IBM Content Manager OnDemand for iSeries Common Server

Web Enablement Kit Installation and Configuration Guide

Version 5 Release 3

SC27-1163-02

IBM Content Manager OnDemand for iSeries Common Server

Web Enablement Kit Installation and Configuration Guide

Version 5 Release 3

SC27-1163-02

Note

Before using this information and the product it supports, read the information in “Notices” on page 143.

Third Edition (May 2004)

This edition applies to IBM Content Manager OnDemand for iSeries Common Server, Version 5 Release 3 and to all subsequent releases and modifications until otherwise indicated in new editions. This edition replaces SC27-1163-01.

Contents

About IBM Content Manager OnDemand for iSeries Common Server Web Enablement Kit Installation and Configuration Guide (SC27-1163)

This book provides information that you can use to plan for, install, configure, and use IBM® Content Manager OnDemand for iSeries™ Version 5 Release 3 Common Server (OnDemand) Web Enablement Kit.

Who should read this book

This book is intended primarily for system administrators who need to implement, install, and maintain the OnDemand Web Enablement Kit (ODWEK) software and applications. It can also be used by programmers who need to integrate OnDemand with Web applications.

How this book is organized

This book provides the information that you need to install and configure ODWEK and plan for users to access data from an IBM Content Manager OnDemand for iSeries Common Server system with a Web browser. This publication contains the following sections:

- Chapter 1, "Overview," on page 1
- Chapter 2, "Installing and configuring the HTTP server," on page 9
- Chapter 3, "Configuring the sample applications," on page 41
- Chapter 4, "Installing the Web viewers," on page 45
- Appendix A, "CGI API reference," on page 57
- Appendix B, "Java servlet reference," on page 85
- Appendix C, "Java API reference," on page 87
- Appendix D, "Java API programming guide," on page 89
- Appendix E, "AFP to HTML transform," on page 127
- Appendix F, "AFP to PDF transform," on page 131
- Appendix G, "HTTP server configuration files," on page 133
- Appendix H, "No HTML output," on page 135
- Appendix I, "National language support," on page 139
- Appendix J, "Problem determination tools," on page 141

Prerequisite and related information

Use the IBM iSeries Information Center as your starting point for looking up iSeries technical information.

You can access the Information Center two ways:

- From the following Web site: `http://www.ibm.com/eserver/iseries/infocenter`
- From CD-ROMs that ship with your Operating System/400® order:

  iSeries Information Center, SK3T-4091-04. This package also includes the PDF versions of iSeries manuals, *iSeries Information Center: Supplemental Manuals*, SK3T-4092-01, which replaces the Softcopy Library CD-ROM.

The Information Center contains advisors and important topics such as Java™, TCP/IP, Web serving, secured networks, logical partitions, clustering, CL commands, and system application programming interfaces (APIs). It also includes links to related IBM Redbooks™ and Internet links to other IBM Web sites such as the IBM home page.

Other information available on the World Wide Web

More iSeries information is available on the World Wide Web. You can access general information from the iSeries home page, which is at the following Web site: `http://www-1.ibm.com/servers/eserver/iseries/`

To access workshops on advanced iSeries functions, use the Technical Studio, located at: `http://www.iseries.ibm.com/tstudio/`

Worldwide, you can read about, select, order and take delivery of iSeries program temporary fixes (PTF) over the Internet. iSeries Internet PTFs (downloads) and Preventive Service Planning (PSP) information are available at the following Internet location: `http://as400service.ibm.com`

iSeries Navigator

IBM iSeries Navigator is a powerful graphical interface for managing your iSeries servers. iSeries Navigator functionality includes system navigation, configuration, planning capabilities, and online help to guide you through your tasks. iSeries Navigator makes operation and administration of the server easier and more productive and is the only user interface to the new, advanced features of the OS/400®. It also includes Management Central for managing multiple servers from a central system.

You can find more information on iSeries Navigator in the IBM iSeries Information Center and at the following Web site:
`http://www.ibm.com/eserver/iseries/navigator/`

How to send your comments

Your feedback is important to providing the most accurate and high-quality technical documentation. Please send any comments that you have about this publication or other OnDemand documentation. You can use either of the following methods to provide comments:

- Fax your comments from the United States, Canada, and Puerto Rico to 1-800-937-3430 (From other countries: 1-507-253-5192)
- Email your comments to: RCHCLERK@us.ibm.com; for the Information Center, email comments to: RCHINFO@us.bim.com

Be sure to include the following information:

- The name of the book of iSeries Information Center topic
- The publication number of a book (found in the lower right corner of the books front cover
- The page number or topic to which your comment applies

Summary of Changes

This edition of *IBM Content Manager OnDemand for iSeries Common Server: Web Enablement Kit Installation and Configuration Guide* contains new and changed technical information. There may be some instances where changes were made, but change bars are missing. Significant changes to note are:

- At Version 5 Release 1, Content Manager OnDemand for iSeries (OnDemand) introduced a new server implementation known as the OnDemand Common Server. The Common Server provides enhanced indexing, searching, viewing, security, PDF, and web enablement capabilities for OnDemand users and administrators. Current OnDemand customers who have implemented Spool File Archive (with or without AnyStore or the existing Server feature) can now migrate to the new Common Server using the instructions outlined in Appendix A of the Content Manager OnDemand for iSeries Common Server Planning and Installation Guide. Note that, throughout the documentation, reference to the migration of Spool File Archive data also implies AnyStore data as well, if AnyStore is installed.
- Significant additions have been made to the Content Manager OnDemand for iSeries Common Server Indexing Reference publication regarding functions supported by the OS/400 Indexer. These additions include topics related to defining multi-key indexes, transaction fields, text search fields, SCS spooled files with AFP overlays, and masks for application fields.
- Content Manager OnDemand for iSeries now supports the new iSeries-supported Plasmon optical libraries.
- Two command parameters for the Start Archived Storage Management for OnDemand (STRASMOND) command have been removed to make the use of the command simpler. See Appendix A of the Content Manager OnDemand for iSeries Common Server Administration Guide for details.
- OS/400 has withdrawn the original HTTP server support. In conjunction with this, the Content Manager OnDemand Web Enablement Kit (ODWEK) support for the original HTTP server has also been withdrawn. The HTTP Apache server is now the only supported HTTP server for ODWEK.

Chapter 1. Overview

ODWEK allows users to access data that is stored in an IBM Content Manager OnDemand server by using a Web browser or user-written program. For example, you can provide some people with the Uniform Resource Locator (URL) of a Web page that permits them to log on to an OnDemand server; you can provide other people 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 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.

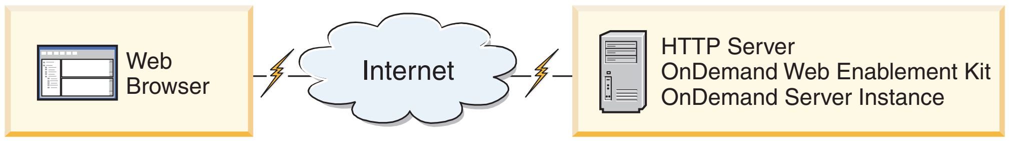

Figure 1. Accessing data stored in OnDemand using ODWEK

| ODWEK can search for and retrieve documents from OnDemand servers that are
| running IBM Content Manager OnDemand for iSeries Common Server Version 5,
| IBM Content Manager OnDemand for Multiplatforms Version 7, and IBM Content
| Manager OnDemand for z/OS® and OS/390®, Version 7.1.

ODWEK contains several components:

- 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. You can use one of the following programming interfaces to control ODWEK:
  - Common Gateway Interface (CGI) program. The CGI program provides a way to access OnDemand data from a Web browser. The CGI program runs on a system that is running an Hypertext Transfer Protocol (HTTP) server, such as the IBM HTTP Server.
  - Java servlet. The CGI program provides a way to access OnDemand data from a Web browser. The servlet runs on a Java-enabled HTTP server that is running a Java application server, such as the IBM WebSphere® Application Server.
  - Java API. The Java API provides a way to access OnDemand data from a user-written program. The Java API requires Java version 1.2.2 or later.
- The IBM OnDemand Advanced Function Presentation™ (AFP™) Web Viewer. The AFP Web Viewer lets users search, retrieve, view, navigate, and print AFP documents from a Web browser.
- The IBM 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.

- The Line Data Java applet. The Line Data applet lets users view line data documents from a Web browser. An administrator enables the use of the Line Data applet by configuring the ARSWWW.INI file.
- The AFP2HTML Java applet. The AFP2HTML applet lets users view the output generated by the IBM AFP2WEB Transform service offering. The AFP2WEB Transform converts AFP documents and resources into Hypertext Markup Language (HTML) files that can be displayed with the AFP2HTML applet. After installing and configuring the AFP2WEB Transform, an administrator enables the use of the AFP2HTML applet by configuring the ARSWWW.INI file.

Note: To view other types of documents stored in 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 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 ASWWW.INI file and ODWEK programming interface, which other instances cannot access. There are three ODWEK programming interfaces:

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

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.

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 involve a single application server instance operating independently of any other applications.

Figure 2 shows an example of a single instance using the CGI interface.

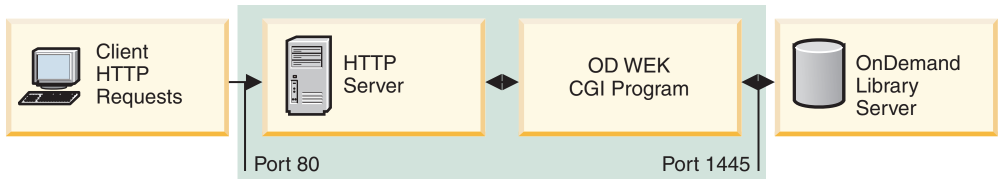

Figure 2. Single instance using CGI interface

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

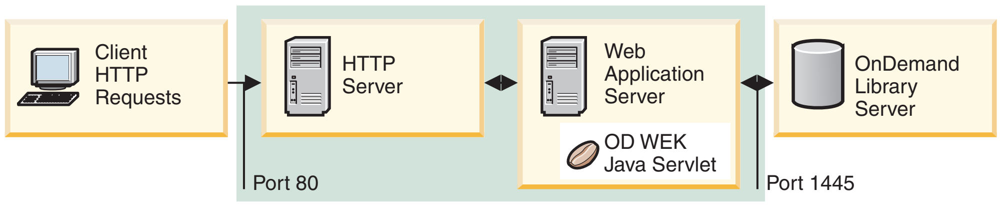

Figure 3. Single instance using Java interface

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

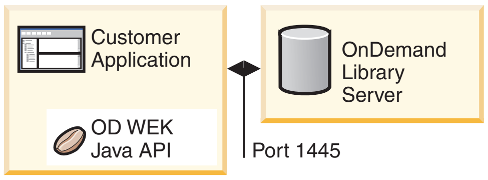

Figure 4. Single instance using Java API interface

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 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 shows an example of the multiple instance topology.

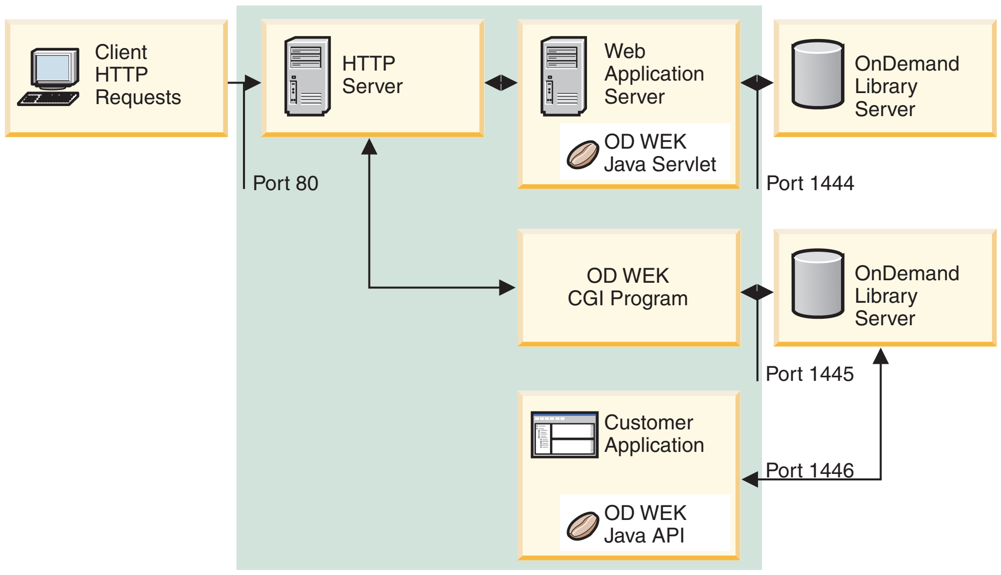

Figure 5. Multiple instance topology

About the viewers

ODWEK provides the following viewers:

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

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 plug-in toolbar provides controls that can help users work with documents. The people in your organization that plan to use the Web viewers to view documents must install them on their workstations.

Note: The installation program will install the viewers as either plug-ins or ActiveX controls. If Internet Explorer is installed on the workstation, then the installation program will install the ActiveX controls; if Netscape is installed on the workstation, then the installation program will install the plug-ins. If you have both Internet Explorer and Netscape installed on the workstation, then the installation program will install the ActiveX controls for Internet Explorer and the plug-ins for Netscape.

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

The AFP2HTML applet lets users view the output generated by the IBM AFP2WEB Transform service offering. The AFP2WEB Transform converts AFP documents and resources into HTML documents. After installing and configuring the AFP2WEB Transform, an administrator enables the use of the AFP2HTML applet by configuring the ARSWWW.INI file. The AFP2HTML applet provides a toolbar with controls that can help users work with documents, including controls for large objects.

One advantage of the applets is that your users never have to install or upgrade software on the workstation to use them, unlike the Web viewers, which must be installed on the workstation. Also, if IBM provides a new version of a Web Viewer, then you must distribute the updated Web Viewer to your 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 client. The client 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 client 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 provided with the product. The LOGON.HTM sample application supports users that are permitted access to several folders. You first modify the LOGON.HTM page with information about your OnDemand server. You 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 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 client. Make sure that your users 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 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 Application Programming Interface (API). See Appendix A, “CGI API reference,” on page 57 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 Annotation Add permission for each application group that contains documents to be annotated. (The Application Group Access permission lets users add annotations.)

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.

| **An important note for customers that have both OnDemand Spool File Archive
| and Common Server environments on the system and are using the
| ARS_MIGR_SERVER entry in the ARS.CFG file to combine Spool File Archive**

and Common Server folders on a single folder selection list: For ODWEK users, the Spool File Archive folders will appear in the ODWEK folder list and can be searched. However, attempting to retrieve a document will fail.

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 an OnDemand server printer. To use server print, the user must be given the Document Print permission for each application group that contains documents that the user needs to print. (The Application Group Access permission lets users print documents.) At least one server printer must be defined on the OnDemand server.

Update Document

The Update Document function allows users to update the database. The Update Document function updates one or more database fields for a specific document.

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 Annotation View permission for each application group that contains annotations that the user needs to view. (The Application Group Access permission lets users view annotations.)

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 OnDemand server

Any user that can access your HTTP server and the programs and Web pages that comprise the front-end to ODWEK can potentially access data stored in 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 HTTP server. For example, many HTTP 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 HTTP server, that requires users to enter a userid and password before accessing the Web pages. However, even though HTTP 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 HTTP server userids and passwords and OnDemand userids and passwords.

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

Note: If you must use the GET method, then you can encrypt the parameters and values by specifying the ENCRYPTURL parameter in the ARSWWW.INI file. See "ENCRYPTURL" on page 33 for more information.

Chapter 2. Installing and configuring the HTTP server

This section defines the installation requirements and explains how to install the ODWEK software on the HTTP server and modify the ODWEK configuration file.

You must install the ODWEK software on an iSeries system that is running the current version of the IBM HTTP Server. In addition, if you plan to use the Java servlet, then make sure that you have the current version of the iSeries Web Application Server (WebSphere) up and running.

ODWEK can search for and retrieve documents from OnDemand servers that are running IBM Content Manager OnDemand for iSeriesVersion 5 Release 3 Common Server.

Installation requirements

ODWEK requires:

- The current version of the IBM HTTP Apache Server. In addition, if you plan to use the Java servlet, then make sure that you have the current version of the iSeries Web Application Server (WebSphere) up and running. The servers must run under OS/400 Version 5 Release 3.
- Appropriate media type for installation.
- Adequate disk space for installation files: approximately 30 MB on the HTTP server
- Adequate disk space for cache storage: by default, 10 MB on the HTTP server. See "CACHESIZE" on page 15 for more information.

Other requirements

ODWEK can *cache* (temporarily store) documents on the HTTP server. This can increase the speed with which previously viewed documents can be sent to users. To enable cache storage for documents, configure the CACHEDOCS parameter in the ARSWWW.INI file. See "CACHEDOCS" on page 14 for details.

By default, ODWEK caches data in the `/QIBM/UserData/OnDemand/WWW/CACHE` directory. You can specify a different cache directory by modifying the ARSWWW.INI file. See "CACHEDIR" on page 14 for details.

Make sure that the processes that run ODWEK programs can read from the directory that contains the programs and can write to the cache directory. When ODWEK is installed, all of the objects are secured by authorization list QONDADM and user profiles QTMHHTTP, QTMHHTP1 and QEJBSVR are added to the authorization list with *CHANGE authority. Also, the QRDARS400 authorization list must have QTMHHTTP, QTMHHTP1, and QEJBSVR on it with *USE authority.

If you plan to use the AFP2HTML applet, then you must obtain the AFP2WEB Transform service offering from IBM and install and configure it on the HTTP server. See your IBM representative for more information about the AFP2WEB Transform service offering. You must also provide configuration options for the

AFP documents and resources that you plan to process with the AFP2WEB Transform. See Appendix E, "AFP to HTML transform," on page 127 for more information about the configuration file.

If you plan to convert AFP documents stored in OnDemand to PDF documents that can be viewed with the Adobe Acrobat viewer, then you must obtain the AFP2PDF Transform service offering from IBM and install and configure it on the HTTP server. See your IBM representative for more information about the AFP2WEB Transform service offering. You must also provide configuration options for the AFP documents and resources that you plan to process with the AFP2PDF Transform. See Appendix F, "AFP to PDF transform," on page 131 for more information about the configuration file. To view converted documents, you must obtain the Adobe Acrobat viewer for the browsers used by your organization.

Installing on OS/400

Setting up ODWEK typically requires that you do the following:

1. Obtain a copy of the latest OnDemand README file. Print and then read the entire file before you begin.
2. To install ODWEK, follow the instructions in the book named *Software Installation* (SC41-5120). The licensed program number is 5722RD1 and the feature is 11.

   Note: The recommended way to install ODWEK is to use the Install licensed programs menu option from the Work with Licensed Programs menu (go licpgm). From the Install licensed programs screen, enter a 1 to Add an option, enter 5722RD1 for the Licensed Program and 11 for the Product Option or scroll through the list of Licensed Programs and Product Options until you find ODWEK and enter a 1 before it. **If you install OnDemand with any other method, errors can occur when you attempt to use it.**
3. IBM recommends that you order, load, and apply all PTFs available for OnDemand after successful installation of the licensed program. Refer to Informational APAR II13680 for a complete list of OnDemand Version 5 Release 3 PTFs. The informational APAR can be ordered electronically using the SNDPTFORD command, specifying II13680 for the PTF number. Be sure to read the PTF cover letters and follow any special instructions.

Your next step

Make sure that you have the current version of the IBM HTTP server up and running on the iSeries system. You will need to configure the HTTP server. See Appendix G, "HTTP server configuration files," on page 133 for an example of an HTTP server configuration file.

If you plan to use the Java servlet, make sure that you have the current version of the iSeries Web Application Server (WebSphere) up and running. You will need to configure WebSphere. For instructions, see the IBM WebSphere Application Server for AS/400® Documentation Center on the Web at www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/. Follow the links to Installation and Initial Configuration for the appropriate version of WebSphere.

After you have installed the ODWEK software, configured the HTTP server, and (optionally) configured WebSphere, you can now configure the ODWEK initialization file for your operating environment. See "Specifying the ARSWWW.INI file."

Specifying 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=/tmp/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@_gunnar]
HOST=gunnar
PORT=1446
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 37 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 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 can contain the following parameters:

PORT

The TCP/IP port number that OnDemand servers use to communicate with ODWEK. If you do not specify the PORT parameter, then the server uses the port number that is specified for OnDemand in the Service Table (WRKSRVTBLE). If you do not specify the PORT parameter and OnDemand is not listed in the Service Table, then the servers will attempt to use port number `1445`. To specify that the servers use the port number that is specified for OnDemand in the Service Table, specify `0` (zero).

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=0
```

PROTOCOL

The networking protocol that OnDemand servers use to communicate with ODWEK. You must specify `0` (zero) for TCP/IP.

You must specify this parameter once in the default section.

This parameter is optional. If you do not specify this parameter, 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 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 required.

This section can contain the following parameters:

HOST

The name of the 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@_gunnar]
HOST=gunnar
```

PORT

The TCP/IP port number that the OnDemand server uses to communicate with ODWEK. 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.

Example:

```
[@SRV@_gunnar]
PORT=0
```

This port number should match the port number specified in the ars.ini file for the instance.

PROTOCOL

The networking protocol that the OnDemand server uses to communicate with ODWEK. You must specify `0` (zero) for TCP/IP.

You can specify this parameter once in the server section.

This parameter is optional. If not specified, the value specified (or defaulted to) in the default server section is used.

Example:

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

[CONFIGURATION]

The CONFIGURATION section contains parameters that are used by ODWEK on the HTTP 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:

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=/QIBM/UserData/OnDemand/www/cache
```

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 AliasMatch:
   - If you specify a directory name, the directory must be relative to the `/QIBM/UserData/OnDemand/WWW` directory. For example, if you specify `appletdir=applets`, the applets must exist in the `/QIBM/UserData/OnDemand/WWW/APPLETS` directory.

- If you specify an AliasMatch, it must be defined in the HTTP server configuration file. For example, if you specify `appletdir=/applets/`, the HTTP server configuration file must have an AliasMatch for `/applets/`. The replacement file path of the AliasMatch rule must be set to the full path name of the directory on the server. For example:

  ```
  AliasMatch ^/applets/com/ibm/edmslod/(.*)$ /QIBM/UserData/OnDemand/www/applets/$1

  AliasMatch ^/applets/(.*)$ /QIBM/UserData/OnDemand/www/applets/$1
  ```

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

Use to specify the directory on the HTTP server in which ODWEK temporarily stores (*caches*) documents (see "CACHEDOCS"). By default, ODWEK caches documents in the `/QIBM/UserData/OnDemand/WWW/CACHE` directory.

Note: 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.

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

This parameter is optional.

Example:

```
[CONFIGURATION]
CACHEDIR=/QIBM/UserData/OnDemand/WWW/CACHE
```

CACHEDOCS

Determines whether ODWEK temporarily stores (*caches*) documents on the HTTP 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 15).

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. However, in general, most customers should always configure cache storage for documents.

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 the oldest items in 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 the oldest items in cache storage when a threshold is reached (see "CACHEMAXTHRESHOLD").

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 HTTP server. Specify the value in megabytes. The default value is `10` (ten megabytes).

Note: To enable cache storage for documents, see "CACHEDOCS" on page 14.

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 HTTP server. Generally, this can increase the speed with which ODWEK sends previously viewed documents to users.

Example:

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

CACHEUSERIDS

Specifies a comma separated list of 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 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 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 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 in cache storage. The users will not be able to access folders created after they log on to an 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 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=user1,user2,user3
```

CODEPAGE

Identifies the code page of the OnDemand database. By default, ODWEK uses the code page of the HTTP server.

This parameter has a global scope, and you specify it only once in the CONFIGURATION section. When using the Logon API, you can override the specified code page with the `_codepage` parameter.

This parameter is optional. However, if the HTTP server is running in a different code page than the database, then you must specify the CODEPAGE parameter.

Example:

```
[CONFIGURATION]
CODEPAGE=37
```

DOCSIZE

When retrieving documents from the 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.

Notes:

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

   ```
   imagedir=pictures
   ```

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

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

2. You can specify a directory name or an AliasMatch:
   - If you specify a directory name, then the directory must be relative to the `/QIBM/UserData/OnDemand/WWW` directory. For example, if you specify `imagedir=pictures`, then the images must exist in the `/QIBM/UserData/OnDemand/WWW/PICTURES` directory.
   - If you specify an AliasMatch rule, then it must be defined in the HTTP server configuration file. For example, if you specify `imagedir=/pictures/`, then the HTTP server configuration file must have an AliasMatch for `/pictures/`. The AliasMatch rule must be set to the full path name of the directory on the server. For example:

     ```
     AliasMatch   ^/images/(.*)$   /QIBM/UserData/OnDemand/WWW/PICTURES/$1
     ```

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=pictures
```

LANGUAGE

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

Value	Territory
ARA	Egypt
CHS	China
CHT	Taiwan
DAN	Denmark
DEU	Germany

Value	**Territory**
ENU	U.S.A. / English
ESP	Spain
FIN	Finland
FRA	France
FRC	Canada
ITA	Italy
JPN	Japan
KOR	Korea
NLD	Netherlands
NOR	Norway
PTB	Brazil
SVE	Sweden

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

This parameter is optional.

Example:

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

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 you do not specify the TEMPDIR parameter, ODWEK will store temporary files in the run-time directory. If you are using the CGI program, the run-time directory is the directory in which the CGI program was installed. If you are using the servlet, the run-time directory is the directory that contains the servlet: for some installations, the run time directory is the location of the java.exe file; for others, the run time directory is the servlets directory, however, the exact location is dependent on the Java application server.

Example:

```
[CONFIGURATION]
TEMPDIR=/QIBM/UserData/OnDemand/WWW/TMP
```

Note: Verify the permissions of the directory that you specify. The processes that run ODWEK programs must write to and read from the temporary directory.

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 `/QIBM/UserData/OnDemand/WWW/SAMPLES` directory.

Note: 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=/QIBM/UserData/OnDemand/WWW/SAMPLES
```

[SECURITY]

The SECURITY section contains the security parameters that are used by ODWEK on the HTTP 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 that ODWEK should use the Inactivity Time Out parameter from the OnDemand server. The Inactivity Time Out parameter determines when a server can terminate a session with an inactive user. To specify that ODWEK should use the Inactivity Time Out parameter, set the REPORTSERVERTIMEOUT parameter to 1 (one).

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 the REPORTSERVERTIMEOUT parameter, then ODWEK will not use the Inactivity Time Out parameter, meaning that ODWEK will not terminate a session with an inactive user. For more information about the Inactivity Time Out parameter, see the online help for the administrative client.

Example:

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

SERVERACCESS

Specifies a comma separated list of the OnDemand servers that ODWEK can access. If you specify the SERVERACCESS parameter, then the clients that use ODWEK and the programs that use the APIs are permitted to access only those servers that you specify. You can specify the TCP/IP address, host name alias, 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.

Example:

```
[SECURITY]
SERVERACCESS=dave,gunnar
```

[AFP2HTML]

The AFP2HTML section contains the parameters that are used by the AFP2WEB Transform. The AFP2WEB Transform 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 AFP2WEB Transform service offering from IBM and install and configure it on the server. See your IBM representative for more information about the AFP2WEB Transform service offering. Someone in your organization must also provide configuration options for the AFP2WEB Transform. See Appendix E, “AFP to HTML transform,” on page 127 for more information about the configuration file.
2. To convert documents with the AFP2WEB Transform, you must specify the `AFPVIEWING=HTML` parameter in the DEFAULT BROWSER section (or other browser sections). See “AFPVIEWING” on page 31 for details. (If you plan to use the Retrieve Document API, then you should specify the `_afp=HTML` parameter. See “Retrieve Document” on page 75 for details.)
3. By default, ODWEK uses the AFP2HTML applet to view converted documents. If a converted document was stored in 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 AFP2WEB Transform to convert AFP documents and resources into HTML data, fonts, and images that can be viewed with the AFP2HTML applet. Appendix E, “AFP to HTML transform,” on page 127 shows the sample configuration file provided with OnDemand. See the AFP2WEB 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 AFP2HTML section.

This parameter is optional.

Example:

```
[AFP2HTML]
CONFIGFILE=afp2html.ini
```

INSTALLDIR

The directory that contains the AFP2WEB Transform programs, configuration files, and mapping files. Specify the full path name of the directory on the HTTP 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 AFP2HTML section.

This parameter is optional.

Example:

```
[AFP2HTML]
INSTALLDIR=/QIBM/UserData/OnDemand/www/bin
```

USEEXECUTABLE

Determines whether ODWEK starts the AFP2WEB Transform by using the shared library (DLL) or the executable (EXE).

Important: ODWEK on the iSeries must use the executable. Therefore, this parameter must always be set to 1 (one).

This parameter has a global scope, and you specify it only once in the AFP2HTML section.

This parameter is optional.

Example:

```
[AFP2HTML]
USEEXECUTABLE=1
```

[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 HTTP server. See your IBM representative for more information about the AFP2PDF Transform service offering. Someone in your organization must also provide configuration options for the AFP2PDF Transform. See Appendix F, "AFP to PDF transform," on page 131 for more information about the configuration file.
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 31 for details. (If you plan to use the Retrieve Document API, then you should specify the `_afp=PDF` parameter. See "Retrieve Document" on page 75 for details.)
3. By default, ODWEK uses the Adobe Acrobat viewer to view converted documents. You must obtain the viewer for the browsers 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. Appendix F, "AFP to PDF transform," on page 131 shows the sample configuration file provided with OnDemand. 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 HTTP 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=/QIBM/UserData/OnDemand/www/bin
```

USEEXECUTABLE

Determines whether ODWEK starts the AFP2WEB Transform by using the shared library (DLL) or the executable (EXE).

Important: ODWEK on the iSeries must use the executable. Therefore, this parameter must always be set 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 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 OnDemand server. The `parameter` identifies the data type of the document in OnDemand. (This is the data type that is assigned to the 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 37), the MIMETYPES section contains a parameter for each of the standard data types supported by OnDemand (AFP, BMP, EMAIL, GIF, JFIF, LINE, PCX, PDF, and TIFF).
3. In addition to the standard data types, 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 OnDemand application on the View Information page. The file extension must also be registered with the operating system on the client. For example, suppose you define an application to store Lotus® WordPro documents in OnDemand. 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/vnd.lotus-wordpro
   ```

Then, when a user retrieves a document from the application, ODWEK sets the MIME content type to `application/vnd.lotus-wordpro` and the system starts Lotus WordPro to open the document. For Netscape, the MIME content type must be defined in Preferences->Navigator->Applications.

Table 1 lists the MIME content types for several PC applications:

Table 1. MIME content types for several PC applications

Application	MIME content types

Lotus Applications	WK1=application/vnd.lotus-1-2-3
	WK3=application/vnd.lotus-1-2-3
	WK4=application/vnd.lotus-1-2-3
	123=application/vnd.lotus-1-2-3
	APR-application/vnd.lotus-approach
	VEW=application/vnd.lotus-approach
	LWP=application/vnd.lotus-wordpro
	SAM=application/vnd.lotus-wordpro
	MWP=application/vnd.lotus-wordpro
	SMM=application/vnd.lotus-wordpro
	PRE=application/vnd.lotus-freelance
	PRZ=application/vnd.lotus-freelance

Microsoft Applications	DOC=application/msword
	XLS=application/vnd.ms-excel
	PPS=application/vnd.ms-powerpoint
	PPT=application/vnd.ms-powerpoint
	MPD=application/vnd.ms-project
	MPP=application/vnd.ms-project
	MPT=application/vnd.ms-project
	MPD=application/vnd.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 31 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 client operating system.

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 32 for more information about processing EMAIL documents before sending them to the client.

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 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 client operating system.

Example:

```
[MIMETYPES]
JFIF=image/IBM-OnDemand
```

LINE

The MIME content type for line data documents. See "LINEVIEWING" on page 33 for more information about processing line data documents before sending them to the client.

This is used when LINEVIEWING=NATIVE is specified in the [DEFAULT BROWSER] section. If you extract and uncompress line data documents from 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 client operating system.

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 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 client operating system.

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). 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 H, “No HTML output,” on page 135 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 specific browser section as detailed in “[browser]” on
| page 36. (The parameters specified in a browser section override those from the
| DEFAULT BROWSER section.)

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.

Note: If you use the Microsoft Internet Explorer browser, then IBM recommends that you specify ADDEXTENSION=1 so that the file extension is added to the URL.

This parameter has a global scope, and you specify it 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. You can set the `Annotations Flags in document database table field` on the Database Information dialog box, from the General page in the OnDemand application group definitions. (Click Advanced to open the Database Information dialog box.)

This parameter has a global scope, and you specify it 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.

Note: Users are permitted or denied the ability to add annotations to documents based on the Annotation permissions in the OnDemand application group.

This parameter has a global scope, and you specify it 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 OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the client. For example, some customers convert AFP documents to HTML with the AFP2WEB Transform 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 client.

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 AFP2WEB Transform.
NATIVE	ODWEK extracts and uncompresses AFP documents and their resources from 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 23 for details.
PDF	ODWEK converts AFP documents to PDF documents with the AFP2WEB Transform. **Note:** 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 23 for details.
PLUGIN	ODWEK does not convert AFP documents (the default).

This parameter has a global scope, and you specify it 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 client 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: Do not enable automatic document retrieval if you plan to use the Microsoft Internet Explorer browser. IBM suggests that you specify AUTODOCRETRIEVAL=0 in any browser sections that you define for Internet Explorer.

This parameter has a global scope, and you specify it 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 OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the client.

You can set this parameter to one of the following values:

NATIVE
: ODWEK extracts and uncompresses EMAIL documents 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 23 for details.

HTML
: ODWEK converts EMAIL documents to HTML documents. This is the default value.

This parameter has a global scope, and you specify it 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, and you specify it 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, and you specify it 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 HTTP server, then you can encrypt these values by specifying ENCRYPTURL=1. See "Server and data security" on page 6 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 36). 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 OnDemand server, the value of this parameter determines what action, if any, ODWEK takes before sending the document to the client.

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 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 23 for details.

This parameter has a global scope, and you specify it 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 the first value specified to determine the number of items to return to the document list:
   a. 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.
   b. 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.
   c. The value of the MAXHITS parameter, if specified.
   d. If none of the above are specified, ODWEK returns a maximum of 200 items to the document list.

This parameter has a global scope, and you specify it 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 suggests that you specify NOLINKS=0 in any browser sections that you define for Internet Explorer.

This parameter has a global scope, and you specify it 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 52.

ODApplet.jre.path.NN
See "Java line data viewer" on page 52.

ODApplet.jre.version
See "Java line data viewer" on page 52.

ODApplet.version
See "Java line data viewer" on page 52.

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 OnDemand server.
2. Users are permitted or denied the ability to print documents based on the Print permissions in the OnDemand application group.

This parameter has a global scope, and you specify it 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 H, "No HTML output," on page 135), determines whether the storage location of the document will appear in the output. See "Document Hit List" on page 137 for details. The default value is `0` (zero) and means that the storage location will not appear in the output.

Note: To display the storage location, you must also set the `Display Document Location` property in the OnDemand folder.

This parameter has a global scope, and you specify it 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.

Note: Users are permitted or denied the ability to view annotations to documents based on the Annotation permissions in the OnDemand application group.

This parameter has a global scope, and you specify it 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. 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 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. If no browser section is found, ODWEK then searches the ARSWWW.INI file for one of the following sections:

   ```
   [browser version(major.minor)/platform]

   [browser version(major.minor)]

   [browser version(major)]

   [browser]

   [DEFAULT BROWSER]
   ```

   ODWEK uses the options from the first section that matches the value.
2. For the `browser`, you can specify `IE` or `Netscape`.
3. For the `platform`, you can specify `WinNT` or `Unix`. |

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 29.

Examples:

```
[IE 5]
AUTODOCRETRIEVAL=0
NOLINKS=0

[Netscape 4.7]
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 must be the first executable statement in the arswww.ini file.

The DEBUG 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:

LOG

Enables ODWEK to write messages and other program information to a log file. (The log file is named ARSWWW.LOG.)

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. Specify a value of 1 (one) to log messages.

LOGDIR

Determines the directory in which ODWEK writes the ARSWWW.LOG file, if logging is enabled using the LOG 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 `/QIBM/UserData/OnDemand/WWW/LOG` directory.

| Example:

```
|     [DEBUG]
|     LOGDIR=/QIBM/UserData/OnDemand/WWW/LOG
|     LOG=1
```

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, Web browsers, and viewers.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Server Configuration   ;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;[DEBUG]
;log=1
;logdir=/QIBM/UserData/OnDemand/www/logs

[@SRV@_<host alias>];
HOST=<host name>;
PORT= ;
PROTOCOL= ;

[@SRV@_myiSeries.mycompany.com]
HOST=myiSeries.mycompany.com
PORT=1450
PROTOCOL=0

[CONFIGURATION]
CodePage=37
Language=ENU
TemplateDir=/QIBM/UserData/OnDemand/www/SAMPLES
ImageDir=/IMAGES/
AppletDir=/applets/
CacheDir=/QIBM/UserData/OnDemand/www/
CacheSize=0
CacheMinThreshold=0
CacheMaxThreshold=0
CacheDocs=0
CacheUserIDs=web,demo,mstephens

[SECURITY]
SERVERACCESS=

[AFP2HTML]
InstallDir=/QIBM/UserData/OnDemand/www/bin
ConfigFile=/QIBM/UserData/OnDemand/www/bin/afp2html.ini
UseExecutable=1

[AFP2PDF]
InstallDir=/QIBM/UserData/OnDemand/www/bin
ConfigFile=/QIBM/UserData/OnDemand/www/bin/afp2pdf.ini
UseExecutable=1

[MIMETYPES]
BMP=image/IBM_OnDemand
GIF=image/IBM_OnDemand
JFIF=image/IBM_OnDemand
PCX=image/IBM_OnDemand
TIFF=image/IBM_OnDemand
PNG=image/IBM_OnDemand
PDF=application/pdf
AFP=application/afp
LINE=application/line
EMAIL=text/html
META=application/unknown

[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=applet
;EmailViewing=[html,native]
EmailViewing=html
;MetaViewing=[xenos,native]
MetaViewing=native
NoLinks=1
ViewNotes=1
AddNotes=1
ServerPrint=1
ServerPrinters=P
AutoDocRetrieval=1
MaxHits=200
ShowDocLocation=1

[IE]
NoLinks=0
AddExtension=1
AddFieldsToDocid=0
```

Your next step

After you have installed the ODWEK software and configured the ARSWWW.INI file, you should now configure the sample applications. See Chapter 3, “Configuring the sample applications,” on page 41.

Chapter 3. Configuring the sample applications

This chapter explains how to customize the sample applications that are provided with ODWEK:

- LOGON.HTM. This application supports users that are permitted to access several folders. Each user is defined to the OnDemand server. After logging on to the server, ODWEK shows the user the list of folders that the user is permitted to open.
- 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" contains instructions for customizing this application.

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 one of the sample applications. Use the instructions as a guide for customizing other applications that you may need.

Note: 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 42 for important information about modifying this file.

LOGON.HTM

1. Copy the `logon.htm` file from the installation directory (`/QIBM/ProdData/OnDemand/www.samples`) to the document root directory of the HTTP server.
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 `/QIBM/UserData/OnDemand/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 HTTP 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 19 for details.

Your next step

After you have installed the ODWEK software, configured the ARSWWW.INI file, and configured the sample applications, you should now install the Web viewers on the user workstations. See Chapter 4, “Installing the Web viewers,” on page 45.

Chapter 4. Installing the Web viewers

Overview

IBM provides viewers for the standard types of documents that can be retrieved from OnDemand. The installation requirements vary, depending on the viewers that the people in your organization need to use.

- To view line data documents, IBM recommends that you use the Line Data applet. The Line Data applet is stored on the HTTP server. After you enable the use of the Line Data applet, 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 OnDemand AFP Web Viewer, the AFP2HTML applet, or the Adobe Acrobat viewer.
  - To view AFP documents with the IBM OnDemand AFP Web Viewer, users must install it on their workstations.
  - To view AFP documents with the AFP2HTML applet, an administrator must install and configure the AFP2WEB Transform on the HTTP server and configure the ARSWWW.INI file. The AFP2HTML applet is stored on the HTTP server. After an administrator enables the use of the AFP2HTML applet, it is automatically loaded into memory on the workstation when the user selects to view an AFP document.
  - To view AFP documents with the Adobe Acrobat viewer, an administrator must install and configure the AFP2PDF Transform on the HTTP server 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 workstation.

  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 OnDemand Image Web Viewer on their workstations; 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.

Notes:

1. To view other types of data, you may need to install other viewers. For example, to view PDF documents that are retrieved from the OnDemand server, IBM recommends that you obtain and install the Adobe Acrobat viewer for the browsers used in your organization.
2. The `nppdf32.dll` file is required in the browser plugin directory to view PDF documents. For Internet Explorer, it should be in the `\Program Files\Internet Explorer\PLUGINS` directory. For Netscape, it should be in the `\Program Files\Netscape\Communicator\Program\Plugins` directory. If the file is not in the browser directory, you will need to reinstall the Adobe software.

Requirements

The viewers that are provided by IBM require Netscape Navigator 4.7 or later or Microsoft Internet Explorer 5.5 or later.

ODWEK requires the ability to write cookie data on the PC. Make sure that your users configure their browsers to accept cookies.

IBM provides two versions of the Java line data viewer in the applets directory:

ODLineDataViewer.jar is the old Java line data viewer, which requires Java support in the browser. Java support is most likely provided by a Java Virtual Machine (JVM). **Note:** For Microsoft Internet Explorer, the user may need to install the JVM by using the Custom installation option of the browser.

ODLineDataViewer2.jar is the new Java line data viewer, which requires Version 1.4.1 or later of the Java plugin. The new Java line data viewer does not use the Java support in the browser. The user must install the Java plugin on the PC to use the new Java line data viewer. See “Java line data viewer” on page 52 for important configuration information.

The default product installation will use the old Java line data viewer.

The browser must run under Windows® 2000, Windows XP, or Windows Server 2003 and requires the following hardware and software:

- Physical connection to the network, such as a Token Ring or Ethernet network adapter
- Windows TCP/IP support
- A minimum of 256 MB of RAM
- Pentium® or Pentium compatible 800 MHz or faster processor
- A super VGA display and adapter with a minimum resolution of 800 x 600
- A minimum of 20 MB of free disk space to view documents
- Approximately 3 MB on each workstation that needs the IBM OnDemand AFP Web Viewer and 2 MB on each workstation that needs the IBM OnDemand Image Web Viewer.

Installation

Note: 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 47 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 system 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 `/QIBM/ProdData/OnDemand/www/plugins` directory:

- afpplgus.exe - IBM OnDemand AFP Web Viewer - English only
- afpplgin.exe - IBM OnDemand AFP Web Viewer - All languages including DBCS support
- afpplgin.zip - IBM OnDemand AFP Web Viewer - Zip format for all languages include DBCS support

- imgplgin.exe - IBM 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 workstation. The Image Web Viewer requires approximately 2 MB of space on the workstation. Remind your users to restart their browser if it is active during the installation process.

Note: The installation program will install the viewers as either plug-ins or ActiveX controls. If Internet Explorer is installed on the workstation, then the installation program will install the ActiveX controls; if Netscape is installed on the workstation, then the installation program will install the plug-ins. If you have both Internet Explorer and Netscape installed on the workstation, then the installation program will install the ActiveX controls for Internet Explorer and the plug-ins for Netscape.

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.

Note: 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 FLDPORT2.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 HTTP 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 a 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" and "Storing user-defined files" on page 49 for details.
  5. If distributing user-defined Adobe Type 1 font files, then create a font configuration file. See "Configuring font files" on page 49 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 50 for details.
  7. Replace the EXE file provided by IBM for a Standard Install with the self-extracting EXE file that you built.
  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 client installation directory. For example, you could name the main client installation directory \ONDEMAND\AFP32.

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

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

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

Note: 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" on page 48 for more information.
2. Create a Type 1 font configuration file. The following information describes how to create the Type 1 font configuration file.

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 48 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:

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

```
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.

   Note: 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.

Note: 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 plug-in. 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 code resides.

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.

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.

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

Image Web viewer

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

Note: 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

IBM now provides an enhanced Java line data viewer. Functional improvements include enhanced printing functions, such as printing the entire width of the page. The graphical user interface is based on the Swing library.

IBM now provides two versions of the Java line data viewer in the applets directory:

ODLineDataViewer.jar is the old Java line data viewer, which requires Version 1.1.8 or later of the Java plugin.

ODLineDataViewer2.jar is the new Java line data viewer, which requires Version 1.4.1 or later of the Java plugin.

Customers can use the new Java line data viewer or the old Java line data viewer. The choice is specified by setting parameters in the [DEFAULT BROWSER] section of the ARSWWW.INI file. In addition, the new Java line data viewer requires Version 1.4.1 or later of the Java plugin for the browser. Additional parameters in the ARSWWW.INI file determine the version number and the location of the Java plugin installation file for users that do not have the required version of the Java plugin installed on their workstations.

Table 2 on page 53 describes new parameters in the ARSWWW.INI file that support the Java line data viewer.

Table 2. Parameters in ARSWWW.INI File for Java Line Data Viewer

Parameter	Value	Comments
ODApplet.version	1	Specifies to invoke the old Java line data viewer. If specified, ignore the remaining parameters. **Note:** This is the default value. Also, if this parameter is omitted, ODWEK will use the old Java line data viewer.
	2	Specifies to invoke the new Java line data viewer (enhanced version). If specified, use the following three parameters.
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. **Note:** The user may need to restart the browser after installing the plugin.
	`<location>`	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` **Note:** 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 2. Parameters in ARSWWW.INI File for Java Line Data Viewer (continued)

Parameter	Value	Comments
ODApplet.jre.path.NN	http://java.sun.com/j2se/1.4.1/download.html	For Netscape. Specifies to open the JRE/J2SE Download page for choosing the Java plugin to install. The user would follow the link to download the Java plugin installation file for their platform. After downloading the Java plugin installation file, the user must install the plugin on the workstation. The user may need to restart the browser after installing the plugin.
	`<location>`	Specifies the location of the plugin file(s) within a company's intranet. The location must include a valid browser protocol, such as http, file, or ftp. For example: `http://webServer/tmp/ondemand/java/plugins` **Note:** An administrator must download the plugin file(s) and store them in the specified location. You cannot specify a path to a specific file because it is not known on which operating system that Netscape is running. Also, the specified format allows the administrator to download the plugin for other platforms, if desired. The user must install the Java plugin on the workstation. The user may need to restart the browser after installing the plugin.

Table 2. Parameters in ARSWWW.INI File for Java Line Data Viewer (continued)

Parameter	Value	Comments
ODApplet.jre.version	`<version>`	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 example shows how to configure the ARSWWW.INI file to support the old Java line data viewer.

```
[DEFAULT BROWSER]
ODApplet.version=1
```

Notes:

1. If you omit the ODApplet.version parameter from the ARSWWW.INI file, ODWEK will use the old Java line data viewer.
2. The ODApplet 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.)

The following shows an example of how to configure the ARSWWW.INI file to support the new Java line data viewer (enhanced version) and 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. For Netscape, an administrator has stored copies of the Java plugin installation files for the various platforms in the specified location on a local Web server so that users do not have to go to the java.sun.com JRE/J2SE Download page. **Note:** 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.version=2
ODApplet.jre.path.IE=http://java.sun.com/getjava/installer.html
ODApplet.jre.path.NN=http://localWebServer/java/plugins
ODApplet.jre.version=1.4
```

Your next step

After you have installed the ODWEK software, configured the ARSWWW.INI file, configured the sample applications, and installed the Web viewers, you can now begin using ODWEK.

Appendix A. CGI API reference

This chapter 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.

Note: Parameter values are standard text. It is 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 the Add Annotation permission in the OnDemand application group. (The Access permission also lets users add annotations.)

Parameters

Table 3. 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 the Annotation Add permission for each application group that contains documents to be annotated. (The Application Group Access permission lets users add annotations.)
_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 with View Annotation permission for 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.

Table 3. Add Annotation 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, 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.
_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 H, "No HTML output," on page 135 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 port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the HTTP 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=od400&_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 4. 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 H, "No HTML output," on page 135 for details about the delimited ASCII output.
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the HTTP 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 4. 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 HTTP 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=od400&_user=web&_password=web
&_newpassword=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 a document, ODWEK retrieves the document from the server and displays it in the browser window using the appropriate viewer.

Parameters

Table 5. 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 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 5. 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 the first value specified to determine the number of items to return to the document list: 1. The value of the Maximum Hits field (specified on the folder Permissions page). This value overrides all other values. 2. The value of the **_max_hits** parameter, if specified. This value overrides the `MAXHITS` parameter from the ARSWWW.INI file. 3. The value of the `MAXHITS` parameter, if specified. 4. 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 the *IBM Content Manager OnDemand for iSeries V5R1 Common Server Administration Guide*, SC27–1161 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), ODWEK generates delimited ASCII output. See Appendix H, "No HTML output," on page 135 for details about the delimited ASCII output.
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.

Table 5. 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 HTTP 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 command 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 HTTP server configuration file. By default, ODWEK retrieves the CGI program from the /QIBM/Proddata/OnDemand/www/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=od400&_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 6. 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 H, “No HTML output,” on page 135 for details about the delimited ASCII output.
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.

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 7. 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 the *IBM Content Manager OnDemand for iSeries V5R1 Common Server Administration Guide*, SC27–1161 for details about date formats supported by OnDemand.

Table 7. 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 H, "No HTML output," on page 135 for details about the delimited ASCII output.
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the HTTP 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 HTTP 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=od400&_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 be given the Print Document permission in the OnDemand application group. (The Access permission also lets users print documents.) At least one server printer must be defined on the specified OnDemand server.

Parameters

Table 8. Print Document function

Name=Value	Purpose
_function=printdocs	Print documents.
_server=*value*	The name of the OnDemand server.
_user=*value*	The OnDemand userid. The user must be given the Document Print permission for each application group that contains documents to be printed. (The Application Group Access permission lets users print documents.)
_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 8. 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 H, “No HTML output,” on page 135 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. **Note:** If the number of document identifiers exceeds 200, then you must specify the **_max_hits** parameter.
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the HTTP 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 8. 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. **Note:** 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=printdocs
&_server=od400&_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, viewer, or other program that is associated with the document type.

Parameters

Table 9. 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 H, “No HTML output,” on page 135 for details about the delimited ASCII output.
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.

Table 9. 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 HTTP 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 HTTP 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 client. For example, some customers convert AFP documents to HTML with the AFP2WEB Transform 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 client. The *value* can be: **ASCII** ODWEK converts the AFP document to ASCII text. **HTML** ODWEK converts the AFP document to HTML with the AFP2WEB Transform. **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 23 for more information). **PDF** ODWEK converts the AFP document to PDF with the AFP2WEB Transform. **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 client. 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 23 for more information). **HTML** ODWEK converts the EMAIL document to HTML.

Table 9. 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 client. 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 23 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=od400&_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 10. 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 the *IBM Content Manager OnDemand for iSeries V5R1 Common Server Administration Guide*, SC27–1161 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 H, “No HTML output,” on page 135 for details about the delimited ASCII output.

Table 10. Search Criteria function (continued)

Name=Value	Purpose
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the HTTP 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 HTTP 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=od400&_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 11. 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 Update document permission for 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 H, “No HTML output,” on page 135 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 11. Update Document function (continued)

Name=Value	Purpose
_port=*value*	The port number for the OnDemand server. The default value, 0 (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number 1445. Any value that you specify overrides the value of the PORT parameter in the ARSWWW.INI file.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the HTTP 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=od400&_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 be given the View Annotation permission in the OnDemand application group. (The Access permission also lets users view annotations.)

Parameters

Table 12. 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 given the Annotation View permission for each application group that contains annotations to be viewed. (The Application Group Access permission lets users view annotations.)
_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 H, “No HTML output,” on page 135 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.

Table 12. View Annotations function (continued)

Name=Value	Purpose
_port=*value*	The port number for the OnDemand server. The default value, `0` (zero), means that the server uses the port number that is specified in the Service Table (WRKSRVTBLE). If there is no port number specified in the Service Table, then OnDemand attempts to use port number `1445`. Any value that you specify overrides the value of the `PORT` parameter in the ARSWWW.INI file.
_codepage=*value*	The code page of the OnDemand database. The default code page is the code page of the HTTP 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=od400&_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. Java servlet reference

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, "CGI API reference," on page 57 for a reference of the functions, descriptions, and parameters.

Appendix C. Java API reference

The documentation for the Java API is provided in HTML 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 `/QIBM/ProdData/OnDemand/www/` directory. 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.

Appendix D. Java API programming guide

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. **Note:** See Chapter 1, "Overview," on page 1 for limitations when accessing an OnDemand for OS/390 Version 2 server.
- 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.ODCallback
: This class is used with all methods in which the server operation returns data while processing.

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 may 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.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.

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

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.

To run the Java interpreter on an ODWEK application:

1. Copy the `arswww.ini` file to a user-defined run time directory.
2. Copy the shared library to the directory in which you copied the `arswww.ini` file:

Table 13. Shared Library Filename

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

3. For Windows systems, copy these files to the directory in which you copied the `arswww.ini` file:

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

4. Specify the name of the user-defined directory when you run the Java interpreter on the application. See "Running an ODWEK application" on page 93 for an example.

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.

AIX

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

PATH Make sure your `PATH` contains `/usr/lpp/ars/www`

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

LD_LIBRARY_PATH
Make sure your `LD_LIBRARY_PATH` 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.

PATH Make sure your `PATH` contains `/opt/ondemand/www`

LIBPATH Make sure your `LIBPATH` contains `/opt/ondemand/www`

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.

Linux

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

PATH Make sure your `PATH` contains `/opt/ondemand/www`

LIBPATH Make sure your `LIBPATH` contains `/opt/ondemand/www`

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.

PATH Make sure your `PATH` contains `/opt/ondemand/www`

LIBPATH Make sure your `LIBPATH` contains `/opt/ondemand/www`

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

The following parameters in the ARSWWW.INI file write trace information to the `arswww.log` file in the specified directory:

```
[DEBUG]
 LOG=1
 LOGDIR=/ars/www/log
```

Note: 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 system.

See Appendix J, "Problem determination tools," on page 141 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 an exception. Throwing an exception creates an exception object of `ODException` class or one of its subclasses.

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

When a `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( );
   odServer.initialize( argv[9], "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 an online reference. See Appendix C, "Java API reference," on page 87 for more information.

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, and `/tmp/ondemand/www` is the user-defined run time directory that contains a copy of the `arswww.ini` file. **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 90).

Connecting to 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. Appendix C, "Java API reference," on page 87 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

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( );
odServer.initialize( "c:\odwekdir", "Sample" );
System.out.println( "Logging on to " + "LIBSRVR1" + "..." );
odServer.logon( "LIBSRVR1", "ADMIN", "PASSWD" );
  .
```

```
  .
  .
odServer.logoff( );
odServer.terminate( );
```

See "Working with an OnDemand server" for the complete sample application from which this example was taken.

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( );
odServer.setServer( "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" for the complete sample application from which this example was taken.

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
- getLocalDir
- getPassword
- getPort
- getServerName
- getUserId
- setApplicationName
- setConnectType
- setLocalDir
- setPassword
- setPort
- setServer
- setUserId

This example uses these run-time parameters:

- Server name
- User Id
- Password
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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;
    String str;
    int j;

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

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "This testcase should:" );
      System.out.println( "  Use ODServer methods setServer, setUserId, and setPassword" );
      System.out.println( "    to prepare for logon" );
      System.out.println( "  Set the application name" );
      System.out.println( "  Display the" );
      System.out.println( "    Local Directory" );
      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( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[3], "TcServerMisc.java" );
      odServer.setServer( argv[0] );
      odServer.setUserId( argv[1] );
      odServer.setPassword( argv[2] );

      System.out.println( "Logging on to " + argv[0] + "..." );
      if ( argv.length == 4 )
        odServer.logon( );
      else
      {
        if ( argv.length == 5 )
        {
          odServer.setLocalDir( argv[4] );
          odServer.logon( odServer.getServerName( ),
                          odServer.getUserId( ),
                          odServer.getPassword( ),
                          ODConstant.CONNECT_TYPE_LOCAL,
                          0,
                          odServer.getLocalDir( ) );
        }
      }

      //----------
      // Test miscelaneous methods
      //----------
      System.out.println( "Setting application name to TcServerMisc.java..." );
      odServer.setApplicationName( "TcServerMisc.java" );

      System.out.println( "Local Dir: " + odServer.getLocalDir( ) );
      System.out.println( "Server Name: " + odServer.getServerName( ) );
      System.out.println( "User Id: " + odServer.getUserId( ) );
      System.out.println( "Password: " + odServer.getPassword( ) );
      System.out.println( "Connect Type: " + getConnectTypeName( odServer.getConnectType( ) ) );

      j = odServer.getPort( );
      System.out.println( "Setting port to " + j + "..." );
      odServer.setPort( j );
      System.out.println( "Port: " + j );

      if ( argv.length == 4 )
      {
        System.out.println( "Setting connect type to ODConstant.CONNECT_TYPE_TCPIP..." );
        odServer.setConnectType( ODConstant.CONNECT_TYPE_TCPIP );
      }
      else
      {
        System.out.println( "Setting connect type to ODConstant.CONNECT_TYPE_LOCAL..." );
        odServer.setConnectType( ODConstant.CONNECT_TYPE_LOCAL );
      }

      //----------
      // 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( );
    }
  }

  static String getConnectTypeName( char type )
  {
    String str;

    switch( type )
    {
      case ODConstant.CONNECT_TYPE_TCPIP:
        str = "TCPIP";
        break;
      case ODConstant.CONNECT_TYPE_LOCAL:
        str = "LOCAL";
        break;
      default:
        str = "*** Unknown connect type";
      break;
    }

    return str;
  }
}
```

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
- getApplGroups
- 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
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

Example of listing the application groups in a folder:

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

public class TcApplGrp
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    Object[] appl_grps;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcApplGrp <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" );
      System.out.println( "  Display the number of application groups" );
      System.out.println( "  Display the name of each application group" );
      System.out.println( "" );
      System.out.println( "---------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to the specified server
      //----------
      odServer = new ODServer( );
      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
      //----------
      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.getApplGroups( );
      for ( j = 0; j < appl_grps.length; j++ )
        System.out.println( "  " + appl_grps[j].toString( ) );

      //----------
      // 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
- setOperand
- setSearchValue
- setSearchValues

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

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
- Configuration directory (location of `arswww.ini` file)

Note: The number of hits may be restricted by the MAXHITS parameter in the `arswww.ini` file.

Example of searching a folder:

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

public class TcSearch
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODCriteria odCrit;
    ODHit odHit;
    Enumeration values_enum;
    Vector hits;
    String[] display_crit;
    String header, line1, line2, hit_value, useable_value;
    boolean mismatch_detected;
    int j, k, opr;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 9 )
    {
      System.out.println( "usage: java TcSearch <server> <userid> <password> <folder> <criteria> <opr> <value1> <value2> <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( "  Display the folder name and description" );
      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( "If arswww.ini is restricting the number of hits, there may be fewer" );
      System.out.println( "hits than displayed using the Windows Client." );
      System.out.println( "" );
      System.out.println( "---------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[8], "TcSearch.java" );
      System.out.println( "Logging on to " + argv[0] + "..." );
      odServer.logon( argv[0], argv[1], argv[2] );

      //----------
      // Open the specified folder and find the requested criteria
      //----------
      System.out.println( "Opening " + argv[3] + " folder..." );
      odFolder = odServer.openFolder( argv[3] );
      System.out.println( "Name='" + odFolder.getName( ) + "' Desc='" + odFolder.getDescription( ) + "'" );
      System.out.println( "Getting " + argv[4] + " criteria..." );
      odCrit = odFolder.getCriteria( argv[4] );

      //----------
      // Convert the operator parameter to the internal operator value and set
      // the criteria operator
      //----------
      System.out.println( "Setting operator to " + argv[5] + "..." );
      if ( argv[5].equals( "eq" ) )
        opr = ODConstant.OPEqual;
      else if ( argv[5].equals( "ne" ) )
        opr = ODConstant.OPNotEqual;
      else if ( argv[5].equals( "lt" ) )
        opr = ODConstant.OPLessThan;
      else if ( argv[5].equals( "le" ) )
        opr = ODConstant.OPLessThanEqual;
      else if ( argv[5].equals( "gt" ) )
        opr = ODConstant.OPGreaterThan;
      else if ( argv[5].equals( "ge" ) )
        opr = ODConstant.OPGreaterThanEqual;
      else if ( argv[5].equals( "in" ) )
        opr = ODConstant.OPIn;
      else if ( argv[5].equals( "ni" ) )
        opr = ODConstant.OPNotIn;
      else if ( argv[5].equals( "li" ) )
        opr = ODConstant.OPLike;
      else if ( argv[5].equals( "nl" ) )
        opr = ODConstant.OPNotLike;
      else if ( argv[5].equals( "be" ) )
        opr = ODConstant.OPBetween;
      else if ( argv[5].equals( "nb" ) )
        opr = ODConstant.OPNotBetween;
      else
        opr = -1;

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

      if ( opr == ODConstant.OPBetween || opr == ODConstant.OPNotBetween )
      {
        odCrit.setSearchValues( argv[6], argv[7] );
        System.out.println( "  " + odCrit.getName( ) + " " + getOperatorName( opr ) + " " + argv[6] + " and " + argv[7] );
      }
      else
      {
        odCrit.setSearchValue( argv[6] );
        System.out.println( "  " + odCrit.getName( ) + " " + getOperatorName( opr ) + " " + argv[6] );
      }

      //----------
      // Search the folder
      //----------
      System.out.println( "  Searching " + argv[3] + "..." );
      hits = odFolder.search( );
      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
- Configuration directory (location of `arswww.ini` file)

Example of searching a folder using an SQL string:

```
//*********************************************************************
//
// Testcase: TcSearchWithSQL
//
// This testcase will:
//   Logon to the specified server
//   Open the specified folder
//   Search the folder with the SQL string
//   Display the number of hits
//   Display the hitlist
//
// Tests the following methods:
//   ODServer
//     initialize
//     logon
//     openFolder
//     logoff
//     terminate
//   ODFolder
//     setApplGroupForSearchWithSQL
//     search
//     getDisplayOrder
//     close
//   ODHit
//     getDisplayValue
//
// Parameters:
//   1. Server name
//   2. User Id
//   3. Password
//   4. Folder name
//   5. Appl Group name
//   6. SQL string
//   7. Configuration directory (contains arswww.ini)
//
//*******************************************************************
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcSearchWithSQL
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    Enumeration values_enum;
    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> <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 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];
      directory  = argv[6];

      odServer = new ODServer( );
      odServer.initialize( directory, "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:
- setOperand
- setSearchValues

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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;
    ODFolder odFolder;
    ODCriteria odCrit;
    TestThread search_thread;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcCancelSearch <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( "  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( "Ensure that the folder contains many hits and that arswww.ini is" );
      System.out.println( "not overly restricting the number of hits which can be returned." );
      System.out.println( "" );
      System.out.println( "---------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[4], "TcCancelSearch.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] + "..." );
      odFolder = odServer.openFolder( argv[3] );
      odCrit = odFolder.getCriteria( "Date" );
      odCrit.setOperand( 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` and `ODCriteria.getValues` methods. Fixed search values are listed for any search fields that are defined as `FixedChoice` or `Segment` .

This example demonstrates these `ODCriteria` methods:
- setOperand
- getValidOperands
- getType
- getValues
- 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
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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( "    Default values (by ODCrit.getValues 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( );
      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.getOperand( );
        System.out.println( "  Default operator: " );
        System.out.println( "    " + getOperatorName( opr ) );

        //----------
        // Display valid operators
        //----------
        valid_oprs = odCrit.getValidOperands( );
        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.getValues( )
        //----------
        value_vec = odCrit.getValues( );
        System.out.println("  Default Value(s) (ODCrit.getValues method):");
        System.out.println( "    '" + value_vec.elementAt( 0 ) + "'" );
        System.out.println( "    '" + value_vec.elementAt( 1 ) + "'" );

        //----------
        // Display default value(s) using ODCrit.getSearchValues( )
        //----------
        search_values = odCrit.getSearchValues( );
        System.out.println("  Default Values (ODCrit.getSearchValues method):");
        for ( j = 0; j < search_values.length; j++ )
          System.out.println("    '" + search_values[j] + "'" );

        //----------
        // 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
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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;
    Enumeration folders_enum;
    String folder_name, 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> <config dir> [<local server dir>]" );
      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( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[3], "TcListFolders.java" );

      System.out.println( "Logging on to " + argv[0] + "..." );
      if ( argv.length == 4 )
        odServer.logon( argv[0], argv[1], argv[2] );
      else
        if ( argv.length == 5 )
          odServer.logon( argv[0], argv[1], argv[2], ODConstant.CONNECT_TYPE_LOCAL, 0, argv[4] );

      //----------
      // 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 );
      }

      //----------
      // 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 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
- Configuration directory (location of `arswww.ini` file)

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( );
  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 demonstrates three different methods of retrieving a document:
- ODServer
- ODFolder
- ODHit

This example logs on to the specified server, opens the specified folder, searches the folder using the default criteria, displays the number of hits, retrieves the data for the first hit using `ODHit.retrieve` , retrieves the data for the first hit using `ODServer.retrieve` , and retrieves the data for the first hit using `ODFolder.retrieve` . This example displays the length of data retrieved from each method, compares the lengths and data retrieved from each method, and displays the result of the comparisons.

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

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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;
    ODHit odHit;
    TcCallback callback;
    Vector hits;
    Vector hit_to_retrieve;
    byte[] data_from_hit;
    byte[] data_from_server;
    byte[] data_from_folder;
```

```
int j;

//----------
// If too few parameters, display syntax and get out
//----------
if ( argv.length < 5 )
{
  System.out.println( "usage: java TcRetrieve <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( "  Search the folder using the default criteria" );
  System.out.println( "  Display the number of hits" );
  System.out.println( "  Retrieve the data for the first hit using ODHit.retrieve" );
  System.out.println( "  Retrieve the data for the first hit using ODServer.retrieve" );
  System.out.println( "  Retrieve the data for the first hit using ODFolder.retrieve" );
  System.out.println( "  Display length of data retrieved from each method" );
  System.out.println( "  Compare the lengths and data retrieved from each method" );
  System.out.println( "  Display the result of the comparisons" );
  System.out.println( "" );
  System.out.println( "--------------------------------------------------" );
  System.out.println( "" );

  //----------
  // Logon to specified server
  //----------
  odServer = new ODServer( );
  odServer.initialize( argv[4], "TcRetrieve.java" );
  System.out.println( "Logging on to " + argv[0] + "..." );
  if ( argv.length == 5 )
    odServer.logon( argv[0], argv[1], argv[2] );
  else
    odServer.logon( argv[0], argv[1], argv[2], ODConstant.CONNECT_TYPE_LOCAL, 0, argv[5] );

  //----------
  // 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( ) );

  //----------
  // Do some retrieves and comparisons
  //----------
  if ( hits.size( ) > 0 )
  {
    odHit = (ODHit)hits.elementAt( 0 );
    System.out.println( "Retrieving data from first hit using ODHit.retrieve..." );
    data_from_hit = odHit.retrieve( "" );
    System.out.println( "Retrieving data from first hit using ODServer.retrieve..." );
    data_from_server = odServer.retrieve( odHit.getDocId( ), argv[3], "" );
    hit_to_retrieve = new Vector( );
    hit_to_retrieve.addElement( odHit );
    System.out.println( "Retrieving data from first hit using ODFolder.retrieve (uses callback method)..." );
    callback = new TcCallback( );
    odFolder.retrieve( hit_to_retrieve, callback );
    data_from_folder = callback.getData( );
    System.out.println( "Length of data from:" );
    System.out.println( "  ODHit.retrieve=" + data_from_hit.length );
    System.out.println( "  ODServer.retrieve=" + data_from_server.length );
    System.out.println( "  ODFolder.retrieve=" + data_from_folder.length );
    if ( data_from_hit.length == data_from_server.length )
    {
      for ( j = 0; j < data_from_hit.length; j++ )
      {
        if ( data_from_hit[j] != data_from_server[j] )
          break;
      }
      if ( j == data_from_hit.length )
      {
        System.out.println( "ODHit vs. ODServer: Length and content of data match" );
        if ( data_from_hit.length == data_from_folder.length )
        {
          for ( j = 0; j < data_from_folder.length; j++ )
          {
            if ( data_from_hit[j] != data_from_folder[j] )
              break;
          }
          if ( j == data_from_folder.length )
            System.out.println( "ODHit vs. ODFolder: Length and content of data matches" );
          else
          {
            System.out.println( "*** ODHit vs. ODFolder: Data mismatch at offset " + j );
            System.out.println( "  ODHit    data is " + data_from_hit[j] );
            System.out.println( "  ODFolder data is " + data_from_folder[j] );
          }
        }
        else
          System.out.println( "*** ODHit vs. ODFolder: Length mismatch" );
      }
```

```
          else
          {
            System.out.println( "*** ODHit vs. ODServer: Data mismatch at offset " + j );
            System.out.println( "      ODHit    data is " + data_from_hit[j] );
            System.out.println( "      ODServer data is " + data_from_server[j] );
          }
        }
        else
          System.out.println( "*** ODHit vs. ODServer: Length mismatch" );
      }
      else
        System.out.println( "There is no document to retrieve" );

      //----------
      // Cleanup
      //----------
      odFolder.close( );
      odServer.logoff( );
      odServer.terminate( );
      System.out.println( "" );
      System.out.println( "---------------------------------------------------" );
      System.out.println( "" );
      System.out.println( "Testcase completed - analyze the result of the comparisons" );
      System.out.println( "" );
      System.out.println( "If the arswww.ini file specifies 'native' for the data type, all" );
      System.out.println( "lengths and data should match; otherwise, differences are expected." );
      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
- printDocs
- close

This example uses these run-time parameters:
- Server name
- User Id
- Password
- Folder name
- Printer name
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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, hit_to_print;
    String [] printers;
    String printer_name;
    boolean match;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 6 )
    {
      System.out.println( "usage: java TcPrintHit <server> <userid> <password> <folder> <printer> <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( "  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( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[5], "TcPrintHit.java" );
      System.out.println( "Logging on to " + argv[0] + "..." );
      if ( argv.length == 6 )
        odServer.logon( argv[0], argv[1], argv[2] );
      else
        odServer.logon( argv[0], argv[1], argv[2], ODConstant.CONNECT_TYPE_LOCAL, 0, argv[6] );

      //----------
      // 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.printDocs( hit_to_print, printer_name );
          }
          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
- Configuration directory (location of arswww.ini file)
- (optional) Local server directory

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;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcListNotes <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( "  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( "  Display info for each note" );
      System.out.println( "" );
      System.out.println( "---------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[4], "TcListNotes.java" );
      System.out.println( "Logging on to " + argv[0] + "..." );
      if ( argv.length == 5 )
        odServer.logon( argv[0], argv[1], argv[2] );
      else
        odServer.logon( argv[0], argv[1], argv[2], ODConstant.CONNECT_TYPE_LOCAL, 0, argv[6] );

      //----------
      // 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 );
        notes = odHit.getNotes( );
        System.out.println("  There are " + notes.size( ) + " notes for the first hit" );
        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( ) );
        }
      }
      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( );
    }
  }
}
```

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
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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, notes;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 6 )
      System.out.println( "usage: java TcAddNote <server> <userid> <password> <folder> <note text> <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( "  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( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[5], "TcAddNote.java" );
      System.out.println( "Logging on to " + argv[0] + "..." );
      if ( argv.length == 6 )
        odServer.logon( argv[0], argv[1], argv[2] );
      else
        odServer.logon( argv[0], argv[1], argv[2], ODConstant.CONNECT_TYPE_LOCAL, 0, argv[6] );

      //----------
      // 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( );
        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( );
    }
  }
}
```

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:
- setOperand
- 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
- Configuration directory (location of `arswww.ini` file)

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, crit1, crit2, value1, value2, 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( "" );

      //----------
      // Logon to specified server
      //----------
      odServer = new ODServer( );
      odServer.initialize( argv[9], "TcUpdate.java" );
      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( crit1 );
      odCrit.setOperand( ODConstant.OPEqual );
      odCrit.setSearchValue( value1 );
      odCrit = odFolder.getCriteria( crit2 );
      odCrit.setOperand( 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:" );
               line = "      ";
               display_crit = odFolder.getDisplayOrder( );
               for( j = 0; j < display_crit.length; j++ )
                 line = line + display_crit[j] + " ";
        System.out.println( line );
        line = "      ";
        odHit = (ODHit)hits.elementAt( 0 );
        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.update( hash );
      }
      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
- Configuration directory (location of `arswww.ini` file)
- (optional) Local server directory

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> <config dir> [<local server dir>]" );
      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];
        odServer = new ODServer( );
        odServer.initialize( argv[4], "TcChangePassword.java" );

        //----------
        // Logon to the server using the original password
        //----------
        System.out.println( "Logging on to " + server + " using original password..." );
        if ( argv.length == 5 )
          odServer.logon( server, userid, original_password );
        else
          if ( argv.length == 6 )
            odServer.logon( server, userid, original_password, ODConstant.CONNECT_TYPE_LOCAL, 0, argv[5] );

        //----------
        // Change to the new password and logoff
        //----------
        System.out.println( "Changing to new password..." );
        odServer.changePassword( new_password );
        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..." );
        if ( argv.length == 5 )
          odServer.logon( server, userid, new_password );
        else
          if ( argv.length == 6 )
            odServer.logon( server, userid, new_password, ODConstant.CONNECT_TYPE_LOCAL, 0, argv[5] );

        //----------
        // Change back to the original password and logoff
        //----------
        System.out.println( "Changing back to original password..." );
        odServer.changePassword( original_password );
        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( );
      }
    }
  }
```

Appendix E. AFP to HTML transform

The AFP to HTML transform process converts AFP documents and resources into HTML documents. The AFP to HTML transform process requires the AFP2WEB Transform service offering from IBM Printing Systems Division. An administrator must install and configure the AFP2WEB Transform on the HTTP server. See your IBM representative for more information about the AFP2WEB Transform service offering. Someone in your organization must also specify configuration options for the AFP documents and resources that you plan to process with the AFP2WEB Transform. This section describes how to specify the configuration options.

Note: In this document, the name AFP2HTML.INI refers to the configuration file. To specify the file that contains the configuration options, see "CONFIGFILE" on page 21.

The AFP2HTML.INI file provides configuration options for the AFP2WEB Transform. You typically configure the AFP2HTML.INI file with options for specific AFP applications. However, you can also provide a set of default options. The AFP2WEB Transform uses the default options when converting documents and resources for AFP applications that are not identified in the AFP2HTML.INI file. To learn more details about the options and the conversion process, see the AFP2WEB Transform documentation.

The following topics provide additional information about the AFP2HTML.INI file:
- Format of the AFP2HTML.INI file
- Options for the AFP2WEB Transform
- Viewing converted documents

Note: To convert documents with the AFP2HTML applet, you must also specify the `AFPVIEWING=HTML` parameter in the DEFAULT BROWSER section (or other browser sections) of the ARSWWW.INI file. See "AFPVIEWING" on page 31 for details. (If you plan to use the Retrieve Document API, then you should specify the `_afp=HTML` parameter. See "Retrieve Document" on page 75 for details.) You must also specify the directory that contains the AFP2WEB Transform programs (see "CONFIGFILE" on page 21).

Format of the AFP2HTML.INI file

The following is an example of an AFP2HTML.INI file:

```
[CREDIT-CREDIT]
UseApplet=FALSE
ScaleFactor=1.0
CreateGIF=TRUE
SuppressFonts=FALSE
FontMapFile=creditFontMap.cfg
ImageMapFile=creditImageMap.cfg

[default]
ScaleFactor=1.0
CreateGIF=TRUE
SuppressFonts=FALSE
FontMapFile=fontmap.cfg
ImageMapFile=imagemap.cfg
```

The structure of the file is similar to a Windows INI file, and contains one stanza for each AFP application and one default stanza. The title line of the stanza identifies the application group and application. For example, the title line:

```
[CREDIT-CREDIT]
```

Identifies the CREDIT application group and the CREDIT application. Use the – (dash) character to separate the names in the title line. The names must match the application group and application names defined to the OnDemand server. If the application group contains more than one application, then create one stanza for each application.

The options in the `[default]` stanza are used by the AFP2WEB Transform to process documents for AFP applications that are not identified in the AFP2HTML.INI file. The defaults are also used if an AFP application stanza does not include one of the options.

The `UseApplet` option is a directive to ODWEK. It determines whether the AFP2HTML applet will be used to view the output from the AFP2WEB Transform. The default value is TRUE. If you specify FALSE, (the AFP2HTML applet is not used to view the output), then the output is formatted and displayed by the Web browser.

The remaining five options are directives to the AFP2WEB Transform. "Options for the AFP2WEB Transform" briefly describes how they are used by the AFP2WEB Transform.

Options for the AFP2WEB Transform

Table 14 lists the options that you can specify in the AFP2HTML.INI file to convert documents with the AFP2WEB Transform.

Table 14. Options for the AFP2WEB Transform

Option in AFP2HTML.INI file	**Description**
`AllObjects`	Determines how ODWEK processes documents that are stored as large objects in OnDemand. The default value is `0` (zero), and means that ODWEK will retrieve only the first segment of a document. If you specify `1` (one), then ODWEK will retrieve all of the segments and convert them before sending the document to the client. **Note:** If you enable large object support for very large documents, then your users may experience a significant delay before they can view the document at the client.
`ScaleFactor`	Scales the output with the given scale factor. The default value is `1.0`. For example, specifying a value of `ScaleFactor=2.0` scales the output to be twice as large as the default size; specifying a value of `ScaleFactor=0.5` scales the output to one half of the default size. The default size is derived from the Zoom setting on the Logical Views page in the OnDemand application.
`SuppressFonts`	Determines whether the AFP text strings are transformed. If you specify `SuppressFonts=TRUE`, any text that uses a font listed in the Font Map file is not transformed. The default value is `FALSE`, which means that all of the AFP text strings are transformed. The Font Map file is identified with the `FontMapFile` option

Table 14. Options for the AFP2WEB Transform (continued)

Option in AFP2HTML.INI file	**Description**
`FontMapFile`	Identifies the full path name of the Font Map file. The Font Map file contains a list of fonts that require special processing. The default Font Map file is named `imagfont.cfg` and resides in the directory that contains the AFP2WEB Transform programs. See the AFP2WEB Transform documentation for details about the Font Map file.
`ImageMapFile`	Identifies the image mapping file. The image mapping file can be used to remove images from the output, improve the look of shaded images, and substitute existing images for images created by the AFP2WEB Transform. Mapping images that are common across your AFP documents (for example, a company logo) reduces the time required to transform documents. If specified, the image mapping file must exist in the directory that contains the AFP2WEB Transform programs. See the AFP2WEB Transform documentation for details about the image mapping file.

Note: ODWEK sends the following options to the AFP2WEB Transform when converting documents. These options are not specified in the AFP2HTML.INI file.

- Orientation. Determines the rotation value to use when viewing the document. The default value is derived from the Orientation setting on the View Information page in the OnDemand application.
- Image Color. Determines the color to use when viewing images and graphics. The default value is derived from the Image Color setting on the Logical Views page in the OnDemand application.

Viewing converted documents

The `UseApplet` option in the AFP2HTML.INI file is a directive to ODWEK that determines whether the AFP2HTML applet will be used to view the converted output. The default value is TRUE. If you specify FALSE, (the AFP2HTML applet is not used to view the output), then the output is formatted and displayed by the Web browser.

In general, IBM recommends that you always use the AFP2HTML applet to view converted documents. If a document was stored in OnDemand as a large object, then the AFP2HTML applet adds controls to help users easily move to any page in the document.

Appendix F. AFP to PDF transform

The AFP2PDF Transform converts AFP documents and resources into PDF documents. The AFP2PDF Transform is a services offering from IBM Printing Systems Division. An administrator must install and configure the AFP2PDF Transform on the HTTP server. See your IBM representative for more information about the AFP2PDF Transform services offering. Someone in your organization must also specify configuration options for the AFP documents and resources that you plan to process with the AFP2PDF Transform. This section describes how to specify the configuration options.

Note: In this document, the name AFP2PDF.INI refers to the configuration file. To specify the file that contains the configuration options, see "CONFIGFILE" on page 22.

The AFP2PDF.INI file provides configuration options for the AFP2PDF Transform. You typically configure the AFP2PDF.INI file with options for specific AFP applications. However, you can also provide a set of default options. The AFP2PDF Transform uses the default options when converting documents and resources for AFP applications that are not identified in the AFP2PDF.INI file. To learn more details about the options and the conversion process, see the AFP2PDF Transform documentation.

The following topics provide additional information about the AFP2PDF.INI file:
- Specifying the AFP2PDF.INI file
- Viewing converted documents

Note: To convert documents, you must also specify the `AFPVIEWING=PDF` parameter in the DEFAULT BROWSER section (or other browser sections) of the ARSWWW.INI file. See "AFPVIEWING" on page 31 for details. (If you plan to use the Retrieve Document API, then you should specify the `_afp=PDF` parameter. See "Retrieve Document" on page 75 for details.)

Specifying the AFP2PDF.INI file

The following is an example of an AFP2PDF.INI file:

```
[CREDIT-CREDIT]
OptionsFile=
ImageMapFile=creditImageMap.cfg

[default]
OptionsFile=
ImageMapFile=imagemap.cfg
AllObjects=0
```

The structure of the file is similar to a Windows INI file, and contains one stanza for each AFP application and one default stanza. The title line of the stanza identifies the application group and application. For example, the title line:

```
[CREDIT-CREDIT]
```

Identifies the CREDIT application group and the CREDIT application. Use the – (dash) character to separate the names in the title line. The names must match the application group and application names defined to the OnDemand server. If the application group contains more than one application, then create one stanza for each application.

The parameters that you specify in the `[default]` stanza are used by the AFP2PDF Transform to process documents for AFP applications that are not identified in the AFP2PDF.INI file. The default parameters are also used if an AFP application stanza does not include one of the parameters specified.

The `OptionsFile` parameter identifies the full path name of the file that contains the transform options used by the AFP2PDF Transform. The transform options are used for AFP documents that require special processing. See the AFP2PDF Transform documentation for details about the transform options file.

The `ImageMapFile` parameter identifies the image mapping file. The image mapping file can be used to remove images from the output, improve the look of shaded images, and substitute existing images for images created by the AFP2PDF Transform. Mapping images that are common in most of your AFP documents (such as a company logo) reduces the time required to transform documents. If specified, the image mapping file must exist in the directory that contains the AFP2PDF Transform programs. To specify the directory that contains the programs for the AFP2PDF Transform, see "INSTALLDIR" on page 22. See the AFP2PDF Transform documentation for details about the image mapping file.

The `AllObjects` parameter determines how ODWEK processes documents that are stored as large objects in OnDemand. The default value is `0` (zero), and means that ODWEK will retrieve only the first segment of a document. If you specify `1` (one), then ODWEK will retrieve all of the segments and convert them before sending the document to the client. **Note:** If you enable large object support for very large documents, then your users may experience a significant delay before they can view the document at the client.

Viewing converted documents

To view converted documents with the Adobe Acrobat viewer, you must obtain the viewer for the browsers used by your organization.

Appendix G. HTTP server configuration files

This section contains examples of the following HTTP server configuration files:

- HTTP Original Server (no longer supported)
- HTTP Apache Server
- WebSphere Application Server

Note: Please consult your HTTP documentation for configuration assistance.

HTTP Apache Server

Notes:

1. In the `Listen` line, replace the string **n.n.n.n** with the IP address of the iSeries machine that is running the OnDemand server software.
2. In the `Listen` line, replace **p** with the TCP/IP port number used for the client and HTTP server to communicate. The default TCP/IP port number is 80.
3. In the `DefaultNetCCSID` line, replace the string **xxxx** with the CCSID of the OnDemand instance.
4. In the `ServerName` line, replace the string **f.q.h.n** with the fully-qualified TCP/IP host name of the iSeries machine that is running the OnDemand server software.
5. For DBCS languages, change the `CGIConvMode` line to:

   ```
   CGIConvMode %%EBCDIC_JCD/MIXED%%
   ```

```
#HTTP Apache Server Configuration for the ODWAPACHE server
Listen n.n.n.n:p
DefaultNetCCSID xxxx
HostNameLookups off
RuleCaseSense OFF
LimitRequestBody 102400
ErrorLog /www/odwapache/logs/error_log
ServerName f.q.h.n
DocumentRoot /www/odwapache
UseCanonicalName Off
CGIConvMode %%EBCDIC/MIXED%%
ScriptLogLength 200
ScriptLog /www/odwapache/logs/cgi_log
Alias      /logon      /www/odwapache/odw_logon.htm
AliasMatch ^/images/(.*)$                      /QIBM/UserData/OnDemand/www/images/$1
AliasMatch ^/applets/com/ibm/edms/od/(.*)$   /QIBM/ProdData/OnDemand/www/applets/$1
AliasMatch ^/applets/(.*)$                     /QIBM/ProdData/OnDemand/www/applets/$1
ScriptAliasMatch ^/scripts/arswww.cgi$       /QSYS.LIB/QRDARS.LIB/ARS3WCGI.PGM
ScriptAliasMatch ^/scripts/arswww\.cgi/(.*)$ /QSYS.LIB/QRDARS.LIB/ARS3WCGI.PGM
AddType www/unknown cab
AddType www/unknown jar
AlwaysDirectoryIndex On
DirectoryIndex index.html
<Directory/>
   Options None
   Options +ExecCGI
   AllowOverride None
   Order deny,allow
   Deny from all
   <LimitExcept GET HEAD OPTIONS TRACE POST PUT>
   </LimitExcept>
</Directory>
<Directory /QIBM/ProdData/OnDemand/www/applets/>
```

```
   Order allow,deny
   Allow from All
</Directory>
<Directory /QIBM/ProdData/OnDemand/www/images/>
   Order allow,deny
   Allow from All
   Options +FollowSymLinks
</Directory>
<Directory /QIBM/ProdData/OnDemand/www/samples/>
   Order allow,deny
   Allow from All
   Options +FollowSymLinks
</Directory>
<Directory /www/odwapache/>
   Order allow,deny
   Allow from All
</Directory>
<Directory /QSYS.LIB/QRDARS.LIB>
   Order allow,deny
   Allow from All
   Options +ExecCGI
</Directory>
```

| ## WebSphere Application Server

| A sample WebSphere configuration file can be obtained from the IBM Content
| Manager OnDemand for iSeries support Web page at
| http://www.ibm.com/software/data/ondemand/400/support.html. Search for
| *ODWEK WebSphere example* in the Technotes category.

Appendix H. 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 28 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`. See Table 15.

Table 15. 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 `criteriaN` 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 I. National language support

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 OnDemand database on the 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 OnDemand database on the 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 16 lists the DBCS code pages and languages supported by 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 OnDemand database. The **LANGUAGE=** column lists the values that are associated with the translated message catalogs.

Note: Linux is not listed in the table because Linux cannot be the target server for an ODWEK application. That is, although ODWEK may run on a Linux system, the OnDemand database (OnDemand library server) cannot run on a Linux system.

Table 16. DBCS code pages, languages, code sets and locales

Territory	LANGUAGE=	OS	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 or OS/390 (EBCDIC)	935	935	IBM-935	—
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 or OS/390 (EBCDIC)	939	939	IBM-939	—

Table 16. DBCS code pages, languages, code sets and locales (continued)

Territory	**LANGUAGE=**	**OS**	**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 or OS/390 (EBCDIC)	933	933	IBM-933	—
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 or OS/390 (EBCDIC)	937	937	IBM-937	—

For more information about configuring the OnDemand system for DBCS languages, see “National Language Support” in *Planning and Installing*.

Appendix J. Problem determination tools

You can use the tools listed in Table 17 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 plug-ins.

Table 17. 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
Server Log Files	Save access information, errors, and server information.	Do the following: 1. In the DEBUG section of the ARSWWW.INI file, set the `LOG` parameter to 1 (one). The log file that is generated by ODWEK is named ARSWWW.LOG and is written to the directory specified by the `LOGDIR` parameter. (The default directory is `/QIBM/UserData/OnDemand/WWW/LOG`.) **Important:** If specified, the DEBUG section must be the first executable statement in the ARSWWW.INI file. 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.) **Note:** 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.
Java Console	Display messages generated by the applets.	• Netscape: From the Communicator menu, select Tools and then Java Console. • Internet Explorer: 1. From the View menu, select Internet Options. 2. On the Advanced page, select Java Console. 3. Restart the browser. 4. From the View menu, select Java Console.

Table 17. Problem Determination Tools (continued)

Tool	**Purpose**	**How to Enable**

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.

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.	Do the following: 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.

Notices

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
Software Interoperability Coordinator
3605 Highway 52 N
Rochester, MN 55901–7829
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 information 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.

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.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and service marks

Advanced Function Presentation, AFP, AS/400, IBM, iSeries, Operating System/400, OS/390, OS/400, Redbooks, WebSphere, and z/OS are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Index

Program Number: 5722-RD1

SC27-1163-02