute to your users. The Setup program copies files from this directory to the AFP Web Viewer destination directory on the workstation.
2. Add one or more of the following subdirectories to the CUSTOM directory. The subdirectories you add depend on the type of user-defined files that you want to distribute to your users.
   - Create a FONT subdirectory under the CUSTOM directory to hold AFP font files (file types FNT and MAP). For example:

     ```
     \ondemand\afp32\custom\font
     ```

     The Setup program copies these files to the AFP Web Viewer FONT directory on the workstation.
   - Create a TYPEONE subdirectory under the CUSTOM directory to hold Adobe Type 1 font files (file types PFB and PFM) and the font configuration file. For example:

     ```
     \ondemand\afp32\custom\typeone
     ```

     The Setup program copies these files to a directory specified by the user and installs the fonts in ATM.
   - Create a TRUETYPE subdirectory under the CUSTOM directory to hold Windows TrueType font files (file type TTF). For example:

     ```
     \ondemand\afp32\custom\truetype
     ```

     The Setup program copies files from this directory to the Windows FONT directory and installs the fonts in Windows.

Storing user-defined files

After extracting the IBM-supplied installation files to the work directory and creating the CUSTOM directories, you can store the user-defined files in the individual subdirectories. For example, copy Adobe Type 1 font files (file types PFB and PFM) that you want to distribute to your users to the \ONDEMAND\AFP32\CUSTOM\TYPEONE directory.

Configuring font files

If you plan to distribute user-defined Adobe Type 1 font files to your users, then you must complete the following steps:

1. Store the user-defined Type 1 font files (file types PFB and PFM) in the TYPEONE subdirectory of the CUSTOM directory. See “Adding subdirectories” for more information.
2. Create a Type 1 font configuration file. The following information describes how to create the Type 1 font configuration file.

1. Other than AFP font files, Adobe Type 1 font files, and Windows TrueType font files.

The Type 1 font configuration file must be named ATM_INI.CFG and must be stored in the TYPEONE subdirectory of the CUSTOM directory. See "Adding subdirectories" on page 77 for more information about the distribution directories.

Each record (line) in the Type 1 font configuration file identifies one and only one user-defined Adobe Type 1 font that you want to distribute to your users. The format of a record is:

```
fontname=filename.PFM,filename.PFB
```

Where `fontname` is the name of the Type 1 font as it appears in the ATM Control Panel fonts list, `filename.PFM` is the name of the PFM file for the font, and `filename.PFB` is the name of the PFB file for the font. The following example shows a Type 1 font configuration file with two records:

```
Courier,BOLD=coub.pfm,coub.pfb
SonoranSansSerif_36,BOLDITALIC=c0a175z0.pfm,c0a175z0.pfb
```

The first record in the file identifies the font named `Courier,BOLD` and its PFM font file `coub.pfm` and PFB font file `coub.pfb`. The second record in the file identifies the font named `SonoranSansSerif_36,BOLDITALIC` and its PFM font file `c0a175z0.pfm` and PFB font file `c0a175z0.pfb`.

When a user runs a AFP Web Viewer installation file that contains user-defined Adobe Type 1 font files, the Setup program processes font files in the following way:

1. Copies all of the user-defined Adobe Type 1 font files (file types PFB and PFM) found in the TYPEONE directory to the destination directory. The user specifies the destination directory.
2. Verifies that two font files were copied for each font identified in the Type 1 font configuration file (ATM_INI.CFG). The name of the files copied to the workstation must match the names specified in the font configuration file.

   Important: If the names of the font files specified in the font configuration file do not match the names of the files copied to the workstation, the Setup program displays a warning message and does not install the font.
3. Adds path information for the PFB and PFM files, using the destination directory specified by the user.
4. Installs the fonts in ATM.

Building the AFP Web Viewer installation file

After you have finished creating directories and storing files in the CUSTOM directory tree, you must create an installation file that contains your user-defined files and the AFP Web Viewer files supplied by IBM. The installation file is usually named `Setup.exe`.

Several companies make software for packaging files and applications into a single, self-extracting AFP Web Viewer executable file for distribution. For example, the InstallShield Software Corporation offers a product called PackageForTheWeb.

Important: Software provided by other companies is not supported by IBM.

After you have obtained the packaging software, run it and follow the instructions provided to create a AFP Web Viewer installation file that contains your user-defined files and the AFP Web Viewer files supplied by IBM.

Installing the AFP Web Viewer on a user's workstation

After you set up the CUSTOM directory tree, build the AFP Web Viewer installation file, and replace the AFP Web Viewer installation file on the server, users can begin installing the AFP Web Viewer and the user-defined files. The next time a user activates the link to the AFP Web Viewer installation file from the server, the Setup program installs the AFP Web Viewer on the user's workstation and copies all of the user-defined files that you packaged with the AFP Web Viewer installation file to the user's workstation.

Mapping AFP fonts

AFP fonts that a document was created with need to be mapped to fonts that can be displayed using the AFP Web Viewer. ODWEK provides font definition files that map the IBM Core Interchange (Latin only) and compatibility fonts to TrueType fonts. The font definition files and font map files are stored in the FONT subdirectory in which the AFP Web Viewer files reside.

If your documents use fonts that are not defined to the AFP Web Viewer, if you or others in your organization have modified the IBM Core fonts, or if you or others in your organization have created AFP fonts, then you must define the fonts in the font definition files so that the AFP Web Viewer can correctly display the documents. Refer to the *AFP Workbench Technical Reference* for details about how to map AFP fonts, font definition files, and other technical information related to AFP and TrueType fonts.

Displaying AFP reports

The FTDPORT2.INI file, which resides in the AFP Web Viewer installation directory, contains modifiable parameters that can affect how AFP reports display. This section describes these parameters and their possible values.

- Tracing

  To create a trace file for the AFP Web Viewer, set `Trace=TRUE` in the Misc section of the FTDPORT2.INI file.

- Rules and lines

  If rules or lines do not display correctly when you view an AFP report, the problem might be due to display driver differences. Use another method for displaying rules. In the Misc section of the FTDPORT2.INI file, change:

  ```
  RuleFix=FALSE
  ```

  to:

  ```
  RuleFix=TRUE
  ```

- Text fidelity

  If fonts are not substituted correctly and the text alignment is not correct—especially if the `Text Fidelity` parameter is set to `Character`—it might be because your report was created with 300–pel metrics rather than 240–pel metrics. If you specify `240Fidelity=FALSE`, the report is displayed using 300–pel metrics. If you specify `240Fidelity=TRUE`, the report is displayed using 240–pel metrics. The default is 240–pel metrics.

- Print dialog box default

  When the Print dialog box displays, the default is to print the current page of the report. You can change the default to print all of the pages of the report by specifying `PrintAllPages=TRUE` in the Settings section of the FTDPORT2.INI file.

- User-defined page sizes

  You can define two page sizes for viewing reports that contain non-standard page sizes. These two user-defined page sizes are added to the list of other page

sizes that can be selected when viewing a report. To define the two page sizes, modify the following two lines in the FTDPORT2.INI file:

`PaperSize1=`*`width`*`,` *`length`*

`PaperSize2=`*`width`*`,` *`length`*

Specify the width and length, respectively, of each page in the report. All values must be in units of 1440ths of an inch.

- If the page size is in inches, multiple it by 1440.
- If the page size is in millimeters, multiply it by 56.7 and round the result to the nearest whole number.

If no values are set for `PaperSize1` and `PaperSize2`, the default page size for reports that contain non-standard page sizes is 8.5x11 inches.

- True Type fonts

  If you want to view reports using True Type fonts:

  1. If Adobe Type Manager (ATM) is installed on your PC, disable or remove it. If Type 1 fonts are installed, they must be removed.
  2. Add the following line to the Misc section of the FTDPORT2.INI file:

     ```
     TTONLY=TRUE
     ```

Displaying overlays

If a Content Manager OnDemand Thick Client view of an AFP data stream displays an overlay, and the ODWEK AFP Web Viewer does not display the overlay, the overlay resource probably was not found by the AFP Web Viewer.

To configure the AFP Web Viewer to display the overlay, specify the resource directory in the file named FTDPORT2.INI. Using a text editor, like Microsoft Notepad, open the file and find the entry named ResourceDataPath under [Preferences]. An example follows:

```
[Preferences]
DefaultView=DEFAULT
ViewDataPath=C:\Program Files\IBM\OnDemand AFP Web Viewer\Data
ResourceDataPath=C:\Program Files\IBM\OnDemand AFP Web Viewer\Resource
FontDataPath=C:\Program Files\IBM\OnDemand AFP Web Viewer\Font
```

The ResourceDataPath entry that is used for the OnDemand Client should match the same entry used for the AFP Web Viewer. Both the OnDemand Client and the AFP Web Viewer should have an FTDPORT2.INI file.

Important: External overlay resources are not downloaded with the AFP document. If the resource is external (if it is not stored in the same file as the AFP document), then the resource must be downloaded with the AFP document. If the resource is external, it must be stored in the directory specified by the ResourceDataPath parameter.

The AFP Web Viewer does not download overlays to the resource directory specified by the ResourceDataPath; therefore, if the resource cannot be downloaded to the client workstation by some other means, then the AFP data stream must be modified to include the resource so that the AFP document and the AFP resource reside in the same file.

Image Web Viewer

The following information applies when using the Image Web Viewer to view multi-page images.

Important: The following procedure requires that you edit the registry on the computer. You should not edit the registry unless it is absolutely necessary. If there is an error in the registry, the computer may not function properly. Before you proceed, you should make a backup copy of the registry and you should be familiar with how to restore the registry to the same version you were using when you last successfully started the computer. For instructions, see your Windows information.

For multi-page images, when the vertical scroll bar tab is dragged a small window will appear next to the tab. This window will display the page number corresponding to the tab position and the number of pages in the image. For example, a display of `5 / 10` indicates that there are ten pages in the image and that, if the tab is released, page number five will become the current page.

This behavior can be suppressed by a registry setting within this key:

HKEY_LOCAL_MACHINE\Software\IBM\OnDemand Image Web Viewer\Preferences

If the string value `PageNumberScroll` is set to `0` (zero), the page number window will not be displayed when the scroll bar tab is dragged.

Within the same registry key, if the string value `PageNumberToolbar` is set to `1` (one), page number information will be displayed on the toolbar for multi-page images. For example, a display of `3 / 5` indicates that there are five pages in the image and that page number three is the current page.

Java line data viewer

ODLineDataViewer2.jar is the Java line data viewer.

For Java line data viewer requirements, see http://www.ibm.com/support/docview.wss?rs=129&uid=swg27016455 or search for 7016455 at: www.ibm.com.

An additional parameter in the [2] file determines the location of the Java plugin installation file for Internet Explorer users that do not have the required version of the Java plugin installed on their workstations.

Table 3 on page 82 describes new parameters in the ARSWWW.INI file that support the Java line data viewer.

2. ARSWWW.INI

Table 3. Parameters in ARSWWW.INI File for Java Line Data Viewer

Parameter	Value	Comments
ODApplet.jre.path.IE	http://java.sun.com/getjava/installer.html	For Internet Explorer. Specifies to automatically download and install the latest version of the Java plugin from the java.sun.com Web site. See http://java.sun.com/getjava/install-windows.html for a preview of what happens when users automatically download and install the Java plugin. The user may need to restart the browser after installing the plugin.
	`<location>`	For Internet Explorer. Specifies the location of the Java plugin installation file within a company's intranet. The location must include a valid browser protocol, such as http, file, or ftp. For example: `file://shareName/java/plugins/plugin.exe` An administrator must download the Java plugin installation file and store it in the specified location. By specifying the location of the installation file, the browser will automatically install the Java plugin on the workstation. The user may need to restart the browser after the installation completes.

Table 3. Parameters in ARSWWW.INI File for Java Line Data Viewer (continued)

Parameter	Value	Comments

ODApplet.jre.version	`<version>`	For Internet Explorer. Specifies the version of the Java plugin to use. Must specify version 1.4 or later. Specify a major version number (for example, 1.4) to support any release of the plugin at that level (for example, 1.4.0, 1.4.0_03, 1.4.1_01). Specify a specific version number (for example 1.4.1_01) to support only that version of the Java plugin. Obtain valid version numbers from the java.sun.com Web site. For example:
		`1.4`
		
		or:
		`1.4.1_01`

The following shows an example of how to configure the ARSWWW.INI file to support Version 1.4 or later of the Java plugin. For Internet Explorer, users can automatically download and install the latest version of the Java plugin from the java.sun.com Web site. **Important:** Only users that do not have Version 1.4 or later of the Java plugin installed on their workstations will be prompted to download / install the plugin.

```
[DEFAULT BROWSER]
ODApplet.jre.path.IE=http://java.sun.com/getjava/installer.html
ODApplet.jre.version=1.4
```

Your next step

After you have installed the ODWEK software, configured the HTTP server, configured the Web application server, verified the ARSWWW.INI file, configured the sample applications, and installed the Web viewers, you should verify the installation before you begin using ODWEK. For verification procedures, see Chapter 9, “Verifying the installation,” on page 85.

Chapter 9. Verifying the installation

At this point, you should have completed all of the steps in the basic installation for ODWEK.

If you use the Java API, see Chapter 5, "Deploying and programming Java API," on page 21 for information about configuring the system and using the Java interpreter to run an ODWEK application.

You can verify that ODWEK is installed correctly by logging on to an OnDemand library server and opening a folder. If you are using the CGI program, go to "Verifying the CGI program." If you are using the Java servlet, go to "Verifying the servlet" on page 86.

Verifying the CGI program

You can verify the installation by performing the following steps:

Important: Before you begin, restart the HTTP server to initialize the system with the changes that you have made to the configuration files.

1. Verify the HOST, PORT, and PROTOCOL parameters in the `[@SRV@_default]` section of the `arswww.ini` file. The default location of the `arswww.ini` file is:

AIX	/usr/lpp/ars/www/cgi-bin
HP-UX	/opt/ondemand/www/cgi-bin
Linux	/opt/ondemand/www/cgi-bin
Solaris	/opt/ondemand/www/cgi-bin
Windows	x:\yyyyyyyy\BIN , where `x` is the installation drive and `yyyyyyyy` is the directory in which you installed the ODWEK software

   See "[@SRV@_DEFAULT]" on page 90. **Note:** The value of the PORT parameter in the ARSWWW.INI file is the port number on which the OnDemand library server is running, not the port number that the IBM HTTP Server listens for requests from the client (browser).
2. Start a browser.
3. On the Address line of the browser, type a URL that includes the OnDemand library server, HTTP port, and logon function. For example:

   ```
   http://odserver1.xyz.com:80/logon.htm
   ```

   Where `odserver1.xyz.com` is the value of the HOST parameter in the ARSWWW.INI file, `80` is the HTTP port, and `logon.htm` specifies the function that ODWEK should invoke. In this example, ODWEK will invoke the logon function to log on to the specified OnDemand library server. (The `logon.htm` file is one of the sample applications that are provided with ODWEK. See Chapter 7, "Configuring the sample applications," on page 69 for instructions about how to deploy the sample applications.)
4. If the system is configured correctly, ODWEK should display the Logon screen. Figure 6 on page 86 shows an example.

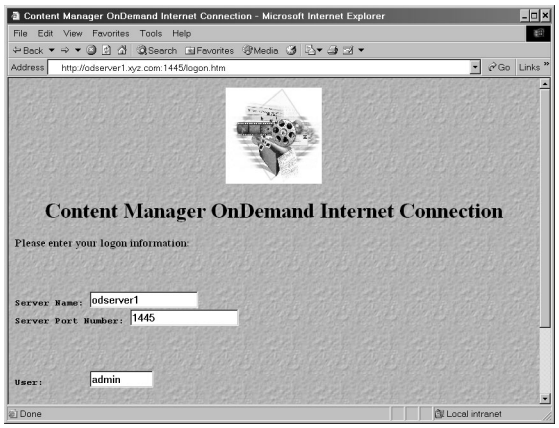

Figure 6. Logon Screen in ODWEK.

5. If the Logon screen did not appear, see "Troubleshooting" on page 87.
6. From the Logon screen, type a userid and password that is valid on the OnDemand library server. Click Submit to proceed to the Open a Folder screen. Figure 7 shows an example.

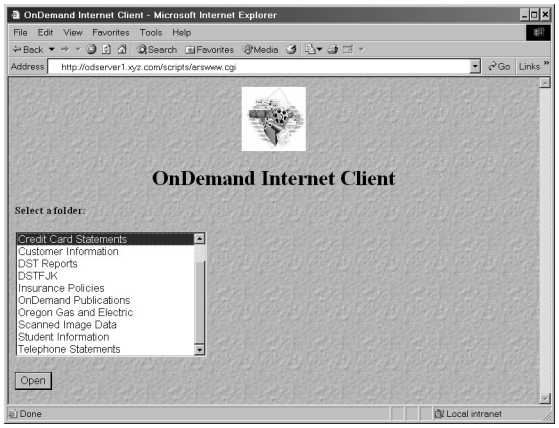

Figure 7. Open a Folder Screen

7. At this point, the basic installation is successful. However, you may need to continue the validation process by retrieving different types of documents to test any transforms that you may have integrated with ODWEK.

Verifying the servlet

Note: Before you proceed, if you have not already done so, stop and restart the HTTP server and the Web application server.

To verify that the servlet works correctly, start a Web browser and open the servlet. Specify the location of the servlet. For example:

```
http://server/od/arswww
```

Where `server` is the hostname of the system on which you are deploying the servlet, `od` is the Content Root that you specified in Step 13 on page 151 and `arswww` is the Servlet Mapping that you specified in Step 27 on page 152.

If you see a Web page with the text `Internet Connection Version 8.4.1.x` and `The argument '_function' was not specified`, then the deployment was successful.

Troubleshooting

This section describes the typical errors when attempting to verify the installation and contains possible solutions for those errors.

Table 4. Troubleshooting ODWEK Installation.

Problem	Solution
The Logon screen did not appear.	If the Logon screen did not appear, the most likely cause is that the mapping rules for the `logon.htm` file are incorrect. The mapping rules are specified in the `httpd.conf` file. See your HTTP server information for directions. If the mapping rules are correct: 1. Verify the permissions of the samples directory: **AIX** /usr/lpp/ars/www/samples **HP-UX** /opt/ondemand/www/samples **Linux** /opt/ondemand/www/samples **Solaris** /opt/ondemand/www/samples **Windows** x:\yyyyyyyy\SAMPLES, where `x` is the installation drive and `yyyyyyyy` is the directory in which you installed the ODWEK software 2. Verify the permissions of the `logon.htm` file in the samples directory: **AIX** /usr/lpp/ars/www/samples **HP-UX** /opt/ondemand/www/samples **Linux** /opt/ondemand/www/samples **Solaris** /opt/ondemand/www/samples **Windows** x:\yyyyyyyy\SAMPLES, where `X` is the installation drive and `yyyyyyyy` is the directory in which you installed the ODWEK software 3. Verify that the HTTP server is operational. Make the necessary corrections, then restart the HTTP server and the Web application server and try the logon process again.
Error 404, file not found	If the Logon screen appeared, but you received an Error 404 message when you tried to log on to the server, verify the file mappings in the `httpd.conf` file. See your HTTP server information for directions. Make the necessary corrections, restart the HTTP server and the Web application server and try the logon process again.

Table 4. Troubleshooting ODWEK Installation. (continued)

Problem	Solution
Error 500, Server Error	If the Logon screen appeared successfully, but you received an Error 500 message, you are most likely running the Java servlet version of ODWEK. Verify that the `ServerInit` statement in the Servlet Support section of the `httpd.conf` file identifies the name and location of the `was.conf` file. In most installations, the `was.conf` file will be in the IBM HTTP server root directory. See your HTTP server information for directions. Make the necessary corrections, then restart the HTTP server and the Web application server and try the logon process again.

Appendix A. Deploying the CGI program

This section explains how to deploy the CGI version of ODWEK.

Before you begin

Before you deploy the CGI program:

- Make sure that you have completed the software installation. See Chapter 4, "Installing ODWEK," on page 15.
- Make sure that you have the current version of the IBM HTTP server installed, configured, and operating on the system.

Copying CGI program files

To copy the CGI program files:

1. Copy the ARSWWW.CGI file to a directory on the system that has been designated as executable and that can contain CGI programs. For example:

   ```
   <server_root>/cgi-bin
   ```

2. For Windows systems, copy the following files to the directory in which you copied the ARSWWW.CGI file:

   ```
   icudt36.dll
   icuuc36.dll
   icuin36.dll
   ```

3. Copy the following files to the directory in which you copied the ARSWWW.CGI file:

   ```
   ARSWWW.INI
   AFP2HTML.INI
   AFP2PDF.INI
   ```

Specifying the ARSWWW.INI file

Important: The `ARSWWW.INI` file is used with CGI program or Java servlet only. Do not use it with the Java API interface.

After you have deployed the CGI program, configure the ODWEK initialization file for your operating environment, that is, the `ARSWWW.INI file`.

The ARSWWW.INI file is an ASCII text file that contains parameters that are read by ODWEK programs (such as the CGI program or the Java servlet). You specify each parameter on a separate line using the following format: `PARAMETER=value`. For example:

```
AFPVIEWING=plugin
CACHEDIR=/ars/www/cache
LANGUAGE=ENU
```

The parameters in the ARSWWW.INI file are grouped into sections. You specify the beginning of a section using a section header, in the following format: `[sectionHeader]`. You specify the parameters for a section after the section header. For example:

```
[@SRV@_odserver1]
HOST=odserver1.xyz.com
PORT=1445
PROTOCOL=0
```

An example ARSWWW.INI configuration file is provided with the product. The example configuration file provides a set of the most commonly used values. "Example ARSWWW.INI file" on page 120 shows the example.

The sections and parameters for the ARSWWW.INI file are as follows:

[@SRV@_DEFAULT]

The default server section. You can use the default server section to specify parameters that are common to the Content Manager OnDemand servers with which ODWEK will communicate. The parameters and values that you specify in this section will be used unless you specify them in a server section.

This section has a global scope for all servers, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section may contain the following parameters:

PORT

The TCP/IP port number that the Content Manager OnDemand server listens on. This is the port number that ODWEK will connect to. Before using any port number, verify with your TCP/IP administrator that the port is available. You can ask that a port be reserved for ODWEK by specifying it in the TCP/IP services file. Unless you specify otherwise, the Content Manager OnDemand server will attempt to use port number `1445`.

You can specify this parameter once in the default section. When using the Logon API, you can override the specified port number with the `_port` parameter.

This parameter is optional.

Example:

```
[@SRV@_DEFAULT]
PORT=1444
```

PROTOCOL

The networking protocol that is used to communicate between the Content Manager OnDemand server and ODWEK. The only valid value is `0` (zero), for TCP/IP.

You must specify this parameter once in the default section.

This parameter is optional. If not specified, a value of `0` (zero) is used.

Example:

```
[@SRV@_DEFAULT]
PROTOCOL=0
```

[@SRV@_server]

A server section. You must specify one server section for each Content Manager OnDemand server with which ODWEK will communicate. A server section contains the parameters and values for a specific server. The section header must include the string that identifies the server. The parameters specified in a server section override the parameters found in the default server section.

You must specify one server section for each server.

This section is optional. However, if not specified, the system will use the values that are specified (or defaulted to) in the [@SRV@_DEFAULT] section.

This section may contain the following parameters:

HOST

The name of the Content Manager OnDemand server. You can specify the TCP/IP address, host name alias, or fully-qualified host name of the server.

You must specify this parameter once in the server section.

This parameter is required.

Example:

```
[@SRV@_odserver1]
HOST=odserver1.xyz.com
```

PORT

The TCP/IP port number that the Content Manager OnDemand server listens on. This is the port number that ODWEK will connect to. If you do not specify the PORT parameter, then the server uses the port number that is specified (or defaulted to) in the default server section.

You can specify this parameter once in the server section. When using the Logon API, you can override the specified port number with the `_port` parameter.

This parameter is optional. If not specified, the value that is specified (or defaulted to) in the [@SRV@_DEFAULT] section is used.

Example:

```
[@SRV@_odserver1]
PORT=1446
```

PROTOCOL

The networking protocol that is used to communicate between the Content Manager OnDemand server and ODWEK. The only valid value is `0` (zero), for TCP/IP.

You can specify this parameter once in the server section.

This parameter is optional. If not specified, the value that is specified (or defaulted to) in the [@SRV@_DEFAULT] section is used.

Example:

```
[@SRV@_odserver1]
PROTOCOL=0
```

[CONFIGURATION]

The CONFIGURATION section contains parameters that are used by ODWEK on the Web server.

This section has a global scope, and you specify it only once in the ARSWWW.INI file.

This section is required.

This section can contain the following parameters:

APPLETCACHEDIR

Specifies the directory in which the Line Data applet and the AFP2HTML applet temporarily store documents. The directory can be local to the user's workstation or on a network drive. All users must have write access to the specified directory.

Example:

```
[Configuration]
APPLETCACHEDIR=n:\temp
```

Notes:

1. The APPLETCACHEDIR parameter has a global scope.
2. The APPLETCACHEDIR parameter is optional. However, if this parameter is not specified, the applets will attempt to store documents in the Java working directory.
3. If the specified directory does not exist, the applets will attempt to store documents in the Java working directory.
4. The applet removes a document from the cache directory when the user leaves the applet (for example, closes the document).

APPLETDIR

Identifies the directory that contains the Line Data and AFP2HTML applets.

Notes:

1. You can specify a directory name or an *alias*:
   - If you specify a directory name, the directory must be relative to the `ServerRoot` directory. For example, if you specify `appletdir=applets` and the `ServerRoot` directive is set to `/usr/lpp/ars/www`, then the applets must exist in the `/usr/lpp/ars/www/applets` directory. The `ServerRoot` directive is defined in the Web server configuration file.
   - If you specify an alias, then it must be defined in the Web server configuration file. For example, if you specify `appletdir=/applets/`, then the Web server configuration file must have an alias for `/applets/`. The alias must be set to the full path name of the directory on the server. For example:

     ```
     Alias     /applets/     /usr/lpp/ars/www/applets
     ```

     Note: Appendix A, "Deploying the CGI program," on page 89 and Appendix B, "Deploying the Java servlet," on page 149 contain examples of Web server configuration files.
2. Verify the permissions of the directory that you specify. The processes that run ODWEK programs must read the applet directory.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is required.

Example:

```
[CONFIGURATION]
APPLETDIR=applets
```

CACHEDIR

Specifies the directory on the Web server in which ODWEK temporarily stores (*caches*) server data and optionally, documents (see "CACHEDOCS").

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is required.

Example:

```
[CONFIGURATION]
CACHEDIR=/ars/www/cache
```

Notes:

1. See Chapter 3, "Installation requirements," on page 13 for information about the cache directory.
2. Verify the permissions of the directory that you specify. The processes that run ODWEK programs must write to and read from the cache storage directory.
3. For performance reasons, the directory and mount point that you specify for the CACHEDIR parameter should be different than those that you specify for the TEMPDIR parameter.

CACHEDOCS

Determines whether ODWEK temporarily stores (*caches*) documents on the Web server. Cache storage can increase the speed with which previously viewed documents are retrieved from the server. The default value is `0` (zero), which means that cache storage for documents is not enabled. Specify a 1 (one) to enable cache storage for documents. If you enable cache storage for documents, verify the directory in which ODWEK caches documents (see "CACHEDIR") and the amount of disk space reserved for cache storage (see "CACHESIZE" on page 94).

Note: IBM recommends that you always enable cache storage for documents when you use the Microsoft Internet Explorer browser and the AFP Web Viewer or the Image Web Viewer.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional. **Note:** If usage patterns are such that a document is viewed only once by a single user, then caching may degrade the performance of the system.

Example:

```
[CONFIGURATION]
CACHEDOCS=1
```

CACHEMAXTHRESHOLD

Determines when ODWEK begins deleting data and documents from cache storage. ODWEK begins deleting data and documents when the percentage of disk space used in cache storage is equal to or greater than the value specified. The

default value is 80 (eighty percent). ODWEK deletes items from cache storage until a threshold is reached (see "CACHEMINTHRESHOLD").

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional.

Example:

```
[CONFIGURATION]
CACHEMAXTHRESHOLD=80
```

CACHEMINTHRESHOLD

Determines when ODWEK stops deleting data and documents from cache storage. ODWEK stops deleting data and documents when the percentage of disk space used in cache storage is less than or equal to the value specified. The default value is 40 (forty percent). ODWEK begins deleting items from cache storage when a threshold is reached (see "CACHEMAXTHRESHOLD" on page 93).

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional.

Example:

```
[CONFIGURATION]
CACHEMINTHRESHOLD=40
```

CACHESIZE

The amount of disk space that ODWEK can use to temporarily store (*cache*) data and documents on the Web server. Specify the value in megabytes. The default value is 10 (ten megabytes).

Note: To enable cache storage for documents, see "CACHEDOCS" on page 93.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional. However, when caching documents, the more disk space that you allocate, the more documents ODWEK can store on the Web server. Generally, this can increase the speed with which ODWEK sends previously viewed documents to users. **Note:** If usage patterns is such that a document is viewed only once by a single user, then caching may degrade the performance of the system.

Example:

```
[CONFIGURATION]
CACHESIZE=1024
```

CACHEUSERIDS

Specifies a comma separated list of Content Manager OnDemand userids for which ODWEK uses data from cache storage to complete the logon process. For the specified userids, multiple logon attempts will bypass the standard Content Manager OnDemand logon processing, except if the data is not in cache storage or

if the Inactivity Time Out value (see the system parameters on the Content Manager OnDemand server) is reached. Separate each userid with the comma character.

Notes:

1. If the userid is case sensitive on the server (see the system parameters on the Content Manager OnDemand server), then you must specify the userid exactly as it was defined to OnDemand.
2. The userids listed in the CACHEUSERIDS list can access only those folders whose names and other information are currently in cache storage. The users will not be able to access folders created after they log on to a Content Manager OnDemand server. To allow a userid listed in the CACHEUSERIDS list to access a new folder, either delete the user's name from the CACHEUSERIDS list or purge the cache.
3. To specify that ODWEK should use data from cache storage for all Content Manager OnDemand users, specify `CACHEUSERIDS=*`.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional.

Example:

```
[CONFIGURATION]
CACHEUSERIDS=oduser1,oduser2,oduser3
```

CODEPAGE

Identifies the code page of the Content Manager OnDemand database. The default code page is the code page of the Web server.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section. For z/OS, the ASCII equivalent of the database codepage should be specified.

This parameter is optional. However, if the Web server is running in a different code page than the database, then you must specify the CODEPAGE parameter. See Appendix D, “National language support,” on page 159 for more information.

Example:

```
[CONFIGURATION]
CODEPAGE=943
```

Tip: To create a new code page for line data on an AIX web serve, follow these steps:

1. Enter a new line to the /usr/lpp/ars/locale/arscpcs.cfg file referencing the new code page number.
2. Copy the conversion file IBM-xxx to the /usr/lpp/ars/locale/uconvtab directory.

DOCSIZE

When retrieving documents from the Content Manager OnDemand server, determines the maximum size (in bytes) of a document that can be written directly to memory instead of first writing the document to disk. Any document that is less than or equal to the value specified will be written directly to memory. Any document that exceeds the specified value will first be written to disk and then

read from disk into memory before it is delivered to the browser. A lower value may conserve system resources, while a higher value will improve viewing performance. The range is from 0 (zero) to *n* bytes, where *n* is the amount of available memory on the system. A value of zero defaults the size to 1 MB. If this parameter is not specified or if the value is not defined or recognized, the size defaults to 1 MB.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional.

Example:

```
[CONFIGURATION]
DOCSIZE=524287
```

IMAGEDIR

Identifies the directory that contains image files used by ODWEK. The default value is /usr/lpp/ars/www/images.

Notes:

1. ODWEK concatenates the value that you specify with the file names found on HTML image tags. For example, if you specify:

   ```
   imagedir=images
   ```

   Then the HTML image tag for the View Document function will appear in the output as follows:

   ```
   <IMG SRC="images/odic_vd.gif">
   ```

2. You can specify a directory name or an *alias*:
   - If you specify a directory name, then the directory must be relative to the ServerRoot directory. For example, if you specify imagedir=images and the ServerRoot directory is set to /usr/lpp/ars/www, then the images must exist in the /usr/lpp/ars/www/images directory. The ServerRoot directory is set in the Web server configuration file.
   - If you specify an alias, then the alias must be defined in the Web server configuration file. For example, if you specify imagedir=/images/, then the Web server configuration file must have an alias for /images/. The alias must be set to the full path name of the directory on the server. For example:

     ```
     Alias     /images/     /usr/lpp/ars/www/images
     ```

     Tip: Appendix A, "Deploying the CGI program," on page 89 and Appendix B, "Deploying the Java servlet," on page 149 contain examples of Web server configuration files.
3. Verify the permissions of the directory that you specify. The processes that run ODWEK programs must read the image directory.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is required.

Example:

```
[CONFIGURATION]
IMAGEDIR=/usr/lpp/ars/www/images
```

LANGUAGE

Identifies the language in which ODWEK displays messages. The default language is English (ENU). ODWEK supports the following languages:

Language	Value
Brazilian Portuguese	PTB
Canadian French	FRC
Croatian	HRV
Czech	CZE
Danish	DAN
Finnish	FIN
French	FRA
German	DEU
Greek	ELL
Holland Dutch	NLD
Hungarian	HUN
Italian	ITA
Japanese	JPN
Korean	KOR
Norwegian	NOR
Polish	PLK
Russian	RUS
Simplified Chinese	CHS
Slovak	SKY
Slovenian	SLO
Spanish	ESP
Swedish	SVE
Traditional Chinese	CHT

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional.

Example:

```
[CONFIGURATION]
LANGUAGE=JPN
```

ServerFind

Determines whether server-based text search is active or inactive.

When the ServerFind parameter is set to `0` (zero), the Line Data applet on the user's workstation downloads each segment of the large object, searches in the large object, and builds a list of locations where the desired text is found. When the search is complete, the list is displayed to the user. When the user selects a location from the list, that page in the document is displayed, and the specified line of text is highlighted.

The advantages and disadvantages of the above process are:

- **Advantages:** This process offloads the work to the user's workstation, and all segments in the document are immediately available for display or more text searching.
- **Disadvantages:** The search is slow because each segment must be downloaded. It also increases network traffic.

If you set the ServerFind parameter to 1 (one), the large object text search occurs on the OnDemand server, and is not done by the Line Data Applet. When the server completes the search, a list of locations where the desired text was found is sent to the Line Data Applet and displayed to the user. When the user selects a location from the list, the segment containing that page in the document is downloaded and displayed, and the specified line of text is highlighted.

The advantages and disadvantages of the above process are:

- **Advantages:** The search is quicker because the segments are not downloaded. There is less network traffic, and only the segments of interest are downloaded.
- **Disadvantages:** It is more work for the OnDemand server and does not reduce search time, because the server must retrieve the document and search it each time.

This parameter is optional. If not specified, the default value is 0 (zero; inactive).

Example:

```
[CONFIGURATION]
ServerFind=1
```

ShowSearchString

Determines whether the Auto Find function is active or inactive. The Auto Find function supports transaction and text searching of line data documents from the Java line data viewer. The Auto Find function will automatically find and highlight the specific line in any document that matches the search criteria specified by the user.

When the Auto Find function is activated and a user performs either a transaction or text search and opens a document from the resulting document list, the system automatically searches the text of the document for the specified search criteria. If the search criteria is found, the line containing the search criteria is highlighted; otherwise the appropriate message is displayed. When the user opens another document for viewing (or reopens a previously viewed document), the search is performed again.

To activate the Auto Find function, set the ShowSearchString parameter to 1 (one). To deactivate the Auto Find function, set the ShowSearchString parameter to 0 (zero).

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional. If not specified, the default value is 0 (zero; inactive).

Example:

```
[CONFIGURATION]
ShowSearchString=1
```

TEMPDIR

Use to specify the directory in which ODWEK will store temporary files.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional. If not specified, the directory specified for CACHEDIR is used.

Example:

```
[CONFIGURATION]
TEMPDIR=/ars/www/tmp
```

Notes:

1. See Chapter 3, "Installation requirements," on page 13 for information about the temporary directory.
2. Verify the permissions of the directory that you specify. The processes that run ODWEK programs must write to and read from the cache storage directory.
3. For performance reasons, the directory and mount point that you specify for the TEMPDIR parameter should be different than those that you specify for the CACHEDIR parameter.

TEMPLATEDIR

Identifies the directory that contains the HTML template files. ODWEK uses the template files to generate Web pages in response to the various product functions (such as Logon, Search, Retrieve Document, and so forth). By default, ODWEK retrieves the template files from the `usr/lpp/ars/www/samples` directory.

Important: Verify the permissions of the directory that you specify. The processes that run ODWEK programs must read the template directory.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section.

This parameter is optional.

Example:

```
[CONFIGURATION]
TEMPLATEDIR=/usr/lpp/ars/www/samples
```

[SECURITY]

The SECURITY section contains the security parameters that are used by ODWEK on the Web server.

This section has a global scope, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

REPORTSERVERTIMEOUT

Use to specify whether ODWEK should report that the session with the Content Manager OnDemand server timed out. The Inactivity Time Out parameter on the Content Manager OnDemand server determines when the server ends a session

with an inactive user. To specify that ODWEK should use the Inactivity Time Out parameter to report timeouts, set the REPORTSERVERTIMEOUT parameter to 1 (one). This is a reporting mechanism only. The browser continues to interact with ODWEK and might reestablish sessions to the server. If the same user ID is active in multiple browsers, this parameter might not function properly.

This parameter has a global scope, and you specify it once only in the SECURITY section.

This parameter is optional. If it is not specified or set to 0 (zero), ODWEK does not use the Inactivity Time Out parameter, in other words, ODWEK does not report that an inactive user session has ended. For more information about the Inactivity Time Out parameter, see the administrative client online help.

If the REPORTSERVERTIMEOUT parameter is set to 1 (one), ODWEK updates a timestamp after each transaction against a server. When the Inactivity Time Out period expires, REPORTSERVERTIMEOUT reports that the user's session with the Content Manager OnDemand server has ended. If you use this option, then the temporary files must not be purged from ODWEK cache while the ODWEK instance is running. Doing so prevents ODWEK from maintaining the timing of the client requests.

Example:

```
[SECURITY]
REPORTSERVERTIMEOUT=1
```

UPDATETIMESTAMP

Use to specify that ODWEK should update a timestamp after each transaction against a server. If an Inactivity Time Out value is set, that value is compared to the time elapsed since the user's most recent transaction. The intent is to avoid unnecessary additional logons. To specify that ODWEK should update a timestamp after each transaction, set the UPDATETIMESTAMP parameter to 1 (one). If the same user ID is active in multiple browsers, this parameter might not function properly.

This parameter has a global scope, and you specify it once only in the SECURITY section.

This parameter is optional. If it is not specified or set to 0 (zero), and REPORTSERVERTIMEOUT is not set to 1 (one), ODWEK does not update a timestamp after each transaction against a server. If an Inactivity Time Out value is set, that value is compared to the time elapsed since the logon for the user rather than since the user's most recent transaction. As a result, there might be unnecessary additional logons.

For more information about the Inactivity Time Out parameter, see the administrative client online help.

The UPDATETIMESTAMP and REPORTSERVERTIMEOUT parameters are similar. If they are set to 1 (one), both update a timestamp after each transaction against a server. The difference occurs when the Inactivity Time Out period expires. REPORTSERVERTIMEOUT reports an error. UPDATETIMESTAMP performs a new logon for the user and does not report an error. If neither parameter is set to 1 (one), a timestamp is not updated and the Inactivity Time Out value is compared to the time elapsed since logon.

Example:

```
[SECURITY]
UPDATETIMESTAMP=1
```

SERVERACCESS

Specifies a comma separated list of the Content Manager OnDemand servers that ODWEK can access. If specified, the users that use ODWEK and the programs that use the APIs are permitted to access only those servers that are listed. You can specify the host name alias of the server, or the TCP/IP address or fully qualified host name of the server.

This parameter has a global scope, and you specify it only once in the SECURITY section.

This parameter is optional. If you do not specify this parameter (or you specify a blank list), then ODWEK can access any Content Manager OnDemand server.

Example:

```
[SECURITY]
SERVERACCESS=odserver1,odserver2
```

[AFP2HTML]

The AFP2HTML section contains the parameters that are used by the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms. The Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms converts AFP documents and resources into HTML documents that can be displayed with the AFP2HTML applet.

Notes:

1. To convert AFP documents to HTML documents, an administrator must obtain the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms service offering from IBM and install and configure it on the server. See your IBM representative for more information about the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms service offering.
2. To convert documents with the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms, you must specify the `AFPVIEWING=HTML` parameter in the DEFAULT BROWSER section (or other browser sections). See "AFPVIEWING" on page 111 for details. (If you plan to use the Retrieve Document API, then you should specify the `_afp=HTML` parameter. See "Retrieve Document" on page 139 for details.)
3. By default, ODWEK uses the AFP2HTML applet to view converted documents. If a converted document was stored in Content Manager OnDemand as a large object, then the AFP2HTML applet provides controls to help users easily move to any page in the document.

This section has a global scope, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

CONFIGFILE

The configuration file that contains the options used by the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms to convert AFP documents and resources into HTML data, fonts, and images that can be viewed with the AFP2HTML applet. See the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms documentation for details about the options that you can specify in the configuration file.

This parameter has a global scope, and you specify it only once in the AFP2HTML section.

This parameter is optional.

Example:

```
[AFP2HTML]
CONFIGFILE=afp2html.ini
```

INSTALLDIR

The directory that contains the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms programs, configuration files, and mapping files. Specify the full path name of the directory on the Web server.

Important: Verify the permissions of the directory that you specify. The processes that run ODWEK programs must read the install directory.

This parameter has a global scope, and you specify it only once in the AFP2HTML section.

This parameter is optional.

Example:

```
[AFP2HTML]
INSTALLDIR=/usr/lpp/ars/www/bin/afp2web
```

[AFP2PDF]

The AFP2PDF section contains the parameters that are used by the IBM AFP2PDF Transform. The AFP2PDF Transform converts AFP documents and resources into PDF documents that can be viewed with the Adobe Acrobat viewer.

Notes:

1. To convert AFP documents to PDF documents, an administrator must obtain the AFP2PDF Transform service offering from IBM and install and configure it on the Web server. See your IBM representative for more information about the AFP2PDF Transform service offering.
2. To convert documents with the AFP2PDF Transform, you must specify the `AFPVIEWING=PDF` parameter in the DEFAULT BROWSER (or other browser sections). See “AFPVIEWING” on page 111 for details. (If you plan to use the Retrieve Document API, then you should specify the `_afp=PDF` parameter. See “Retrieve Document” on page 139 for details.)
3. By default, ODWEK uses the Adobe Acrobat viewer to view converted documents. You must obtain the Adobe Acrobat viewer for the browsers that are used in your organization.

This section has a global scope, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

CONFIGFILE

The configuration file that contains the options used by the AFP2PDF Transform to convert AFP documents and resources into PDF documents that can be viewed with the Adobe Acrobat viewer. See the AFP2PDF Transform documentation for details about the options that you can specify in the configuration file.

This parameter has a global scope, and you specify it only once in the AFP2PDF section.

This parameter is optional.

Example:

```
[AFP2PDF]
CONFIGFILE=afp2pdf.ini
```

INSTALLDIR

The directory that contains the AFP2PDF Transform programs, configuration files, and mapping files. Specify the full path name of the directory on the Web server.

Note: Verify the permissions of the directory that you specify. The processes that run ODWEK programs must read the install directory.

This parameter has a global scope, and you specify it only once in the AFP2PDF section.

This parameter is optional.

Example:

```
[AFP2PDF]
INSTALLDIR=/usr/lpp/ars/www/bin/afp2pdf
```

USEEXECUTABLE

Determines whether ODWEK starts the AFP2PDF Transform by using the shared library or the executable. The default value is `0` (zero) and means that ODWEK uses the shared library. If you experience problems running the transform from the shared library, then set this parameter to `1` (one).

This parameter has a global scope, and you specify it only once in the AFP2PDF section.

This parameter is optional.

Example:

```
[AFP2PDF]
USEEXECUTABLE=1
```

[MIMETYPES]

The MIMETYPES section identifies the Multipurpose Internet Mail Extension (MIME) content type for documents that will be retrieved from the Content

Manager OnDemand server. The browser uses the MIME content type to format and display the document, to choose the correct applet or viewer to open the document, or to start a user-defined program to open the document.

Notes:

1. The MIMETYPES section should contain a `parameter=value` pair for each type of document that you plan to retrieve from the Content Manager OnDemand server. The `parameter` identifies the data type of the document in Content Manager OnDemand. (This is the data type that is assigned to the Content Manager OnDemand application on the View Information page.) The *value* determines the program that is started to open the document. The *value* is case sensitive.
2. In the example ARSWWW.INI file (see "Example ARSWWW.INI file" on page 120), the MIMETYPES section contains a parameter for each of the standard data types supported by Content Manager OnDemand (AFP, BMP, EMAIL, GIF, JFIF, LINE, PCX, PDF, and TIFF).
3. In addition to the standard data types, Content Manager OnDemand also supports user-defined data types. A user-defined data type can identify any other type of data that you want to store on the system. Before users can view documents that have a user-defined data type, you must add a parameter to the MIMETYPE section. The parameter must identify the MIME content type of the data and the file extension that was specified for the Content Manager OnDemand application on the View Information page. The file extension must also be registered with the operating system on the PC. For example, suppose you define an application to store Lotus WordPro documents in Content Manager OnDemand and you specify the file extension as LWP on the application View Information page. To configure the system to recognize documents retrieved from the application, add the following parameter to ARSWWW.INI file:

   ```
   [MIMETYPES]
   LWP=application/lwp
   ```

   Then, when a user retrieves a document from the application, ODWEK sets the MIME content type to `application/lwp` and the system starts Lotus WordPro to open the document.

   Table 5 lists the MIME content types for several PC applications:

Table 5. MIME content types for several PC applications

Application	MIME Content Types
Lotus Applications	WK1=application/lotus-1-2-3 WK3=application/lotus-1-2-3 WK4=application/lotus-1-2-3 123=application/lotus-1-2-3 APR-application/lotus-approach VEW=application/lotus-approach LWP=application/lotus-wordpro SAM=application/lotus-wordpro MWP=application/lotus-wordpro SMM=application/lotus-wordpro PRE=application/lotus-freelance PRZ=application/lotus-freelance

Table 5. MIME content types for several PC applications (continued)

Application	MIME Content Types
Microsoft Applications	DOC=application/ms-word XLS=application/ms-excel PPS=application/ms-powerpoint PPT=application/ms-powerpoint MPD=application/ms-project MPP=application/ms-project MPT=application/ms-project MPD=application/ms-project
HTML Applications	HTML=application/html HTM=application/htm

This section has a global scope, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

AFP

The MIME content type for AFP documents, when AFPVIEWING=NATIVE is specified in the [DEFAULT BROWSER] section. See "AFPVIEWING" on page 111 for more information. This specifies the MIME type for the document that the browser then uses to determine what program should be used process the document.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional.

Example:

```
[MIMETYPES]
AFP=application/afp
```

BMP

The MIME content type for BMP documents. By default, BMP documents are displayed using the Image Web Viewer.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional. However, if you do not specify this parameter, then ODWEK sets the MIME content type to `image/bmp` and starts the program that is associated with the BMP file type on the PC.

Example:

```
[MIMETYPES]
BMP=image/IBM-OnDemand
```

GIF

The MIME content type for GIF documents. By default, GIF documents are displayed using the Image Web Viewer.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional. However, if you do not specify this parameter, then ODWEK sets the MIME content type to `image/gif` and uses the browser's built-in viewer to display GIF documents.

Example:

```
[MIMETYPES]
GIF=image/IBM-OnDemand
```

EMAIL

The MIME content type for EMAIL documents. See "EMAILVIEWING" on page 113 for more information about processing EMAIL documents before sending them to the viewer.

Notes:

1. If you convert EMAIL documents to HTML, ODWEK sets the MIME content type to `text/html`. ODWEK ignores the value of the EMAIL parameter, if specified.
2. If you extract and uncompress EMAIL documents from Content Manager OnDemand, ODWEK uses the value of the EMAIL parameter to determine the program to open the document.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional.

Example:

```
[MIMETYPES]
EMAIL=text/plain
```

JFIF

The MIME content type for JFIF (JPEG) documents. By default, JFIF documents are displayed using the Image Web Viewer.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional. However, if you do not specify this parameter, then ODWEK sets the MIME content type to `image/jpeg` and starts the program that is associated with the JPEG file type on the PC.

Example:

```
[MIMETYPES]
JFIF=image/IBM-OnDemand
```

LINE

The MIME content type for line data documents. See "LINEVIEWING" on page 114 for more information about processing line data documents before sending them to the viewer.

This is used when LINEVIEWING=NATIVE is specified in the [DEFAULT BROWSER] section. If you extract and uncompress line data documents from Content Manager OnDemand, ODWEK uses the value of the LINE parameter to determine the program to start to open the document.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional.

Example:

```
[MIMETYPES]
LINE=text/html
```

PCX

The MIME content type for PCX documents. By default, PCX documents are displayed using the Image Web Viewer.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional. However, if you do not specify this parameter, then ODWEK sets the MIME content type to `image/pcx` and starts the program that is associated with the PCX file type on the PC.

Example:

```
[MIMTYPES]
PCX=image/IBM-OnDemand
```

PDF

The MIME content type for PDF documents.

Notes:

1. ODWEK uses the value of the PDF parameter to determine the program to start to open PDF documents. By default, PDF documents are opened with the Adobe Acrobat viewer.
2. To view PDF documents, you should obtain and install the Adobe Acrobat viewer for the browsers that are used by your organization.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional.

Example:

```
[MIMETYPES]
PDF=application/pdf
```

TIFF

The MIME content type for TIFF documents. By default, TIFF documents are displayed using the Image Web Viewer.

This parameter has a global scope, and you specify it only once in the MIMETYPES section.

This parameter is optional. However, if you do not specify this parameter, then ODWEK sets the MIME content type to `image/tiff` and starts the program that is associated with the TIFF file type on the PC.

Example:

```
[MIMETYPES]
TIFF=image/IBM-OnDemand
```

[ATTACHMENT IMAGES]

The ATTACHMENT IMAGES section identifies the image files that ODWEK uses to display attachments to a document. Each image file should contain an icon that represents a specific type of attachment. For example, you can identify an image file that contains an icon for a text attachment, a bitmap attachment, and so forth.

Notes:

1. Each parameter that you specify must identify the file type that the operating system associates with the type of attachment. The file type determines the program that the operating system starts to process the attachment. For example, if the operating system associates the file type TXT with text file attachments, add a `TXT=`*value* parameter to the ATTACHMENT IMAGES section. As the *value*, specify the name of the file that contains the icon that you want to use to indicate a text attachment to a document. When the user clicks on the icon, the operating system starts the program that is registered to open TXT documents.
2. By default, all attachments to a document are indicated by the `odic_att.gif` file (which is located in the directory that is specified by the IMAGEDIR parameter in the CONFIGURATION section). Content Manager OnDemand also uses the `odic_att.gif` file for any file types for which a parameter is not specified in the ATTACHMENT IMAGES section.

This section has a global scope, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

BMP

The parameter identifies the bitmap data type. The value identifies the file that contains the icon to represent a bitmap image attached to the document.

This parameter has a global scope, and you specify it only once in the ATTACHMENT IMAGES section.

This parameter is optional.

Example:

```
[ATTACHMENT IMAGES]
BMP=userBitMap.gif
```

GIF

The parameter identifies the GIF data type. The value identifies the file that contains the icon to represent a GIF image attached to the document.

This parameter has a global scope, and you specify it only once in the ATTACHMENT IMAGES section.

This parameter is optional.

Example:

```
[ATTACHMENT IMAGES]
GIF=userGIF.gif
```

TXT

The parameter identifies the TXT data type. The value identifies the file that contains the icon to represent a text file attached to the document.

This parameter has a global scope, and you specify it only once in the ATTACHMENT IMAGES section.

This parameter is optional.

Example:

```
[ATTACHMENT IMAGES]
TXT=userText.gif
```

[NO HTML]

The NO HTML section contains the parameters that are used to override the default characters that delimit strings and separate a list of values in the delimited ASCII output. A function generates delimited ASCII output when you set its `_nohtml` parameter to 1 (one). See Appendix C, "No HTML output," on page 155 for details about the delimited ASCII output.

This section has a global scope, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

BEGIN

The character that ODWEK uses to delimit the beginning of a string or a string of values. You must change the BEGIN delimiter if a string contains the default character (the [character).

This parameter has a global scope, and you specify it only once in the NO HTML section.

This parameter is optional.

Example:

```
[NO HTML]
BEGIN=<
```

END

The character that ODWEK uses to delimit the end of a string or a string of values. You must change the END delimiter if a string contains the default character (the] character).

This parameter has a global scope, and you specify it only once in the NO HTML section.

This parameter is optional.

Example:

```
[NO HTML]
END=>
```

SEPARATOR

The character that ODWEK uses to separate a string of values. You must change the SEPARATOR delimiter if a string contains the default character (the ^character).

This parameter has a global scope, and you specify it only once in the NO HTML section.

This parameter is optional.

Example:

```
[NO HTML]
SEPARATOR=;
```

[DEFAULT BROWSER]

You can use the DEFAULT BROWSER section to specify parameters for the browsers used by your organization. The parameters that you specify will be used unless you specify them in a browser section. (The parameters specified in a browser section override those from the DEFAULT BROWSER section. See "[browser]" on page 117.)

This section has a global scope for all browsers, and you specify it only once in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

ADDEXTENSION

Determines whether the three-character file extension of the document is added to the extra path information of the URL that is returned to the browser. Adding the file extension to the URL can help browsers determine the correct viewer to start for the document. The default value is `0` (zero) and means that the file extension is not added to the URL.

Tip: If you use the Microsoft Internet Explorer browser, IBM recommends that you specify ADDEXTENSION=1 so that the file extension is added to the URL.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
ADDEXTENSION=1
```

ADDFIELDSTODOCID

Determines whether the field values are added to the document identifiers. (The document identifiers are returned by the Document Hit List function.) The default value is `0` (zero) and means that the field values are not added to the document

identifiers. If you enable ODWEK to add the field values to the document identifiers, then they will also appear in the system log, provided that you have configured the system to save application group messages in the system log.

Notes:

1. If you use the Update Document function, then you must specify ADDFIELDSTODOCID=1.
2. If the `Annotation Flags in the document database table field` is set to `Yes`, then you must specify ADDFIELDSTODOCID=1 or else the addnote will fail. You can set the `Annotations Flags in document database table field` on the Database Information dialog box, from the General page in the Content Manager OnDemand application group definitions. (Click Advanced to open the Database Information dialog box.)

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
ADDFIELDSTODOCID=1
```

ADDNOTES

Determines whether annotations can be added to documents. If enabled, ODWEK puts a control for adding annotations next to each document in the document list. The default value is `0` (zero) and means that annotations cannot be added to documents.

Notes:

1. Users are permitted or denied the ability to add annotations to documents based on the Annotation permissions in the Content Manager OnDemand application group.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
ADDNOTES=1
```

AFPVIEWING

When a user retrieves an AFP document from the Content Manager OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the viewer. For example, some customers convert AFP documents to HTML with the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms and use the AFP2HTML applet to view the HTML output. Those customers should specify AFPVIEWING=HTML so that ODWEK will convert the AFP document before sending it to the viewer.

You can set the parameter to one of the following values:

ASCII ODWEK converts AFP documents to ASCII text.

HTML ODWEK converts AFP documents to HTML documents with the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms.

NATIVE ODWEK extracts and uncompresses AFP documents and their resources from Content Manager OnDemand.

Note: If you specify AFPVIEWING=NATIVE, verify that the MIME content type for AFP documents identifies the viewer that you want to use. See "[MIMETYPES]" on page 103 for details.

PDF ODWEK converts AFP documents to PDF documents with the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms.

Important: If you specify AFPVIEWING=PDF, verify that the MIME content type for PDF documents identifies the viewer that you want to use. See "[MIMETYPES]" on page 103 for details.

PLUGIN ODWEK does not convert AFP documents (the default).

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section. When using the Retrieve Document function, you can override the specified action with the `_afp` parameter.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
AFPVIEWING=PLUGIN
```

AUTODOCRETRIEVAL

Specifies whether the viewer automatically displays a document when one and only one document matches the query. This capability means that, for queries that you know will match only one document, you can set up the system to bypass the document list Web page and display the document without the user taking action. The default value is `0` (zero) and means that ODWEK will display the document list Web page, even if only one document matches the query.

Important: If you are using Internet Explorer, you must specify `AUTODOCRETRIEVAL=0` to disable the automatic document retrieval function. Beginning with Version 7.1.0.11, IBM has set `AUTODOCRETRIEVAL=0` as the default value in the [IE] section of the ARSWWW.INI file that is distributed with ODWEK.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
AUTODOCRETRIEVAL=1
```

EMAILVIEWING

When a user retrieves an EMAIL document from the Content Manager OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the viewer.

You can set this parameter to one of the following values:

NATIVE ODWEK extracts and uncompresses EMAIL documents from Content Manager OnDemand.

Note: If you specify EMAIL=NATIVE, verify that the MIME content type identifies the viewer that you want to use. See "[MIMETYPES]" on page 103 for details.

HTML ODWEK converts EMAIL documents to HTML documents.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section. When using the Retrieve Document function, you can override the specified action with the `_email` parameter.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
EMAILVIEWING=HTML
```

ENCRYPTCOOKIES

Determines whether ODWEK encrypts cookies that are sent to the browser. The default value is `0` (zero), meaning that cookies will not be encrypted. Specify `1` (one) to encrypt all cookies that are sent to the browser.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
ENCRYPTCOOKIES=1
```

ENCRYPTURL

Determines whether ODWEK encrypts the `server`, `userid`, `password`, and `docid` values that are contained in the URL that is sent to the browser. The default value is `0` (zero), meaning that these values will not be encrypted. Specify `1` (one) to encrypt these values.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional. However, if you must use the GET method to transfer form parameters and values between the browser and the Web server, then you

can encrypt these values by specifying ENCRYPTURL=1. See “Server and data security” on page 9 for more information about the method attribute of the form tag.

Example:

```
[DEFAULT BROWSER]
ENCRYPTURL=1
```

FOLDERDESC

Specifies whether the folder description is displayed to the right of the folder name on the folder selection page. The default value is `0` (zero), meaning that the folder description will not be displayed. Specify `1` (one) to display the folder description. If this parameter is not specified or if the value is not defined or recognized, the folder description will not be displayed.

This parameter has a global scope, unless overridden in a browser section (see “[browser]” on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
FOLDERDESC=1
```

LINEVIEWING

When a user retrieves a line data document from the Content Manager OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the viewer.

You can set this parameter to one of the following values:

APPLET ODWEK converts line data documents for viewing with the Line Data applet (the default).

ASCII ODWEK converts line data documents to ASCII text.

NATIVE ODWEK extracts and uncompresses line data documents from Content Manager OnDemand.

Note: If you specify LINEVIEWING=NATIVE, verify that the MIME content type identifies the viewer that you want to use. See “[MIMETYPES]” on page 103 for details.

This parameter has a global scope, unless overridden in a browser section (see “[browser]” on page 117). You specify this parameter only once in the DEFAULT BROWSER section. When using the Retrieve Document function, you can override the specified action with the `_line` parameter.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
LINEVIEWING=APPLET
```

MAXHITS

The maximum number of items returned to the document list, regardless of the number of items that match the query.

Notes:

1. The document list is filled with items that match a query in the order in which the items were loaded into the database.
2. ODWEK uses one of the following values to determine the number of items to return to the document list:
   - For the Document Hit List function, the value of the Maximum Hits field (specified on the folder Permissions page). This value overrides all other values.
   - For the Document Hit List and Print Document functions, the value of the `_max_hits` parameter, if specified for a function. The value of the `_max_hits` parameter overrides the MAXHITS parameter.
   - The value of the MAXHITS parameter, if specified.
   - If none of the above are specified, ODWEK returns a maximum of 200 items to the document list.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
MAXHITS=200
```

NOLINKS

Determines whether the document list contains controls for viewing documents. If enabled, ODWEK adds a control next to each document. To view a document, the user must use the control. The default value is `0` (zero) and means that the user must use a text link to view a document.

Important: You must set NOLINKS=0 if you are using the Microsoft Internet Explorer browser. IBM recommends that you specify NOLINKS=0 in any browser sections that you define for Internet Explorer.

This parameter has a global scope, unless overridden in a browser section (see "[browser]" on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
NOLINKS=1
```

ODApplet.jre.path.IE

See "Java line data viewer" on page 81.

ODApplet.jre.path.NN

See "Java line data viewer" on page 81.

ODApplet.jre.version

See "Java line data viewer" on page 81.

ODApplet.version

See “Java line data viewer” on page 81.

SERVERPRINT

Determines whether the document list contains controls for sending documents to a server printer. If enabled, ODWEK adds a control next to each document. The default value is `0` (zero) and means that users must open a document before they can send it to a server printer.

Notes:

1. To use server print, at least one server printer must be defined to the Content Manager OnDemand server.
2. Users are permitted or denied the ability to print documents based on the Print permissions in the Content Manager OnDemand application group.

This parameter has a global scope, unless overridden in a browser section (see “[browser]” on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
SERVERPRINT=1
```

SERVERPRINTERS

Use to specify the type of server print devices that the user can select. There are three types of server print devices:

P Server Printer

I Server Printer with Information

F Server Fax

You can specify from zero to three types, in a comma-separated list.

The following example shows how to specify that the user can select server printer and server fax devices:

```
[DEFAULT BROWSER]
SERVERPRINTERS=P,F
```

SHOWDOCLOCATION

When generating delimited ASCII output rather than HTML (see Appendix C, “No HTML output,” on page 155), determines whether the storage location of the document will appear in the output. See “Document Hit List” on page 157 for details. The default value is `0` (zero) and means that the storage location will not appear in the output.

Notes:

1. To display the storage location, you must also set the `Display Document Location` property in the Content Manager OnDemand folder.

This parameter has a global scope, unless overridden in a browser section (see “[browser]” on page 117). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
SHOWDOCLOCATION=1
```

VIEWNOTES

Determines whether annotations to documents can be viewed. If enabled, ODWEK puts a control for viewing annotations next to each document in the document list. The default value is `0` (zero) and means that annotations cannot be viewed.

Notes:

1. Users are permitted or denied the ability to view annotations to documents based on the Annotation permissions in the Content Manager OnDemand application group.

This parameter has a global scope, unless overridden in a browser section (see "[browser]"). You specify this parameter only once in the DEFAULT BROWSER section.

This parameter is optional.

Example:

```
[DEFAULT BROWSER]
VIEWNOTES=1
```

[browser]

You can specify options for the specific browsers used by your organization. Except for the parameters noted in Item 1 below, the parameters that you specify in a browser section override the parameters from the DEFAULT BROWSER section of the ARSWWW.INI file. (The parameters that you specify in the DEFAULT BROWSER section will be used unless you specify them in a browser section.)

Notes:

1. The following parameters have a global scope and may only be specified in the DEFAULT BROWSER section. (If these parameters are specified in some other browser section, they will be ignored.)
   - ODApplet.jre.path.IE
   - ODApplet.jre.path.NN
   - ODApplet.jre.version
   - ODApplet.version
2. The section header must contain a string that identifies the browser for which you want to specify the options. ODWEK extracts the value of the HTTP_USER_AGENT environment variable to determine the browser being used. ODWEK then searches the ARSWWW.INI file for a browser section that matches the value.

A browser section has a global scope for the specified browser. Specify only one browser section for each browser. You should specify only the parameters that you need to override from the DEFAULT BROWSER section.

This section is optional.

This section can contain the same parameters that are defined for the default browser. See "[DEFAULT BROWSER]" on page 110.

Examples:

```
[IE 5]
AUTODOCRETRIEVAL=0
NOLINKS=0

[Firefox/3.5.1]
AUTODOCRETRIEVAL=1
NOLINKS=1
```

[DEBUG]

The DEBUG section contains options that you can use to help solve problems that you and others in your organization are having using ODWEK.

The DEBUG section has a global scope, and you specify it only once in the ARSWWW.INI file. **Important:** If specified, the DEBUG section must be the first executable statement in the ARSWWW.INI file.

This section is optional.

This section can contain the following parameters:

TRACE

Enables ODWEK to write messages and other program information to a log file. The log file is named arswww.trace.

This parameter has a global scope, and you specify it only once in the DEBUG section.

This parameter is optional. By default, ODWEK does not write messages to a log file.

Example:

```
[DEBUG]
Trace=4
```

TRACEDIR

Determines the directory in which ODWEK writes the `arswww.trace` file, if logging is enabled by using the TRACE parameter.

This parameter has a global scope, and you specify it only once in the DEBUG section.

This parameter is optional. By default, if logging is enabled, ODWEK writes the log file to the `/tmp` directory.

Example:

```
[DEBUG]
TraceDir=/tmp
```

Notes:

1. See Chapter 3, "Installation requirements," on page 13 for information about the log file directory.
2. Verify the permissions of the directory that you specify. The processes that run ODWEK programs must write to and read from the log directory.
3. For performance reasons, the directory and mount point that you specify for the TRACEDIR parameter should be different from those that you specify for

the CACHEDIR and TEMPDIR parameters. Also, it is recommended that you use a directory that is local to the system, and not an NFS mount.

4. When Trace is enabled, ODWEK creates a new `arswww.trace` file at application startup and appends trace messages to the `arswww.trace` file until the application is stopped. Upon restart, a new `arswww.trace` file is created, and the previous file is renamed to `arswww.trace.`. There are four trace levels that can be specified:

   1 ERROR tracing. Only errors are reported.

   2 WARNING and ERROR messages are reported.

   3 INFO messages are captured along with the ERROR and WARNING.

   4 FULL TRACE. All messages are captured, including function trace.

   It is recommended that you set `Trace=1` for normal operating environments. This way basic information is available for general error capture and diagnosis with minimal system performance overhead.

Example ARSWWW.INI file

An example ARSWWW.INI configuration file is provided with the product. The example configuration file sets the most commonly used default values for servers, browsers, and viewers.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Server Configuration      ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;[@SRV@_<host alias>]
;HOST=<host name>
;PORT=
;PROTOCOL=

[debug]
trace=4
tracedir=c:\temp\8400


[Configuration]
TemplateDir=c:\program files\ibm\ondemand web enablement kit\samples
ImageDir=/images
AppletDir=/applets
TempDir=C:\temp
CacheDir=c:\temp\cache
CacheSize=512
CacheMinThreshold=40
CacheMaxThreshold=80

[afp2html]
InstallDir=c:\opt\afp2web
ConfigFile=c:\opt\afp2web\afp2html.ini

[afp2pdf]
InstallDir=c:\opt\afp2pdf
ConfigFile=c:\opt\afp2pdf\afp2pdf.ini
```

(Continued on next page.)

(Example ARSWWW.INI file, continued.)

```
[xenos]
InstallDir=c:\Program Files\IBM HTTP Server 2.0\cgi-bin
ConfigFile=c:\javatest\backup\arsxenos.ini

[mimetypes]
BMP=image/IBM_OnDemand
GIF=image/IBM_OnDemand
;JFIF=image/IBM_OnDemand
JPEG=image/IBM_OnDemand
PCX=image/IBM_OnDemand
TIFF=image/IBM_OnDemand
TIFF=image/TIF
PNG=image/IBM_OnDemand
PDF=application/pdf
AFP=application/afp
LINE=application/line
EMAIL=text/html
META=application/unknown
XFDL=application/unknown
HTML=application/html

[attachment images]
TXT=userText.gif
BMP=userBitMap.gif
GIF=userGIF.gif

[no html]
Begin=[
End=]
Separator=^

;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Default Browser    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;

[default browser]
;AfpViewing=[ascii,html,native,pdf,plugin,xenos]
AfpViewing=plugin
;LineViewing=[ascii,applet,native]
LineViewing=html
;EmailViewing=[html,native]
EmailViewing=html
;MetaViewing=[xenos,native]
MetaViewing=native
NoLinks=1
ViewNotes=0
AddNotes=0
ServerPrint=0
ServerPrinters=P
AutoDocRetrieval=0
MaxHits=200

[IE]
NoLinks=0
AddExtension=0
```

Your next step

After you have verified the ARSWWW.INI file, go to Chapter 7, “Configuring the sample applications,” on page 69

CGI API reference

This section contains information about the programming functions that are available with ODWEK. This chapter is of primary interest to programmers responsible for integrating ODWEK with Web browsers.

Tip: Parameter values are standard text. Is it possible that the text could consist of characters that will confuse browsers. To prevent possible errors, you must code all special characters in their corresponding hexadecimal codes. These special characters include control characters and certain alphanumeric symbols. For example, the string:

```
The post date is 12/31/95
```

would be converted to:

```
The%20post%20date%20is%2012%2f31%2f95
```

Parameter values include folder names, folder field names, and search criteria.

Add Annotation

Add an annotation to the specified document

Purpose

The Add Annotation function enables users to add an annotation to the specified document. To add an annotation, the user must be given permission to add annotations in the OnDemand application group.

Parameters

Table 6. Add Annotation function

Name=Value	Purpose
_function=addnote	Add an annotation.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid. The user must be given permission to add annotations to the OnDemand application group(s).
_password=*value*	The password for the user.
_folder=*value*	The name of the folder.
_perm=*value*	Determines whether the annotation is Public (`0`), Private (`1`), or Private for Group (`2`). Public annotations can be viewed by any user who has been given the View permission under Annotation in the application group. Private annotations can be viewed by the user that created the annotation, application group administrators, and system administrators. Private for Group annotations can be viewed by users in the specified group, application group administrators, and system administrators. The **_group** parameter contains the name of the group. The default value is `0` (Public).
_group=*groupName*	If the **_perm** parameter is set to `2` (Private for Group), names the group.
_copy=*value*	Determines whether the annotation should remain attached to the document if the document is exported to another server. The default value is `off`, meaning the annotation is not attached to the document. A value of `on` means the annotation is attached to the document if the document is exported to another server.
_text=*value*	The text of the annotation.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, ODWEK uses the ADDNOTE.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - –> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the add an annotation function.

Table 6. Add Annotation function (continued)

Name=Value	Purpose
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, "No HTML output," on page 155 for details about the delimited ASCII output.
_docid=*documentID*	The identifier of the document to which the annotation is to be attached. The document identifier is returned by the Document Hit List function.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.
_logoff=1	Automatically disconnects the user from the OnDemand server after adding the annotation. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is 1 (one).

Usage

The following parameters are required:

_function
_server
_user
_password
_text
_docid

The following parameters are optional:

_perm
_group (required if _perm specifies Private for Group)
_html
_nohtml
_port
_codepage
_logoff

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=addnote
&_server=myzos&_user=web&_password=web
&_folder=credit%20card%20statements
&_text=Test%20note%20from%20the%20OnDemand%20Internet%20Client
&_docid=6850-6851-SUA17-1FAAA-225712-1634-132014-132172-89-76-11-25-0
&_perm=1&_logoff=1
```

Change Password

Change the OnDemand logon password

Purpose

The Change Password function permits users to change their OnDemand passwords.

Parameters

Table 7. Change Password function

Name=Value	Purpose
_function=chgpassword	Change the OnDemand password for the userid.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid.
_password=*value*	The password for the userid.
_new_password=*value*	The new password for the userid.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the CHGPASSWORD.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - -> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the change password function.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.

Table 7. Change Password function (continued)

Name=Value	Purpose
_cgibin=*program*	Used by the CGI program when generating the next output page. If specified, then the page will contain a call to the specified program instead of the default program (ARSWWW.CGI). This parameter is primarily used by programmers who are creating a front-end CGI program or servlet to the CGI program or servlet provided by IBM. The *program* can name a directory that is relative to the `ServerRoot` directive or name an *alias* that is defined in the Web server configuration file. By default, ODWEK retrieves the CGI program from the CGI-BIN directory.
_logoff=1	Automatically disconnects the user from the OnDemand server after changing the password. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is 1 (one).

Usage

The following parameters are required:

_function
_server
_user
_password
_new_password

The following parameters are optional:

_html
_nohtml
_port
_codepage
_logoff
_cgibin

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=chgpassword
&_server=myzos&_user=web&_password=web
&_new_password=newpw&_html=template.htm&_logoff=1
```

Document Hit List

Display the list of documents that match the search criteria

Purpose

The Document Hit List function displays the list of documents that match the search criteria for a specific folder. Each document is represented by a link to the document on the OnDemand server. After clicking on a link, ODWEK retrieves the document from the server and displays it in the browser window using the appropriate viewer.

Parameters

Table 8. Document Hit List function

Name=Value	Purpose
_function=dochitlist	Display list of documents that match the search criteria.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid.
_password=*value*	The password for the userid.
_folder=*value*	The name of the folder.
*folder field name***=***value*	The name of a folder search field and the search value. You can specify one or more sets of field names and search values, up to the number of fields defined for the folder.
*folder field name***2=***value*	For folder search fields that use the BETWEEN or NOT BETWEEN search operators, the upper value with which to search the field.
*folder field name***OP=***value*	The operator to be used to override the default operator for a folder search field. The *value* must be one of the following: **1** to indicate Equal **2** to indicate Not Equal **4** to indicate Less Than **8** to indicate Less Than or Equal **16** to indicate Greater Than **32** to indicate Greater Than or Equal **64** to indicate In **128** to indicate Not In **256** to indicate Like **512** to indicate Not Like **1024** to indicate Between **2048** to indicate Not Between
_display_fields=*value[,value,...]*	A comma separated list that contains the names of the folder display fields. You can specify one or more field names. If you do not specify this parameter, the output page contains all of the folder display fields.
_sort_field=*value[,value,...]*	Determines the folder search field or fields that OnDemand uses to sort items in the document list. If you specify more than one field, separate the field names with a comma. For example: `_sort_field=Account,Account+Balance,Date` . The default sort fields are defined on the Field Information page for the folder.
_sort_order=*value[,value,...]*	For each folder search field that is specified in the sort_field parameter, determines whether OnDemand sorts items first to last or last to first. Specify an `A` (ascending) to sort items first to last. Specify any other character to sort items last to first (descending). `For example: _sort_order=A,D,A` . The default sort order is determined by the sort order that is defined on the Field Information page for the folder.

Table 8. Document Hit List function (continued)

Name=Value	Purpose
_max_hits=*value*	Determines the maximum number of items that ODWEK returns to the document list, regardless of the number of items that match the query. ODWEK fills the document list with items that match a query in the order in which matching items were loaded into the database. ODWEK uses one of the following values to determine the number of items to return to the document list: • The value of the Maximum Hits field (specified on the Permissions page in the OnDemand folder). This value overrides all other values. • The value of the **_max_hits** parameter, if specified. This value overrides the `MAXHITS` parameter from the ARSWWW.INI file. • The value of the `MAXHITS` parameter, if specified. • If none of the above are specified, ODWEK returns a maximum of 200 items to the document list.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the DOCHITLIST.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - -> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the document hit list function.
_frame=*value*	The output of this command will include a `target=value` attribute. This parameter makes building HTML frames simpler. This is an optional parameter.
_datefmt=*value*	Determines the format of date values used by ODWEK to search the database and display items that match a query. The default date format is set on the folder Field Information page. See *IBM Content Manager OnDemand for Multiplatforms: Administration Guide* for details about date formats that are supported by OnDemand.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.

Table 8. Document Hit List function (continued)

Name=Value	Purpose
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.
_sql=*string*	Specifies the SQL query that OnDemand uses to search the folder. If you specify this parameter, the SQL query is used to search the folder rather than any folder field name / value pairs that may be specified. OnDemand does not validate the query string. When using an SQL string, you must specify application group database field names and values. If you plan to query date fields, you must specify OnDemand internal date values. For example, the date January 1, 1999 would be specified as 10593. You can use the ARSDATE program to list the internal date value for a given date. The SQL string is used to search all of the application groups contained in the folder. If the SQL string contains a database field name that is in one application group but not in another application group, then the query will fail.
_date1=*value*	Use to specify the beginning date in a range of dates to search. If you specify the **_date1** and **_date2** parameters, OnDemand limits the query to the table or tables that contain one or both of the specified dates. The format of the date string that you specify must match the display format of the folder field. (You can use the administrative client to list the display format of the folder field.)
_date2=*value*	Use to specify the ending date in a range of dates to search. If you specify the **_date1** and **_date2** parameters, OnDemand limits the query to the table or tables that contain one or both of the specified dates. The format of the date string that you specify must match the display format of the folder field. (You can use the administrative client to list the display format of the folder field.)
_cgibin=*program*	Used by the CGI program when generating the next output page. If specified, then the page will contain a call to the specified program instead of the default program (ARSWWW.CGI). This parameter is primarily used by programmers who are creating a front-end CGI program or servlet to the CGI program or servlet provided by IBM. The *program* can name a directory that is relative to the `ServerRoot` directive or name an *alias* that is defined in the Web server configuration file. By default, ODWEK retrieves the CGI program from the CGI-BIN directory.
_or=*value*	Specify a `1` (one) to connect search fields using the OR logical operator; an item must match at least one of the specified search values. The default value is `0` (zero), which means that OnDemand connects search fields with the AND logical operator (an item must match all of the specified search values).
_logoff=1	Automatically disconnects the user from the OnDemand server after creating the document list. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is 1 (one).

Usage

The following parameters are required:

_function

_server
_user
_password
_folder

The following parameters are optional:

folder field name
*folder field name*2
*folder field name*OP
_display_fields
_sort_field
_sort_order
_max_hits
_frame
_datefmt
_sql
_date1
_date2
_or
_html
_nohtml
_port
_codepage
_logoff
_cgibin

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=dochitlist
&_server=myzos&_user=web&_password=web
&_folder=credit%20card%20statements
&account%20number=1000100010009999&date=1%2f1%2f96&date2=12%2f31%2f96
&nameOP=256&name=%AA
&_sort_field=Account,Account%20Balance,Date&_sort_order=A,D,A
&_logoff=1
&_html=template.htm
```

Logoff

Logoff an OnDemand server

Purpose

The Logoff function attempts to log a user off an OnDemand server. The name of the server and the userid to log off are stored in a browser cookie on the client by the Logon function. If the server is not a valid OnDemand server, an error message is returned. If the userid is not logged on to the specified server, an error message is returned.

Parameters

Table 9. Logoff function

Name=Value	Purpose
_function=logoff	Log off an OnDemand server.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the LOGOFF.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– —> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the logoff function.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.

Usage

The following parameters are required:

_function

The following parameters are optional:

_html

_nohtml
_port

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=logoff
&_html=template.htm
```

Logon

Logon to an OnDemand server

Purpose

The Logon function attempts to access an OnDemand server using the values of the server, user, and password parameters. The Logon function verifies that the specified user is authorized to logon to the specified server and verifies the password. If the user is not authorized to logon to the server, an error message is returned. If the server is not a valid OnDemand server, an error message is returned. If the password is not valid for the user, an error message is returned. After a successful logon, the Logon function displays a Web page that contains a list of the folders that the user is authorized to access.

Parameters

Table 10. Logon function

Name=Value	Purpose
_function=logon	Logon to an OnDemand server.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid.
_password=*value*	The password for the userid.
_new_password=*value*	The new password for the userid. Allows the password to be changed after successfully logging on to the OnDemand server. This is an optional parameter.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the LOGON.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - -> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the logon function.
_frame=*value*	The output of this command will include a `target=value` attribute. This parameter makes building HTML frames simpler. This is an optional parameter.
_datefmt=*value*	Determines the format of date values used by ODWEK to search the database and display items that match a query. The default date format is set on the folder Field Information page. See *IBM Content Manager OnDemand for Multiplatforms: Administration Guide* for details about date formats supported by OnDemand.

Table 10. Logon function (continued)

Name=Value	Purpose
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, "No HTML output," on page 155 for details about the delimited ASCII output.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.
_cgibin=*program*	Used by the CGI program when generating the next output page. If specified, then the page will contain a call to the specified program instead of the default program (ARSWWW.CGI). This parameter is primarily used by programmers who are creating a front-end CGI program or servlet to the CGI program or servlet provided by IBM. The *program* can name a directory that is relative to the `ServerRoot` directive or name an *alias* that is defined in the Web server configuration file. By default, ODWEK retrieves the CGI program from the CGI-BIN directory.

Usage

The following parameters are required:

_function
_server
_user
_password

The following parameters are optional:

_new_password
_frame
_datefmt
_html
_nohtml
_port
_codepage
_logoff
_cgibin

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=logon
&_server=myzos&_user=web&_password=web
&_html=template.htm
```

Print Document (Server)

Sends one or more documents to the specified server printer

Purpose

The Print Document function sends copies of documents to an OnDemand server printer. To use the server print facility, the user must have the Print permission under Document in the OnDemand application group. At least one server printer must be defined on the specified OnDemand server.

Parameters

Table 11. Print Document function

Name=Value	Purpose
_function=printDocuments	Print documents.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid. The user must have the Print permission under Document in the application group.
_password=*value*	The password for the user.
_folder=*value*	The name of the folder.
_printer=*value*	The name of the OnDemand server printer. When the specified printer is a FAX or a Printer with Information, then you can specify the following additional parameters: **_recv_name=***value* The receiver's name. **_recv_comp=***value* The receiver's company name. **_recv_fax=***value* The receiver's fax number. **_send_name=***value* The sender's name. **_send_comp=***value* The sender's company name. **_send_tel=***value* The sender's telephone number. **_send_fax=***value* The sender's fax number. **_send_cover=***value* A user-defined overlay that the Header Page Exit program merges with the values of the other parameters to produce a cover page for the document. **_subject=***value* A string that represents the subject of the document. **_notes=***value* A string that represents a note about the document.

Table 11. Print Document function (continued)

Name=Value	Purpose
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the PRINTDOCS.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - -> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the print documents function.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.
_docids=*documentIDList*	A list of document identifiers for the documents to be printed. The document identifiers are returned by the Document Hit List function. If you specify more than one document identifier, then you must separate the document identifiers with the `\003` character. **Important:** If the number of document identifiers exceeds 200, then you must specify the **_max_hits** parameter.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.

Table 11. Print Document function (continued)

Name=Value	Purpose
_max_hits=*value*	Use this parameter to specify the number document identifiers to process. Specify a value that is equal to or greater than the number of document identifiers specified with the **_docids** parameter. **Important:** If the number of document identifiers exceeds the value specified by the MAXHITS parameter in the ARSWWW.CGI file (or 200, if not specified), then you must specify the **_max_hits** parameter. If you do not specify the **_max_hits** parameter (or you do not specify a value for the MAXHITS parameter), a maximum of 200 document identifiers will be processed, regardless of the number of document identifiers that you specified with the **_docids** parameter. ODWEK uses one of the following values to determine the number of document identifiers to process: • The value of the **_max_hits** parameter, if specified. This value overrides the value of the MAXHITS parameter. • The value of the MAXHITS parameter, if specified. • If none of the above are specified, ODWEK processes a maximum of 200 document identifiers.
_logoff=1	Automatically disconnects the user from the OnDemand server after printing the document. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is 1 (one).

Usage

The following parameters are required:

_function
_server
_user
_password
_folder
_printer
_docids

The following parameters are optional:

_recv_name
_recv_comp
_recv_fax
_send_name
_send_comp
_send_tel
_send_fax
_send_cover
_subject
_notes
_max_hits
_html
_nohtml
_port
_codepage
_logoff

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=printDocuments
&_server=myzos&_user=web&_password=web
&_folder=credit%20card%20statements
&_printer=infoprint60
&_docids=6850-6851-SUA17-1FAAA-225712-1634-132014-132172-89-76-11-25-0
&_logoff=1
```

Retrieve Document

Retrieves the selected document from OnDemand

Purpose

The Retrieve Document function retrieves the selected document from the OnDemand server. ODWEK displays the document in the browser window using the applet, plug-in, or other program that is associated with the document type.

Parameters

Table 12. Retrieve Document function

Name=Value	Purpose
_function=retrieve	Retrieve the selected document.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid.
_password=*value*	The password for the userid.
_folder=*value*	The name of the folder.
folder field name=*value*	The name of a folder search field and the search value. You can specify one or more sets of field names and search values, up to the number of fields defined for the folder.
_html=*value*	When an error occurs retrieving a document, determines the HTML file that ODWEK uses as a template to generate the (error) output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the RETRIEVE.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - –> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the retrieve function.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.

Table 12. Retrieve Document function (continued)

Name=Value	Purpose
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.
_cgibin=*program*	Used by the CGI program when generating the next output page. If specified, then the page will contain a call to the specified program instead of the default program (ARSWWW.CGI). This parameter is primarily used by programmers who are creating a front-end CGI program or servlet to the CGI program or servlet that is provided by IBM. The *program* can name a directory that is relative to the `ServerRoot` directive or name an *alias* that is defined in the Web server configuration file. By default, ODWEK retrieves the CGI program from the CGI-BIN directory.
_or=*value*	Specify a `1` (one) to connect search fields using the OR logical operator; an item must match at least one of the specified search values. The default value is `0` (zero), which means that OnDemand connects search fields with the AND logical operator (an item must match all of the specified search values).
_afp=*value*	When you retrieve an AFP document from the OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the viewer. For example, some customers convert AFP documents to HTML with the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms and use the AFP2HTML applet to view the HTML output. Those customers should specify `_afp=HTML` so that ODWEK will convert the AFP document before sending it to the viewer. The *value* can be: **ASCII** ODWEK converts the AFP document to ASCII text. **HTML** ODWEK converts the AFP document to HTML with the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms. **NATIVE** ODWEK extracts and uncompresses the AFP document and its resources from OnDemand. **Note:** If you specify `_afp=NATIVE`, verify that the MIME content type identifies the viewer that you want to use (see "[MIMETYPES]" on page 103 for more information). **PDF** ODWEK converts the AFP document to PDF with the Content Manager OnDemand Advanced Function Presentation Transformations for Multiplatforms. **PLUGIN** ODWEK does not convert the AFP document (the default).
_email=*value*	When you retrieve an EMAIL document from the OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the viewer. The *value* can be: **NATIVE** ODWEK extracts and uncompresses the EMAIL document from OnDemand. **Note:** If you specify `_email=NATIVE`, verify that the MIME content type identifies the viewer that you want to use (see "[MIMETYPES]" on page 103 for more information). **HTML** ODWEK converts the EMAIL document to HTML.

Table 12. Retrieve Document function (continued)

Name=Value	Purpose
_line=*value*	When you retrieve a line data document from the OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the viewer. The *value* can be: **APPLET** ODWEK converts the line data document for viewing with the Line Data applet (the default). **ASCII** ODWEK converts the line data document to ASCII text. **NATIVE** ODWEK extracts and uncompresses the line data document from OnDemand. **Note:** If you specify `_line=NATIVE`, verify that the MIME content type identifies the viewer that you want to use (see "[MIMETYPES]" on page 103 for more information).
_docid=*documentID*	The identifier of the document to be retrieved. The document identifier is returned by the Document Hit List function.
_logoff=1	Automatically disconnects the user from the OnDemand server after retrieving the document. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is 1 (one).

Usage

The following parameters are required:

_function
_server
_user
_password
_folder

The following parameters are optional:

folder field name
_docid
_or
_afp
_email
_line
_html
_nohtml
_port
_codepage
_logoff
_cgibin

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=retrieve
&_server=myzos&_user=web&_password=web
&_folder=credit%20card%20statements
&account%20number=1000100010009999&date=1%2f1%2f96
&_html=template.htm&_logoff=1
```

Search Criteria

Display search criteria for a specific folder

Purpose

The Search Criteria function displays the search criteria for a specific folder using a form. The user can accept the default search criteria or enter search criteria to search for a specific document. After clicking the Submit button, ODWEK displays a Web page that lists the documents that match the search criteria.

Parameters

Table 13. Search Criteria function

Name=Value	Purpose
_function=searchcrit	Display search criteria for a specific folder.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid.
_password=*value*	The password for the userid.
_folder=*value*	The name of the folder to search.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the SEARCHCRIT.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` variable. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - -> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the search criteria function.
_frame=*value*	The output of this command will include a `target=value` attribute. This parameter makes building HTML frames simpler. This is an optional parameter.
_datefmt=*value*	Determines the format of date values used by ODWEK to search the database and display items that match a query. The default date format is set on the folder Field Information page. See *IBM Content Manager OnDemand for Multiplatforms: Administration Guide* for details about date formats supported by OnDemand.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.

Table 13. Search Criteria function (continued)

Name=Value	Purpose
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.
_cgibin=*program*	Used by the CGI program when generating the next output page. If specified, then the page will contain a call to the specified program instead of the default program (ARSWWW.CGI). This parameter is primarily used by programmers who are creating a front-end CGI program or servlet to the CGI program or servlet provided by IBM. The *program* can name a directory that is relative to the `ServerRoot` directive or name an *alias* that is defined in the Web server configuration file. By default, ODWEK retrieves the CGI program from the CGI-BIN directory.
_logoff=1	Automatically disconnects the user from the OnDemand server after displaying the search criteria. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is `1` (one).

Usage

The following parameters are required:

_function
_server
_user
_password
_folder

The following parameters are optional:

_frame
_datefmt
_html
_nohtml
_port
_codepage
_logoff
_cgibin

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=searchcrit
&_server=myzos&_user=web&_password=web
&_folder=credit%20card%20statements&_html=template.htm
&_logoff=1
```

Update Document

Updates one or more database values for the specified document

Purpose

The Update Document function allows authorized users to update documents. The Update Document function updates one or more database values for a specific document.

Parameters

Table 14. Update Document function

Name=Value	Purpose
_function=updatedoc	Update the database.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid. The user must have the Update permission under Document in the application group.
_password=*value*	The password for the user.
_folder=*value*	The name of the folder.
folder field name=*value*	The name of the field that you want to update and the value that you want put in the field. You can specify one or more sets of field names and values, up to the number of fields defined for the folder.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the UPDATE.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the `TEMPLATEDIR` parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - -> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM file is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the update function.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.
_docid=*documentID*	The identifier of the document to be updated. The document identifier is returned by the Document Hit List function.

Table 14. Update Document function (continued)

Name=Value	Purpose
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the `CODEPAGE` parameter in the ARSWWW.INI file.
_logoff=1	Automatically disconnects the user from the OnDemand server after updating the document. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is `1` (one).

Usage

The following parameters are required:

_function
_server
_user
_password
_folder

The following parameters are optional:

folder field name
_docid
_html
_nohtml
_port
_codepage
_logoff

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=updatedoc
&_server=myzos&_user=web&_password=web
&_folder=credit%20card%20statements
&account%20number=1000100010009999
&_docid=6850-6851-SUA17-1FAAA-225712-1634-132014-132172-89-76-11-25-0
&_html=template.htm&_logoff=1
```

View Annotations

View annotations attached to the specified document

Purpose

The View Annotations function enables users to view the annotations attached to the specified document. To view annotations, the user must have the View permission under Annotation in the OnDemand application group.

Parameters

Table 15. View Annotations function

Name=Value	Purpose
_function=getnotes	View annotations.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid. The user must be have the View permission under Annotation in the application group(s).
_password=*value*	The password for the user.
_folder=*value*	The name of the folder.
_html=*value*	Determines the HTML file that ODWEK uses as a template to generate the output Web page. The value can be a file name or an * (asterisk). If the value is an asterisk, then ODWEK uses the GETNOTES.HTML file found in the directory named by the `TEMPLATEDIR` parameter in the ARSWWW.INI file. If the value is a file name without a path name, then the file must be located in the directory named by the TEMPLATEDIR parameter. If the value includes a path name, then the path should be relative to the directory named by the `TEMPLATEDIR` parameter. The overall content of the HTML file is defined by the customer. However, the file must contain the following comment line: <!– - -AOI# Marker– - -> The location of the comment line determines where ODWEK places its output. All lines above the comment line will be written before the output generated by ODWEK. All lines below the comment line will be written after the output generated by ODWEK. The TEMPLATE.HTM is the sample template file provided with ODWEK. You can use the sample template file to help create your own template file for the view annotations function.
_nohtml=*value*	Determines the type of output generated by ODWEK. The default value is `0` (zero) and means that ODWEK generates HTML output. If you specify `1` (one), then ODWEK generates delimited ASCII output. See Appendix C, “No HTML output,” on page 155 for details about the delimited ASCII output.
_docid=*documentID*	The identifier of the document that contains the annotations to be viewed. The document identifier is returned by the Document Hit List function.
_port=*value*	The TCP/IP port number that ODWEK and the OnDemand library server use to communicate. The default value, `0` (zero), means that the port number that is specified for the OnDemand service in the SERVICES file is used. If the OnDemand service is not specified in the SERVICES file, then port number `1445` will be used. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file. Before using any port number, verify with your TCP/IP administrator that the port is available.

Table 15. View Annotations function (continued)

Name=Value	Purpose
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the Web server. If the code page of the server is different than the code page of the database, then you must specify the code page. Any value that you specify overrides the value of the CODEPAGE parameter in the ARSWWW.INI file.
_logoff=1	Automatically disconnects the user from the OnDemand server after viewing the annotation. Specifying this parameter eliminates the need for your application to call the Logoff function to disconnect the user. The only valid value for this parameter is 1 (one).

Usage

The following parameters are required:

_function
_server
_user
_password
_folder
_docid

The following parameters are optional:

_html
_nohtml
_port
_codepage
_logoff

Sample Function Call

```
http://www.company.com/cgi-bin/arswww.cgi?_function=getnotes
&_server=myzos&_user=web&_password=web
&_folder=credit%20card%20statements
&_docid=6850-6851-SUA17-1FAAA-225712-1634-132014-132172-89-76-11-25-0
&_logoff=1
```

Appendix B. Deploying the Java servlet

This section explains how to deploy the Java servlet version of ODWEK on the system.

The Java servlet acts as a controller of your Web application, performing functions and common tasks before and after an action, such as management of the connection to the OnDemand server.

Functions are provided for typical application tasks:

- log on and log off
- search
- retrieve, print, and update documents
- add and view annotations
- change password

You use a set of application functions and parameters to use the servlet in your application.

The Java servlet uses the same functions as the CGI program. See Appendix A, "Deploying the CGI program," on page 89 for a reference of the functions, descriptions, and parameters.

Before you begin

Before you begin deploying the servlet:

- Make sure that you have completed the software installation. See Chapter 4, "Installing ODWEK," on page 15.
- Make sure that you have the current version of the IBM HTTP server installed, configured and operating on the system.
- Make sure that you have the current version of the IBM WebSphere Application Server installed, configured and operating on the system.

IBM recommends that you use the WebSphere tools to deploy the servlet. The WebSphere tools automatically configure the HTTP server and Web application server configuration files. An experienced Web server administrator may choose not to use the WebSphere tools, and deploy the servlet by manually configuring the HTTP server and Web application server configuration files. See the HTTP server and WebSphere information for directions.

To use the WebSphere tools to deploy the servlet:

1. "Copying files."
2. "Deploying the servlet using WebSphere tools" on page 151.

Copying files

Complete the following steps:

1. In most installations, the Web application server locates servlet class files in the servlet root directory:

   ```
   <server_root>/lib
   ```

For example: `c:\WebSphere\AppServer\lib` or `/opt/WebSphere/AppServer/lib`

Copy the `ArsWWWServlet.jar` file to this directory.

2. Copy the `ArsSVTInterface.class` file to a directory that is set by the CLASSPATH variable for the Web application server. Because this file is part of a package (`com.ibm.edms.od`), you must mirror the package structure as subdirectories under this directory.

   For example, if the directory `<server_root>/classes` is set by the CLASSPATH variable, then you must copy the `ArsSVTInterface.class` file to the `<server_root>/classes/com/ibm/edms/od` directory.

3. Copy the shared library to a directory that is set by the shared library path variable:

Table 16. Shared Library Directory

Operating System	Shared Library Path Variable	Shared Library
AIX	LIBPATH	libarswwwsl32.a
HP-UX	SHLIB_PATH	libarswwwsl32.sl
Linux	LD_LIBRARY_PATH	libarswwwsl32.so
Solaris	LD_LIBRARY_PATH	libarswwwsl32.so
Windows	PATH	arswwwsl.dll

4. For Windows systems, copy these files to the directory in which you copied the shared library:

   ```
   ARSSCKNT.DLL
   ARSCT32.DLL
   ```

5. Copy these files:

   ```
   ARSWWW.INI
   AFP2HTML.INI
   AFP2PDF.INI
   ```

   to the HTTP server `bin` directory:

Table 17. HTTP Server Directory

Operating System	HTTP Server Directory
AIX	/usr/lpp/IBM HTTP Server/bin
HP-UX	/opt/IBM HTTP Server/bin
Linux	/opt/IBM HTTP Server/bin
Solaris	/opt/IBM HTTP Server/bin
Windows	c:\IBM HTTP Server\bin

Your next step

To use the WebSphere tools to deploy the servlet, go to “Deploying the servlet using WebSphere tools” on page 151.

Deploying the servlet using WebSphere tools

The WebSphere tools are the recommended method for deploying the servlet. The WebSphere tools can perform all of the tasks required to deploy the servlet.

To deploy the servlet using the WebSphere tools:

1. Assemble the application with the WebSphere Application Assemble Tool (AAT). See "Assembling the application."
2. Install the application from the WebSphere administration console. See "Installing the application" on page 153.

Assembling the application

1. Start the WebSphere Application Assembly Tool (AAT).

Table 18. Starting AAT

Operating System	Start Command
AIX	/usr/lpp/WebSphere/AppServer/bin/assembly.sh
HP-UX	/opt/WebSphere/AppServer/bin/assembly.sh
Linux	/opt/WebSphere/AppServer/bin/assembly.sh
Solaris	/opt/WebSphere/AppServer/bin/assembly.sh
Windows	Start –> Programs –> IBM WebSphere –> Application Server v5.0 –> Administrator's Console

2. Click Create Application Wizard.
3. In the Display Name field, type a name for your ODWEK application. For example: `OnDemand WEK`.
4. In the File Name field, type a name for the configuration files. For example: `odwek.ear`.
5. Click Next.
6. Click Next at the Adding Supplementary Files window.
7. Click Next at the Choosing Application Icons window.
8. Click Next at the Adding EJB Modules window.
9. Click Next at the Adding Web Modules window
10. Click Next at the Adding Application Client Modules
11. Click Finish. A window will open in the AAT with the details of the application that you are creating.
12. Right click on Web Modules and select New from the pop-up list.
13. Complete the fields listed in Table 19.

Table 19. Assembling the application

Field	Value
File Name Specifies the name of the configuration file.	odwek.war
Context Root Specifies the name of the virtual root directory for this module.	/od

Table 19. Assembling the application (continued)

Field	Value
Classpath Specifies the location of the JAR file, from Step 1 on page 149.	/usr/lpp/WebSphere/AppServer/lib (AIX) /opt/WebSphere/AppServer/lib (HP-UX) /opt/WebSphere/AppServer/lib (Linux) /opt/WebSphere/AppServer/lib (Solaris) c:\websphere\appserver\lib (Windows)
Display Name Specifies the name of this module.	ODWEK Module

14. Click OK.
15. Expand ODWEK Module (**Display Name**) by clicking on the +
16. Right click on Web Components and select New from the pop-up list.
17. In the Component Name field, type `ArsWWWServlet`, which is case sensitive.
18. In the Component Type field, click Servlet.
19. For Class Name, click Browse.
20. Locate the ArsWWWServlet.jar file. The location of the JAR file was determined in Step 1 on page 149.
21. Expand the path COM –> IBM –> EDMS –> OD and click on the ArsWWWServlet.class file.
22. Click OK.
23. Expand the ArsWWWServlet Web component.
24. Right click on Initialization Parameters and select New from the pop-up menu.
25. Complete the fields listed in Table 20.

Table 20. Assembling the application

Field	Value
Parameter Name Specifies the parameter to be passed to this module. Important: The value is **case sensitive**.	ConfigDir
Parameter Value Specifies the location of the ARSWWW.INI file. See Step 5 on page 150.	/usr/lpp/IBM HTTP Server/bin (AIX) /opt/IBM HTTP Server/bin (HP-UX) /opt/IBM HTTP Server/bin (Linux) /opt/IBM HTTP Server/bin (Solaris) c:\IBM HTTP Server\bin (Windows)

26. Click OK.
27. Right click on Servlet Mapping and select New from the pop-up menu. The servlet name will already be filled in. You need to specify the URL pattern that you would like to use when calling the servlet from within the Web browser. The URL pattern includes a user-defined name, for example: `arswww`.

    Important: You must specify the URL pattern in the following format: `/arswww/` , where `arswww` is the user-defined name.
28. Click Apply.
29. Click File from the top of the ATT task bar and select Save As from the menu.

30. Choose a location to save the configuration (such as the directory from Step 1 on page 149) and specify a name for the file (such as `odwek.ear`).

Next, install the application. See "Installing the application."

Installing the application

Prerequisite: You must assemble the application before you proceed. See "Assembling the application" on page 151

1. Start the WebSphere administration console.

Table 21. Starting WebSphere administration console

Operating System	Start Command
AIX	/usr/lpp/WebSphere/AppServer/bin/adminclient.sh
HP-UX	/opt/WebSphere/AppServer/bin/adminclient.sh
Linux	/opt/WebSphere/AppServer/bin/adminclient.sh
Solaris	/opt/WebSphere/AppServer/bin/adminclient.sh
Windows	Start –> Programs –> IBM WebSphere –> Application Server v5.0 –> Administrator's Console (Or run adminclient.bat from the WebSphere bin directory.)

2. From the WebSphere administration console, expand WebSphere Admin Domain.
3. Right click on Enterprise Applications and selected Install Enterprise Application from the pop-up menu.
4. Choose the Node for this application.
5. Click Browse to specify the location of the odwek.ear file.
6. Type an Application Name. For example: ODWEK.
7. Click Next.
8. Click Next at Mapping User to Roles.
9. Click Next at Mapping EJB RunAs.
10. Click Next at Binding Enterprise Beans.
11. Click Next at Mapping EJB References.
12. Click Next at Mapping Resource References.
13. Click Next at Specify the Default Datasource.
14. Click Next at Specifying datasources for CMP Beans.
15. Specify the Virtual Host for your Web Module. For example, `Default Host`.
16. Click Next.
17. Select the Application Server for your Application. For example, `Default Server`.
18. Click Next.
19. Click Finish.
20. Expand Nodes.
21. Right Click on your node and select Regen WebSphere Plugin from the pop-up menu.
22. Expand Enterprise Applications.
23. Right click on your ODWEK application and select Start from the pop-up menu.

24. Stop and restart the Web application server.

Specifying the ARSWWW.INI file

Important: The `ARSWWW.INI` file is used with CGI program or Java servlet only. Do not use it with the Java API interface.

After you have deployed the Java servlet using the WebSphere tools, configure the ODWEK initialization file for your operating environment, that is, the `ARSWWW.INI` file. See “Specifying the ARSWWW.INI file” on page 89.

Appendix C. No HTML output

ODWEK uses the _nohtml directive to determine the type of output generated by a function (such as Logon). By default, ODWEK generates HTML output. If you specify _nohtml=1, then ODWEK generates delimited ASCII output. This chapter describes the delimited ASCII output generated by ODWEK.

Delimited ASCII output

The delimited ASCII output generated by ODWEK is a set of output records that contain character string values, keywords, and function, record, and string delimiters and separators:

- Character string values are output data of a function, other than keywords, delimiters, and separators. For example, the next function to be called, the name of the folder, the folder field names, search operators, and field values are character string values.
- Keywords consist of a specific character string. For example, ACTION, DOC, FOLDER, NUMROWS, and ROW are keywords.
- The function delimiters consist of the specific character strings [BEGIN] and [END].
- The record delimiter is the new line character, \n. All records are delimited by the new line character.
- By default, string delimiters and separators are the caret character (^) and the left bracket ([) and right bracket (]) characters. For example:

  ```
  [folderName^folderDesc]
  ```

  If a keyword record contains more than one character string value, then the values are separated by the caret character. Each keyword's set of character string values is delimited by the left bracket and right bracket characters.

  Some character string values may be stored in a list, separated by the caret character and enclosed in left bracket and right bracket characters. For example, the list of valid search operators for a field may appear as follows:

  ```
  [1^2^4^8^16^32]
  ```

  You can override the default characters for the string delimiters and separators. See "[NO HTML]" on page 109 for details.
- A single null character string value is indicated by the absence of a value inside two double quote characters (""). A null list is indicated by the absence of a value inside left bracket and right bracket characters ([ ]).

Logon

The following shows an example of the delimited ASCII output generated by the Logon function:

```
[BEGIN]\n
ACTION=searchCriteriaUrl\n
FOLDER=[folderName^folderDesc]\n
FOLDER=[folderName^folderDesc]\n
```

```
⋮

[END]\n
```

Notes

1. The string `searchCriteriaUrl` identifies the name of the next function to be executed and its parameters.
2. The string `folderName` identifies a folder name. The name is not quoted.
3. The string `folderDesc` is the description of the folder. The description is not quoted.

Search Criteria

The following shows an example of delimited ASCII data generated by the Search Criteria function:

```
[BEGIN]\n
ACTION=hitListUrl\n
DISPLAY_ORDER=[field1^field2^...fieldN]\n
NUMROWS=numberOfRows\n
ROW=[criteriaName^[[validOp]^defOp]^[inpType^inpAssocData]\n

⋮

[END]\n
```

Notes

1. The string `hitListUrl` identifies the name of the next function to be executed and its parameters.
2. The `DISPLAY_ORDER` keyword specifies the order in which the folder fields should be displayed.
3. The string `numberOfRows` identifies the number of `ROW` keyword records that follow. The function generates one `ROW` keyword record for each search field.
4. The string `criteriaName` represents the search criteria for a search field. The search criteria is not quoted.
5. The string `validOp` is the list of integer values that represent the valid search operators for the search field:
   1 Equal
   2 Not Equal
   4 Less Than
   8 Less Than or Equal
   16 Greater Than
   32 Greater Than or Equal
   64 In
   128 Not In
   256 Like
   512 Not Like
   1024 Between
   2048 Not Between
6. The string `defOp` is an integer value representing the default search operator.
7. The string `inpType` represents the type of search field:
   A Annotation Text Search
   C Choice

N Normal
S Segment
T Text Search
Z Annotation Color Search

8. The string `inpAssocData` is a list associated with the `defOp` and `inpType` listed in Table 22.

Table 22. Default operator and input type associated with inpAssocData

defOp	inpType	inpAssocData

Between, Not Between	N	Null: [ ]
		or a list: [defaultField1∧...∧defaultField*N*]
		For example:
		`["01/31/96"∧"01/31/97"]`
		`["01/31/96"∧""]`
		`[""∧"01/31/97"]`

Other valid operators	A, N, T, Z	Null: [ ] or a single string value that represents the default field value

Other valid operators	C, S	[[listOfChoices]∧defaultChoice]
		For example:
		`[["JFIF"∧"TIFF"∧"PCX"]∧"TIFF"]`
		`[["JFIF"∧"TIFF"∧"PCX"]∧""]`

Document Hit List

The following shows an example of delimited ASCII output generated by the Document Hit List function:

```
[BEGIN]\n
ACTION=hitListURL\n
MSG=Only 20 documents can be listed for this folder.
DOC=[criteria1∧criteria2∧criteriaN∧docid∧fileType∧docLocation]\n

⋮

[END]\n
```

Notes

1. The string `hitListURL` identifies the name of the next function to be executed and the parameters for the function.
2. The `MSG` keyword shows an example of an error message in the delimited ASCII output. By default, ODWEK sends error messages to the client. However, when a function contains the `_nohtml=1` directive, ODWEK generates the message text in the delimited ASCII output instead.
3. The strings `criteria1`, `criteria2`, and `criteria`*N* represent search criteria values. The values are listed in the order in which they appear in the document list. The values are not quoted.
4. The string `docid` is the document identifier for the document.
5. The string `fileType` identifies the data type of the document:
   A AFP
   B BMP
   E Email
   F JFIF
   G GIF

L Line
N None
O OD Defined
P PDF
T TIFF
U User Defined
X PCX

6. The string `docLocation` identifies the storage location of the document:
   0 Unknown
   1 OnDemand cache storage
   2 Archive storage
   3 External cache storage

View Annotations

The following shows an example of delimited ASCII output generated by the View Annotations function:

```
[BEGIN]\n
NOTE 4: 15:42:44 PM Mountain Standard Time Thursday November 19, 1998...\n
Public - Cannot be copied to another server\n
Test note from the OnDemand Internet Client.\n
[END]\n
```

Error Message

The following shows an example of delimited ASCII output generated when errors occur:

```
[ERROR]\n
ID=nnnn\n
MSG=errorMessageText\n
```

Notes

1. The string `nnnn` is the error message number.
2. The string `errorMessageText` is the error message text.

Appendix D. National language support

Configuring ODWEK for DBCS languages

This section contains information that may help administrators configure ODWEK for DBCS languages.

The CODEPAGE and LANGUAGE parameters in the ARSWWW.INI file are used to specify National Language (NL) configuration options.

The CODEPAGE parameter identifies the code page of the ODWEK server and needs to be compatible with Content Manager OnDemand database on the Content Manager OnDemand library server. The CODEPAGE parameter need be specified **only** if the code page of the workstation on which you are running your ODWEK application is different than the code page of the Content Manager OnDemand database on the Content Manager OnDemand library server. The system uses the code page of the workstation on which the ODWEK application is running as the default value.

The LANGUAGE parameter determines the message catalog that ODWEK uses to display messages.

Table 23 lists the DBCS code pages and languages supported by Content Manager OnDemand. The **CODEPAGE=** column lists the value for the code page, and need be specified **only** if the code page of the workstation on which you are running your ODWEK application is different than the code page of the Content Manager OnDemand database. The **LANGUAGE=** column lists the values that are associated with the translated message catalogs.

Table 23. DBCS code pages, languages, code sets and locales

Territory	LANGUAGE=	Operating system	Database Code Page	CODEPAGE=	Code Set	Locale
China (PRC)	CHS	AIX	1383	1383	IBM_eucCN	zh_CN
		HP-UX	1383	1383	hp15CN	zh_CN. hp15CN
		Solaris	1383	1383	gb2312	zh
		Windows	1386	1386	GBK	—
		z/OS (EBCDIC)	935	935	IBM-935	—
		Linux	1386	1386	GBK	zh_CN.GBK
Japan	JPN	AIX	954	954	IBM_eucJP	ja_JP
		HP-UX	954	954	eucJP	ja_JP.eucJP
		Solaris	954	954	eucJP	ja
		Windows	943	943	IBM-943	—
		z/OS (EBCDIC)	939	939	IBM-939	—
		Linux	954	954	EUC-JP	ja_JP

Table 23. DBCS code pages, languages, code sets and locales (continued)

Territory	LANGUAGE=	Operating system	Database Code Page	CODEPAGE=	Code Set	Locale
Korea, South	KOR	AIX	970	970	IBM_eucKR	ko_KR
		HP-UX	970	970	eucKR	ko_KR.eucKR
		Solaris	970	970	5601	ko
		Windows	1363	1363	1363	—
		z/OS (EBCDIC)	933	933	IBM-933	—
		Linux	970	970	EUC-KR	ko_KR
Taiwan	CHT	AIX	964	964	IBM_eucTW	zh_TW
		HP-UX	964	964	eucTW	zh_TW.eucTW
		Solaris	964	964	cns11643	zh_TW
		Windows	950	950	big5	—
		z/OS (EBCDIC)	937	937	IBM-937	—
		Linux	950	950	BIG5	zh_TW

Code page conversion in ODWEK

In contrast to the standard Content Manager OnDemand client, ODWEK behaves a little differently when it converts code pages. Because ODWEK is a mid-tier system, there is always an additional presentation layer. In most cases, the layer is a browser; however, it can also be a stand-alone Java application that uses the ODWEK API.

ODWEK internally runs completely in UTF-8, which leads to a conversion of all index and annotation data from UTF-16 (the format in which the data are sent in TCP/IP) to UTF-8.

When you implement a Java application that uses ODWEK, ensure that you correctly handle the information that you pass to and retrieve from ODWEK API. Because Java works in UTF-16 Unicode internally, data that are returned from ODWEK API functions are not of any concern. Java handles the conversion from UTF-8 itself. When you pass strings to ODWEK methods, you do not need to perform any additional tasks, because Java supports only string variables that are in UTF-16. The conversion to UTF-8 is done by native subroutines of ODWEK.

Despite the internal conversion to UTF-8, ODWEK does not do any other conversion on indexes. Therefore, a client that displays index data that are received through ODWEK API needs to be capable of handling UTF-8 Unicode data. If you deliver index data to any external applications using your ODWEK-based Java application, you need to make sure those applications can handle Unicode data or you need to convert the data manually. The same implications apply if you want to save any indexes or annotation data to file. If you do not perform any explicit conversion, the data is written as Unicode datastream. As Web browsers usually are capable of displaying and sending UTF-8 data, it is not a problem when you implement Web applications.

For the actual document data, you can handle the conversion in different ways. If you request raw native document data, ODWEK returns the data in its unaltered

form – in the same codepage in which it was archived. When you request the line data to be displayed as applet, ODWEK internally sends the UTF-8 ASCII data to the applet, but you get only standard HTML code containing applet invocation code back. When you request an ASCII conversion, ODWEK returns a UTF-8 ASCII converted representation of the original AFP or line data document. For most other document types, ODWEK works the same as the OnDemand Windows client – it passes the data through in the data's native format.

ICU conversion library

Sometimes, Content Manager OnDemand might need to convert or map from one code page to another code page. This conversion or mapping is done by using a standard component called ICU – International Components for Unicode.

The International Components for Unicode (ICU) is an open source project developed by IBM and other companies. It is a library that is available for Java and C, and is used for internationalization. ICU provides services such as character conversion between different code pages and language sensitive collation, searching, normalization and locale information.

Earlier versions of Content Manager OnDemand used the ICONV library, another character set conversion engine. Starting from Version 7.1.2.1, Content Manager OnDemand uses ICU. ICONV is a character encoding library that is primarily distributed on UNIX environments, for example, it is part of the GNU C library in most Linux distributions.

Content Manager OnDemand uses ICU at different locations for code page conversion and text operations.

Each component uses ICU for different use cases, but all do code page conversions, which enables communication with other parts of the Content Manager OnDemand infrastructure:

- The Content Manager OnDemand server uses the ICU facility for index and annotation communication. All TCP/IP traffic is in UTF-16, but the index data is in the database code page format, the Content Manager OnDemand server uses ICU for data conversion between the database or instance code page and UTF-16.
- ODWEK needs the ICU facility to convert TCP/IP data in UTF-16 to UTF-8. Also, if you use line data in the line data Java applet, then the actual data gets converted from its original code page to UTF-8, which is used by the applet. Other documents are passed over without code page change.
- The standard Content Manager OnDemand client uses its ICU facility for converting AFP and line data so that they can be properly viewed in the internal viewers. The data gets converted to the local Windows code page.
- The AFP plugin, which works similar to the AFP viewer within the Content Manager OnDemand client, has its own ICU library for converting AFP data into the local Windows code page.

Appendix E. Problem determination tools

You can use the tools listed in Table 24 to gather information about the system and documents. You can use the information to help solve problems you are having configuring ODWEK and help other people in your organization who are having problems using the applets and viewers.

Table 24. Problem determination tools

Tool	Purpose	How to Enable
HTML Output	Save a copy of the HTML that ODWEK is returning to the browser.	Choose Save As from the browser's File menu
Content Manager OnDemand Web Enablement Kit System Tracing	Save access information, errors, and server information.	Do the following: 1. In the DEBUG section of the ARSWWW.INI file, set the `Trace` parameter to `4`. For the Java APIs, use the appropriate ODConfig constructor to specify this. ODWEK generates ARSWWW.TRACE and writes it to the directory that is specified by the `TRACEDIR` parameter. (The default directory is `/tmp`.) 2. Configure logging for your HTTP server. (Each HTTP server may have a different way to configure logging and may have different logs and options you can enable to collect more or less detailed information.) **Important:** Because a significant amount of information can be written to a log file, IBM recommends that you enable logging only when needed, such as when recreating a problem. If you need to enable logging for extended periods of time, make sure that the log file paths point to storage devices with plenty of free space. Remember to periodically delete old log files from the server.

AFP Web Viewer Trace Facility	Capture detailed information about AFP documents being viewed with the AFP Web Viewer.	Make sure the following section exists in the FLDPORT2.INI file on the user's workstation:
		`[Misc]`
		`ViewTraceFile=d:\temp\afpplgin.log`
		`Trace=TRUE`
		
		Verify the path of the log file. Remember to turn off logging when you have gathered the information you need.

Table 24. Problem determination tools (continued)

Tool	Purpose	How to Enable
OnDemand System Log	Save system messages (such as log on and log off) and application group messages having to do with documents (such as query and retrieve) and annotations.	Complete the following steps: 1. Enable system and application group logging for the OnDemand server. Update the system parameters for the server using the administrative client. 2. Enable the specific application group messages that you want to log. Update the message logging options for the application group using the administrative client.

Java Dump

During the run time of a Java process, some Java Virtual Machines (JVMs) might not respond predictably and oftentimes seem to hang up for a long time or until a JVM shutdown occurs. It is not easy to determine the root cause of these problems.

By triggering a javacore when a Java process does not respond, you might be able to collect diagnostic information that is related to the JVM and a Java application captured at a particular point. For example, the information can be about the operating system, the application environment, threads, native stack, locks, and memory. The exact contents depend on the platform on which the application is running.

On some platforms, and in some cases, a javacore is known as a "javadump." The code that creates a javacore is part of the JVM. You can control it by using environment variables and run-time switches. By default, a javacore occurs when the JVM terminates unexpectedly. A javacore can also be triggered by sending specific signals to the JVM. Although a javacore or javadump is present in Sun Solaris JVMs, much of the content of a javacore is added by IBM and, therefore, is present only in IBM JVMs.

The IBM Thread and Monitor Dump Analyzer for Java analyzes javacore and diagnoses monitor locks and thread activities to identify the root cause of hangs, deadlocks, and resource contention or monitor bottlenecks.

IBM Thread and Monitor Dump Analyzer

This technology analyzes each thread and provides diagnostic information, such as current thread information, the signal that caused the javacore, Java heap information (maximum Java heap size, initial Java heap size, garbage collector counter, allocation failure counter, free Java heap size, and allocated Java heap size), number of runnable threads, total number of threads, number of monitors locked, and deadlock information.

In addition, IBM Thread and Monitor Dump Analyzer for Java Technology provides the recommended size of the Java heap cluster (applicable only to IBM SDK Version 1.4.2 and Version 1.3.1 SR7 or above) based on the heuristic analysis engine.

IBM Thread and Monitor Dump Analyzer for Java compares each javacore and provides process ID information for threads, time stamp of the first javacore, time stamp of the last javacore, number of garbage collections per minute, number of

allocation failures per minute, time between the first javacore and the last javacore, number of hang suspects, and list of hang suspects.

This technology also compares all monitor information in javacore and detects deadlock and resource contention or monitor bottlenecks, if there are any.

For more information on IBM Thread and Monitor Dump Analyzer, see http://www.alphaworks.ibm.com/tech/jca.

Java diagnostic commands

jmap

jmap is a command line utility that is included in the Solaris Operating Environment and Linux (but not Windows) releases of the Java Development Kit (JDK). This utility prints memory–related statistics for a running JVM or core file. If jmap is used without any command line options, then it prints the list of shared objects loaded, similar to what the Solaris pmap utility outputs. For more specific information, you can use the –heap, –histo, or –permstat options.

-heap Use the –heap option to obtain information that includes the name of the garbage collector, algorithm–specific details (such as the number of threads used for parallel garbage collection), heap configuration information, and a heap usage summary.

-histo Use the -histo option to obtain a class–wise histogram of the heap. For each class, it prints the number of instances in the heap, the total amount of memory consumed by those objects in bytes, and the fully qualified class name. The histogram is useful when you want to understand how the heap is used.

-permstat
Configuring the size of the permanent generation can be important for applications that dynamically generate and load a large number of classes (for example, Java Server Pages and web containers). If an application loads too many classes, then an `OutOfMemoryError` exception is thrown. Use the –permstat option to the jmap command to get statistics for the objects in the permanent generation.

jstat

The jstat utility uses the built–in instrumentation in the HotSpot JVM to provide information on performance and resource consumption of running applications. You can ue the jstat utility to diagnose performance issues, especially issues that are related to heap sizing and garbage collection. Some of its many options can print statistics regarding garbage collection behavior and the capacities and usage of the various generations.

HPROF: Heap Profiler

HPROF is a simple profiler agent that is shipped with JDK Version 5.0. It is a dynamically–linked library that interfaces to the JVM by using the Java Virtual Machine Tools Interface (JVM TI). It writes out profiling information either to a file or to a socket in ASCII or binary format. This information can be further processed by a profiler front–end tool.

HPROF can present CPU usage, heap allocation statistics, and monitor contention profiles. In addition, it can output complete heap dumps and report the states of

all the monitors and threads in the Java virtual machine. HPROF is useful when you analyze performance, lock contention, memory leaks, and other issues.

HAT: Heap Analysis Tool

The Heap Analysis Tool (HAT) helps debug unintentional object retention. This term is used to describe an object that is no longer needed but is kept alive due to references through some path from a live object. HAT provides a convenient means to browse the object topology in a heap snapshot that is generated using HPROF. The tool allows a number of queries, including “show me all reference paths from the rootset to this object.

Diagnostic Tool for Java Garbage Collector

This technology is a diagnostic tool for optimizing parameters affecting the garbage collector when using the IBM JavaTM Virtual Machine (JVM).

Applications that run under Java use what is known as a Java heap for garbage collection, which serves as a storage manager in IBM's Java development kits and run-time environments.

An analysis of data reflecting the activity of the garbage collector in Java enterprise or stand-alone applications is critical to optimizing tasks running under a JVM. For example, issues such as the frequency of the garbage collection cycle, the time spent in different phases of the garbage collection, the quantities of heap memory involved in the process, the characteristics of the allocation failures from which the garbage collection originate, and the unwanted presence of stack overflows must be considered in optimization of parameters for Java applications and prevention of bottlenecks.

Diagnostic Tool for Java Garbage Collector helps to examine the characteristics of the garbage collection for an application running under a JVM by reading the output of the "verbose" garbage collection and producing textual and graphical visualizations and related statistics. This tool is particularly well suited for looking at the garbage collector activity of a heavily-accessed enterprise application hosted by a WebSphere Application Server. The tool includes a "multiple file analysis" modality that allows the loading of two or more files simultaneously, thereby enabling the comparison of the behavior of two or more application servers in a WebSphere cluster.

Diagnostic Tool for Java Garbage Collector Version 1.3 comes with three built-in parsers for IBM JVM Version 1.5.0, IBM JVM Version 1.4.2, and IBM JVM Version 1.2.2, as well as sample files for evaluating features of the tool and detailed documentation. Each parser is likely to be able to parse the data that are produced by some other versions of the IBM JVM; for instance, the JVM Version 1.4.2 parser also works well with the JVM Version 1.3.x. For unsupported versions of the IBM JVM, the tool contains detailed documentation that allows one to code a different parser in Java by implementing the GCParser interface. The documentation describes in detail the assumptions and the rules necessary for coding of the new parser

For more additional information and product download, see http://www.alphaworks.ibm.com/tech/gcdiag.

HeapAnalyzer

HeapAnalyzer finds a possible Java heap leak area through its heuristic search engine and analysis of the Java heap dump in Java applications.

Java heap areas define objects, arrays, and classes. When the Garbage Collector allocates areas of storage in the heap, an object continues to be live while a reference to it exists somewhere in the active state of the JVM; therefore, the object is reachable. When an object ceases to be referenced from the active state, it becomes garbage and can be reclaimed for reuse. When this reclamation occurs, the Garbage Collector must process a possible finalizer and also ensure that any internal JVM resources that are associated with the object are returned to the pool of such resources. Java heap dumps are snap shots of Java heaps at specific times.

HeapAnalyzer analyzes Java heap dumps by parsing the Java heap dump, creating directional graphs, transforming them into directional trees, and running the heuristic search engine.

The following are examples of features of HeapAnalyzer:

- List of Java heap leak suspects
- Recommendation of the size of kCluster
- List of gaps among allocated objects/classes/arrays
- Java objects/classes/arrays search engine
- List of objects/classes/arrays by type name
- List of objects/classes/arrays by object name
- List of objects/classes/arrays by address
- List of objects/classes/arrays by size
- List of objects/classes/arrays by size of child
- List of objects/classes/arrays by number of child
- List of objects/classes/arrays by frequency
- List of available heap spaces by size
- Tree view of Java heap dump Loading/saving processed Java heap dumps

HeapRoots

HeapRoots is a tool for debugging memory leaks in Java applications through analysis of "heap dumps."

The Java Virtual Machine (JVM) maintains a run-time data area (called a heap) for the allocation of all class instances and array objects. The heap storage for objects is automatically reclaimed by a storage management system known as the garbage collector. If an application requires more heap space than can be made available by the garbage collector, the JVM throws an OutOfMemoryError.

HeapRoots analyses "heap dumps," which are files (typically text files) containing information about the objects in the JVM garbage collected heap.

Some IBM VMs (contained in the IBM Developer Kits for Windows, Java Edition) have the ability to produce heap dumps on demand; heap dumps can also be triggered by out-of-memory situations.

HeapRoots loads these heap dump files and provides commands for analyzing the data. These commands run algorithms on the data or query for information about

the data. HeapRoots provides a command-line interactive interface where one enters commands and gets results. Examples of analysis include the following:

- Searching or filtering of individual objects
- Summary or tabulation of the various types of objects
- Statistics on heap address space (such as gaps between objects)
- Inward and outward references of an object
- Paths between two objects
- Exploring the heap from source or root objects by following references
- Calculation of objects reachable by an object
- Calculation of objects kept alive by an object.

Appendix F. Uninstalling ODWEK

To uninstall ODWEK on AIX®, use the following command:

```
/usr/lpp/ars/www/_uninst/uninstallodwek
```

or

```
/usr/lpp/ars/www/_uninst/uninstallodwek -console
```

To uninstall ODWEK on Solaris, HP-UX, Linux®, or Linux on zSeries®, use the following command:

```
/opt/ondemand/www/_uninst/uninstallodwek
```

or

```
/opt/ondemand/www/_uninst/uninstallodwek -console
```

You can also use the Java™ command to uninstall ODWEK.

Important: Use the Java command only if you cannot use the above commands because the system cannot find a suitable JVM.

On AIX, enter this command:

```
java -jar /usr/lpp/ars/www/_uninst/uninstallodwek.jar
```

or

```
java -jar /usr/lpp/ars/www/_uninst/uninstallodwek.jar -console
```

On Solaris, HP-UX, Linux, or Linux on zSeries, enter this command:

```
java -jar /opt/ondemand/www/_uninst/uninstallodwek.jar
```

or

```
java -jar /opt/ondemand/www/_uninst/uninstallodwek.jar -console
```

To uninstall ODWEK on Windows®, use the InstallShield uninstaller.

Appendix G. Accessibility for the line data applet and the AFP plugin

The line data applet and the AFP plugin can be used in your Web client to view those data formats. They are not fully accessible at this time, therefore, if you need accessible viewers, use the Content Manager OnDemand client.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample

programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at `www.ibm.com/legal/copytrade.shtml`

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Portions of the OnDemand Windows client program contain licensed software from Pixel Translations Incorporated, © Pixel Translations Incorporated 1990, 2003. All rights reserved.

Other company, product or service names may be trademarks or service marks of others.

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

Program Number: 5724-J33

SC19-2941-00