

Middleware - Huh?

Julia Belinski
PRYORity Training and Development
Atlanta, GA
www.pryoritytraining.com/taac7
770-522-9202

NOTE: This presentation will be updated as further information is collected on middleware and its support on Web Servers, operating systems, and platforms.

Middleware (a definition)

In the computer industry, middleware is a general term for any programming that serves to "glue together" or mediate between two separate and often already existing programs. A common application of middleware is to allow programs written for access to a particular database to access other databases.

(*source:* searchWebServices.com)

We will discuss Middleware to the following File Types

- RTF files
- XML Files
- Databases
 - Flat Files – Text (ASCII) file, Spreadsheets (i.e. Excel)
 - Relational database (RDB) - A relational database stores all its data inside tables. There is a relationship between the rows and columns of the table.
 - MySQL, Access, Oracle, DB2
 - Drivers needed to access a database

OUTLINE

Scenario 1	CD-ROM/Local Drive/Network	<i>Input/Output:</i>	Text file (ASCII)
Scenario 2	CD-ROM/Local Drive/Network	<i>Input/Output:</i>	RTF file
Scenario 3	CD-ROM/Local Drive/Network	<i>Input/Output:</i>	Database – no drivers
	CD-ROM/Local Drive/Network	<i>Input/Output:</i>	Database – ODBC drivers
Scenario 4	ALL	<i>Input:</i>	XML
Scenario 5	Internet/Intranet	<i>Input/Output:</i>	Text file (ASCII) – via FTP
Scenario 6	Internet/Intranet	<i>Input/Output:</i>	all databases – via scripting languages

SCENARIO 1:

Delivery: CD-ROM/Local Drive/Network

Input/Output: Text file (ASCII)

Functions: ReadExtFile
WriteExtFile
AppendExtFile

Variables: FileLocation, RecordsLocation, or custom drive letter

Additional Notes:

Cross-platform: determine path format based on platform at runtime

\ (slash) Windows
: (colon) Macintosh

```
if OSNumber = 3 then      --this is a running on a Windows platform
  path := "\\\"
else                      -- this is running on a Macintosh platform
  path := ":"
end if
```

Custom drive letter: options

- . Set at installation – use an ini file or xml file
- . Ask the user during runtime – i.e. login/when writing file/other

SCENARIO 2:

Delivery: CD-ROM/Local Drive/Network

Input/Output: RTF file (rich-text format)

Create RTF file: *RTF editor or rtfSave (an Authorware function)*

NOTE: you can also save to an rtf file just using WriteExtFile or AppendExtFile

Middleware: **RTFobject.u32** (included with Authorware)

Functions: **Part of the RTFobject.u32**

Windows -> Functions->custom category->Load

rtfCreate and **rtfShow** (read and display data from RTF file)

rtfSave – to export from Authorware to an RTF file or graphic (BMP, JPG, GIF). Only exports what it is read from an existing RTF file (this might just be a portion of the file – like page 1 only).

```
--assume filename = file1.rtf
```

```
--creates an RTF object, reading from a file from page 2 - 3 with scrolling
```

```
rtfID = rtfCreate(0, 0, 100, 300, FileLocation ^"file1.rtf", 1, 0, 2, 3)
```

```
rtfShow(rtfID) --display the RTF object just created
```

SCENARIO 3:

Delivery: CD-ROM/Local Drive/Network
Input/Output: Database

Option 1: no database drivers
V12 Database Engine (Integration New Media)

Middleware: V12-DBE for Authorware.x32 (Purchase from Integration New Media)

- no drivers required
- Features: sorting, record count, retrieve specific records, Key (field), large capacity
- cross platform, does not require ODBC or any installation (just serial number input via Authorware)
- Retrieving/writing records: uses V12 functions

Functions: Found in the V12....x32
Window -> Functions->custom category->Load

Distribution:

- V12-DBE for Authorware.x32 (*in the Xtras folder*)
- V12 database (filename.v12)

SCENARIO 3: (continued)

Delivery: CD-ROM/Local Drive/Network

Input/Output: Database

Option 2: ODBC (Open Database Connectivity)

Middleware: ODBC.U32 (included with Authorware)

- **Window -> Functions->custom category->Load**

Requirements:

- define Data Source (alias for the database file).
 - Usually created through the ODBC administrator (Control Panel)
 - Can also purchase programs that will setup the Data Source and set the Registry.
 - Can also use an installation program to modify ODBC.ini and set the registry.
- Driver for database must be installed (usually included with operating system)
- ODBC.U32
- ODBC compliant database

Retrieving/writing records: SQL language**Distribution:**

- ODBC.U32
- Setup data source.
 - Usually setup through Control panel on local machine or network server. This can be done through an installer.
 - When a data source is created, its info is written to the text file called: odbc.ini
- Database driver
 - License to distribute any drivers. Note: most drivers will already reside on a users computer for the most common databases.
- Database file

ODBC example: See odbc.a6p on the Authorware Software CD in the Goodies folder.

SCENARIO 4:

Delivery: all
Input: XML - formatted text files

- XML - Extensible Markup Language
- XML files are ASCII text files that contain tags (called nodes) around data or content.
- XML structures (or describes) data through the use of the nodes.
- Applications process the XML and determine how to display (or use) the documents data.

Example of XML syntax:

```
XML formatting
<nodeL1 attr = "..."_
    <nodeL2 attr = "..."_
        <nodeL3a>

        </nodeL3a>
        <nodeL3b>

        </nodeL3b>
    </nodeL2>
</nodeL1>
```

- Using XML with Authorware, use the XML documents to designate fields or pockets of information (image/audio/text file, content text, test questions).
- XML files are first read, then parsed (nodes are interpreted) and content extracted per the node requested.
- One way to create XML is with Dreamweaver. Create a template and create editable regions that correlate to each data element (node) that will be saved or extracted. Note: non-editable regions will not be exported.

Middleware: XmlParser.X32 (included with Authorware)

Requirement for development and distribution (when used with Authorware):

- XmlParser.U32
 - This xtra is included with Authorware 6 and its functions are found in the list of functions (under XML parser). It is not found under the custom functions.
- Knowing the structure and content of the XML file.
- Limit of XML file - 30K characters (because the whole XML file must be read into the Authorware program)

XMParser example to read data from an XML file

```
If NetConnected = False then  --not on the internet
    mXMLtext := ReadExtFile(filename.xml)
else
    mXMLText := ReadURL(filename.xml, timeout )
end if
--if no text is returned, use ioMessage and ioStatus to check for errors
```

SCENARIO 5:

Delivery: Internet/Intranet

Input/Output: Text file

Input/Retrieve Text File: To read a text file that resides on the Internet/Intranet, no additional tools are needed outside Authorware.

Functions: **ReadExtFile()** –cannot use on a Secure Server usually
ReadURL()

Variable: **NetLocation**

Write Text File: You cannot use WriteExtFile/AppendExtFile to write to files that reside on a Server (Internet/Intranet)

Output Text Option:**FTP (File Transfer Protocol)**

Transfer data to/from internet/intranet

Or Create script using a compiled language such as Java or C or use a scripting language such as ASP/Visual Basic or a CGI Script in PERL or PHP (See Scenario 6)

Middleware for FTP: FTP.u32 (included with Authorware)**Requirements for FTP:**

- Internet/intranet must have FTP capabilities (FTP server)
- Distribution and development:
 - FTP.U32
 - FTP Server information:
 - Server address – in the form of a number (999.999.999.99) or name.
 - Port (usually 21)
 - Userid
 - Password
- Functionality – FTP functions (from UCD)
- Advantages
 - Just uses the functions provided with the UCD. No additional programming required.
 - Password/User protected information
- Disadvantages:
 - Some servers can be very slow
 - Not very secure or reliable.

Steps to use the FTP.U32 in Authorware

- 1 Write the file to the local drive (WriteExtFile). Usually will want to write to the same location as the Web Player and xtras – which is defined by FileLocation when running from a Published piece.
- 2 Initialize an FTP session (FtpOpen function).
- 3 Test for the successful execution of FtpOpen. Error = 0 If no error, the handle to the FTP session is returned. Note: This handle is used in other functions
- 4 Open an FTP connection to a specific server (FtpConnect function). This requires the server name, user name, and password and usually connection to a specific port (often this is 21).
- 5 Test for the successful execution of FtpConnect using the FtpStatus function. 1 means the operation is still waiting. A negative number (< 0) means the operation has failed. 0 means operation was successful.
- 6 Upload the file to the FTP server (FtpStore function)
- 7 Test for the successful execution of FtpStore using the FtpStatus function.
- 8 Shut down the FTP connection (FtpDisconnect function).
- 9 Test for the completion of the FtpDisconnect operation using the FtpStatus function. 1 means the operation is still waiting. A negative number (< 0) means the operation has completed.
- 10 Release the resources allocated for the FTP session (FtpClose function)

TIP: Use **FTPdemo.a6p** found in the Goodies folder to play around and test your FTP connections.

SCENARIO 6:

Delivery: Internet/Intranet

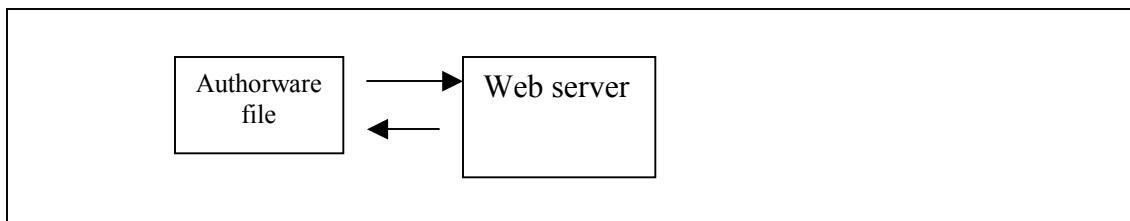
Input/Output: all database types (including flat files)

To process data that will be kept on the server (server-side), the following process is usually followed.

The static Process (when Web Server understands and can process all scripting/code – HTML, JavaScript):

- Web Server responds to client (browser) request when a URL is entered.
 - Authorware – POSTURL or GETURL
- The Web server maps (matches) the URL to a file on the server (or file on the Servers network of drives)
- The Web Server then returns the requested document to the client (browser displays the document).

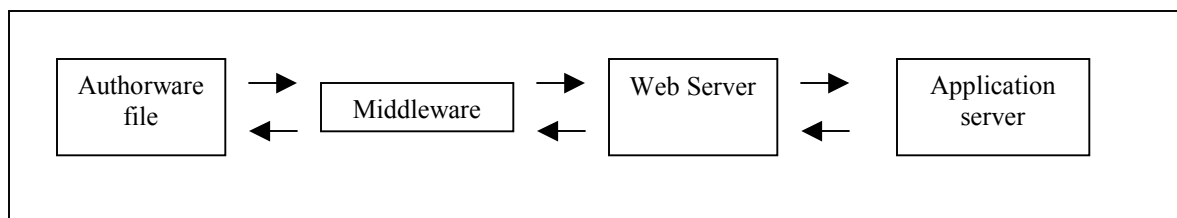
Picture of static process



The dynamic process – Web Server needs help in processing special code/tags – ASP, ColdFusion, CGI scripts):

- Web Server responds to client (browser) request when a URL is entered.
 - Authorware – POSTURL or GETURL
- The Web server maps (matches) the URL to a file on the server (or file on the Servers network of drives)
- An application server processes any special code and sends response back to Web Server.
- The Web Server then returns the requested document to the client (browser displays the document).

Picture of dynamic process



HTTP usually provides the communication protocol for the transfer between client and Web Server.

Two HTTP request types (for information from the server)

1. **GET** – sends (form) content as part of the URL and retrieves the resource from the Web

- **To GET in Authorware, use ReadExtFile() or ReadURL()**
- **Example:**
`readURL("http://www.who.com/search.cgi?fname=Julia&lname=Belinski")`
 - Where everything before the ? is the URL
 - Everything after the ? (question mark) is seen as user-specific input.
 - Each pair (key = value) is separated by & (ampersand)
 - There aren't any quotes around strings
 - There aren't any spaces. Spaces should be replaced with the hexadecimal value (%20). All special characters should be converted to hexadecimal value.
 - The "keys" (i.e. fname and lname) are variables in the middleware program (search.cgi in this example) that it is expecting.
- **Disadvantage of the GET:**
 - Data being sent to the server is visible to the user
 - Limit of the URL (plus the input) = 1024 characters (includes spaces and special characters)

2. **POST** – data is not part of the URL and is not seen by the user.

- Authorware only uses the POST method through the use of PostURL
- The length of the URL is limited to 2K, the content is limited to 30K, and the return string is limited to 32K (because of the string variable limit in Authorware).

Example of PostURL:

```
result := PostURL("http://www.who.com/login.cgi", "fname=Julia&lname=Belinski")
```

Or using variables for the content to Post

```
result := PostURL("http://www.who.com/login.cgi", "fname="^fnameVar ^
"&lname="^lnameVar)
```

--check for error

```
if result = "" | IOStatus <> 0 then --there was an error
```

```
  GoTo(@"displayError") --jump to icon called "displayError" to display the error
end if
```

Use of Scripting languages to access databases (including flat -ASCII text – files) residing on the server.

1. Interpreted languages (most are called scripting languages):

- Execute high-level language programs directly without the need for compiling.
- AGAIN: Do not need to be compiled
- Require the server to have the Scripting interpreter for the language.
- Slower than Compiled languages but easier to update (because they don't have the extra step of being compiled).
- Many scripting languages can be created/written by programs (i.e. Dreamweaver MX) that create the correct (syntax) script for you for a particular scripting language.

2. Compiled (Source) Languages (C, Fortran, Cobol)

- After the program is written, it must be run through a compiler for the computer to understand it.
- Usually much higher learning curve (Programming)

CGI Scripts

- “Common Gateway Interface”
- Interfaces between applications and information servers (HTTP or Web Servers)
- CGI scripts can be written in most languages: PERL, PHP, Python and C++, Applescript and compiled languages like C and Fortran
- Usually will want to use a CGI Bin for the scripts, as they are more secure.
- Among other things, can access and write to databases, create dynamic HTML pages,
- Slowly being replaced by more advanced scripting languages such as ColdFusion and ASP and JSP.

PERL (often used in CGI scripts)

- Interpretative language
- Script itself is just written in ASCII text and can be created with any Text editor package.
- Setup: Application Server/Interpreter: Install PERL software
 - ActivePerl (www.activestate.com) is PERL interpreter for Windows
 - MacPerl (www.macperl.com) for Mac
- PERL is usually already installed on a UNIX system
- CGI.pm is a module (can download from the internet) to create CGI Scripts – including those scripts written in PERL.
- PERL scripts are processed on the Web Server (server-side) not client-side. Because of this, users cannot access the code/script.
- Usually PERL scripts are saved in cgi-bin folder on Website.

PHP

- Extension: .php
- Setup: Application Server: Install PHP software (www.php.net)
- Advantages:
 - Becoming more and more popular (replacing PERL)
 - Can be embedded directly in an HTML document
 - Faster and easier than PERL
 - Runs on anything (platforms, Web servers)
- Disadvantage: Not backed by a company (like ColdFusion is)

Python

- Interpretative language
- Extension: .py
- Becoming more and popular (over PERL and PHP)
- Object-oriented language and is extendable with C++ and C – it has lots of power!
- Cross platform
- Open source
- Setup: Application Server/Interpreter: Install Python software (www.python.org)

ColdFusion

- Interpretative language
- Extension: .cfm
- Web server needs to support it
- Macromedia product
- Dreamweaver MX can create ColdFusion scripts
- Setup: Application Server/Interpreter: Install ColdFusion software (Macromedia site)
- Compared against PHP, ASP, JSP
- Advantages:
 - Shorter development time (vs ASP and CGI scripts). Requires fewer lines of code to handle the same thing
 - Integrates with Macromedia products (Dreamweaver, Flash)
 - Cross platform and many operating systems
 - Becoming very popular and many developers exist for support
- Disadvantage:
 - Can be expensive to have a hosting/dedicated server
 - Expensive software

ASP (Active Server Pages)

- Interpretative language
- Can use any scripting language to create ASP: Vbscript (most common in ASP), C++, Python
- Extension: .asp
- Originally only worked on Windows platform but is becoming more and more available (supported) on any Web server and across many operating systems (not just Windows).
- Server-side
- Interpreter (ASP.dll) for ASP 3.0
- New version is ASP.NET : extension is .aspx
- Disadvantage: couldn't really find that much on the Internet in its favor! Except that there are a lot of people using it.

Things to ask your Web Hosting service (Internet Service Provider - ISP) to help you determine if your choice of middleware is even feasible:

- Do they support the scripting language you are choosing (i.e. ColdFusion, ASP)?
- Do they support (allow) the type of database you want to use? And with the plan that you have signed up for? How much more does it cost?
- Is there a limit to the number of ODBC data sources?
- Do they provide any help with the database creation or scripting?
- If using CGI scripts, does your account provide a CGI Bin?

So what is the bottom line on Middleware? HUH?

Let us know if you would like to receive updates on this topic. Or contribute to this topic. Send us an email at info@prioritytraining.com Subject: Middleware