STELLENT™

Environment Integration Guide

SCS-EN8-610

Table of Contents

Chapter

1

OVERVIEW

INTRODUCTION

The information contained in this guide is based on Stellent™ Content Server 6.1. The information is subject to change as the product technology evolves and as hardware and operating systems are created and modified.

Due to the technical nature of browsers, web servers, and operating systems, Stellent, Inc. cannot warrant compatibility with all versions and features of third-party products.

This chapter contains these topics:

- About this Guide
- Stellent Product Distinctions
- If You Need Assistance

ABOUT THIS GUIDE

This guide assumes that you are familiar with Stellent products and the architecture of the Content Server. In addition, this guide assumes that you are familiar with the underlying architecture of the various protocols, interfaces, and mapping services that are described in each chapter.

Audience

This guide presents protocol integration concepts and is intended for application developers. Developers customize the Stellent software to suit the content management needs specific to a company, agency, or industry. They create and enable custom components to modify Stellent's standard functionality. Common tools include IdocScript, which is the custom scripting language; IdcCommands, which are the command-line entries; and other standard protocols such as SOAP, COM, Java, etc.

Conventions

The following conventions are used throughout this guide:

- The notation **<*install_dir*>/** is used throughout this guide to refer to the location on your system where Stellent Content Server product is installed.
- Forward slashes (/) are used to separate the directory levels in a path name. A forward slash will always appear after the end of a directory name.
- Notes, technical tips, important notices, and cautions use these conventions:

Symbol	Description
	This is a note. It brings special attention to information.

Symbol	Description
	This is a tech tip. It identifies information that can be used to make your tasks easier.
	This is an important notice. It identifies a required step or critical information.
	This is a caution. It identifies information that might cause loss of data or serious system problems.

STELLENT PRODUCT DISTINCTIONS

In this guide, the term *content server* is used generically to refer to both the Content Server and the Collaboration Server. The following table lists the distinctions of these two Stellent content management solutions:

Stellent Content Management Product and Feature Distinction

Product	Description
Stellent Content Server	A fully functional content management system providing end-to-end content management and personalized delivery of that content.
Stellent Collaboration Server	A fully functional content management system providing end-to-end content management and personalized delivery of that content. Additionally, a Stellent Collaboration Server license enables project-level security for collaborative authoring environments.

IF YOU NEED ASSISTANCE

The Stellent family of products is backed by a full range of support options to meet every business need. The service philosophy is to keep your Stellent environment fully operational by providing the best information and solutions available. The Stellent product support team consists of highly trained product engineers who excel at resolving complex technical issues. Every customer inquiry is tracked and managed through automated systems.

Important: The support options that are available for specific systems may vary, depending on the applicable service and maintenance agreements. Please refer to your contract for the support details for your Stellent system.

Support Options

You can choose from the following three support programs offered by Stellent:

- **Standard Maintenance and Support Program:** The standard support program is available during standard business hours domestically and internationally (Monday through Friday from 8 am to 5 pm for every time zone). It provides telephone and e-mail support for troubleshooting, bug fixes, call escalation, modifications, enhancements, and updates.
- **SDK Developer Support Program:** The SDK support program is available Monday through Friday from 8 am to 5 pm (Central Time in the USA, which is -6 hours from GMT). It provides telephone and e-mail support for customers who wish to use the Software Developer's Kit (SDK) to customize their Stellent systems.
- **Extended Support Program:** The extended support program provides the standard support services 24 hours a day and 7 days a week.

Note: Value Added Resellers (VARs) and Original Equipment Manufacturers (OEMs) may have different support programs in place.

Before Contacting Support

When you call or send e-mail, please provide the following information:

- Nature and severity of the problem
- Stellent product and version
- Serial number of the registered Stellent product
- Operating system and version.
- Name and telephone number of the person the support engineers should contact if they need to call back

In addition, depending on the situation, it may be helpful to know the following:

- Database type and version
- Web browser type and version
- Web server type and version

Telephone

Technical support is available from the Support Hotline at 1-888-688-TECH (1-888-688-8324). The Support Hotline is accessible toll-free world-wide.

E-Mail

The Stellent support e-mail address is *support@stellent.com*. It is available for all technical support questions.

Internet

Technical support is also available through the Internet at http://support.stellent.com. You will be prompted for a username and password. To obtain a username and password, contact the Support Hotline at 1-888-688-TECH (1-888-688-8324).

Chapter

2

ENVIRONMENT INTEGRATION

INTRODUCTION

This chapter provides a general introduction, a conceptual overview of integrating with Stellent Content Server, and descriptions of the available integration options.

This chapter contains these topics:

- Stellent Content Integration Kit
- Conceptual Overview
- Integration Options
- Additional Documentation for Developers

STELLENT CONTENT INTEGRATION KIT

This guide together with the *Stellent Infrastructure/Security Integration Guide* form the Stellent Content Integration Kit. These reference guides provides a starting point for the application developer and system integrator. The focus of the *Environment Integration Guide* is to present the available integration options, and suggest an approach, (like IdcCommand X, or persistent URL, or SOAP) and to provide information on where to get the detailed documentation on that approach.

CONCEPTUAL OVERVIEW

The *Environment Integration Guide* provides a general discussion of application integration methodologies. Specifically, this guide provides basic conceptual information about the integration of Stellent Content Server within various network system environments using various protocols, interfaces, and mapping services.

In general. these various integration methodologies serve to translate or pass methods and associated parameters with the goal of executing content server services. The various content server services are the “window” for accessing the content and content management functions within Stellent Content Server.

Note: For general information on the architecture and functionality of the Stellent family of products, as well as a number of key concepts, refer to the *Stellent Content Server Getting Started Guide*.

The Stellent Content Server can be integrated with other enterprise applications to access content and to provide content management capabilities such as full-text and metadata searching, library services, workflow, subscription notifications and content conversion capabilities using a wide array of integration methods.

For example, one simple integration option is to reference content that is managed within Stellent by persistent URL. Other integration options are to use the Stellent Content Server Enterprise JavaBean, the Java API, the COM component, or the ActiveX control.

INTEGRATION OPTIONS

The *Environment Integration Guide* provides a general discussion of these application integration methodologies:

- Chapter 3: Java API (IdcCommand)
- Chapter 4: Component Object Model (COM)
- Chapter 5: Java Server Pages (JSP)
- Chapter 6: Java 2 Enterprise Edition API (J2EE)
- Chapter 7: Common Object Request Broker Architecture (CORBA)
- Chapter 8: Open Document Management API (ODMA)
- Chapter 9: Simple Object Access Protocol (SOAP)
- Chapter 10: Web Distributed Authoring and Versioning (WebDAV)

Java API (IdcCommand)

Use the Stellent Java API to gain access to the content and content management functions within Stellent Content Server. The Stellent IdcCommand Java Command Utility is a stand-alone Java application that enables users to execute content server services.

Component Object Model (COM)

Use a COM interface to integrate Stellent Content Management with Microsoft environments and applications. Stellent provides an ActiveX control and an

OCX component as interface options to gain access to the content and content management functions within Stellent Content Server.

Java Server Page (JSP)

Stellent Content Server core functionality can be accessed from a JSP running in Stellent Content Server, from a JSP through the Stellent Content Server JavaBean, or from a JSP through the Stellent Content Server Enterprise JavaBean (EJB) deployed on your J2EE application server. In addition, the Stellent JSP Tag Libraries are provided to simplify JSP coding and to easily access content server services.

Java 2 Enterprise Edition API (J2EE)

Use the J2EE API to gain access to the content and content management functions within Stellent Content Server by deploying the Stellent Content Server Enterprise JavaBean on your J2EE compliant application server.

Common Object Request Broker Architecture (CORBA)

Use the CORBA API to gain access to the content and content management functions within Stellent Content Server by implementing RMI over IIOP as the transport protocol and referencing the Stellent Enterprise JavaBean.

Open Document Management API (ODMA)

Use the Stellent ODMA-based plug-in to gain access to the content and content management functions within Stellent Content Server (for ODMA-compliant desktop applications).

Simple Object Access Protocol (SOAP)

Use a SOAP interface to access the content and content management functions within Stellent Content Server and to deploy your content management capabilities as a Web service. The SOAP protocol integrates .NET servers, J2EE application servers, or other systems with XML-based interfaces.

Web Distributed Authoring and Versioning (WebDAV)

Use a WebDAV-enabled desktop or business application to connect directly to the Stellent Content Management system and avoid the need for additional client-side software.

ADDITIONAL DOCUMENTATION FOR DEVELOPERS

The following documentation is available for developers of the Stellent software:

- Customizing Content Server (PDF)

  This document provides a general explanation of how the system works and background information required for performing customizations. It supplies the information developers need to develop custom components for Stellent Content Server. Information includes code references, technical tips, and examples.

- Idoc Script Reference Guide (PDF)

  This document provides information about IdocScript applications, functions, predefined variables, configuration settings, HTML forms scripting, and web server variables. The document contains syntax, code references, examples, and descriptions.

- IdcCommand—Java Command Utility Reference Guide (PDF)

  This document provides information on the Java Command Utility and ActiveX Command Utility for Stellent Content Server. The IdcCommand utility is a stand-alone Java application that enables users to execute services. IdcCommandX is an ActiveX control that enables a program to execute a service and retrieve file path information.

- Creating Custom Conversion Engines (PDF)

  This document provides information on creating custom conversion engines for the Inbound Refinery and Visual Basic module API specifications. It provides developers with the information they need to create and implement multiple custom conversion engines for Inbound Refinery.

- IntradocClient OCX Component Reference Guide (PDF)

  This document provides information on the IntradocClient OCX Component. This component is an ActiveX control used to connect to a remote Content Server and perform server functions. A description of the setup process for the component in the Microsoft Visual Basic development environment and the IntradocClient OCX API specifications listing the Properties, Methods, and Events are provided.

- Using SOAP to Connect to Stellent Content Server (PDF)

  This document provides information on using the Simple Object Access Protocol (SOAP) interface to access the content and content management functions within Stellent Content Server and to deploy your content management capabilities as a Web service. The SOAP messaging protocol integrates .NET servers, J2EE application servers, or other systems with Extensible Markup Language (XML) based interfaces. The Stellent SOAP component allows a user to call an IdcService on a content server by constructing a SOAP-XML formatted request and passing this request via HTTP to the content server. The content server will then pass back a response in SOAP-XML format.

Chapter

3

JAVA API INTEGRATION

INTRODUCTION

You can use the Stellent Java API to gain access to the content and content management functions within Stellent Content Server. The Stellent IdcCommand Java Command Utility is a stand-alone Java application that enables you to execute content server services.

The IdcCommand Java Command Utility reads a command file containing commands and parameters and calls the specified services. A log file records the time that the call was executed, whether the command was successfully executed, and if there were execution errors.

INTEGRATION NOTES

The IdcCommand utility requires a command file and content server username to execute the commands. Some commands will require administrative access, other commands may require only write permission.

You can specify command options on the command line or in the intradoc.cfg configuration file. Specifying a command line option overrides the setting in the configuration file.

Note: Refer to “Command Line Options” and “Configuration File Options” in the *IdcCommand Reference Guide* for additional information.

- The IdcCommandFile specifies the file containing the commands to execute. The command line option -f overrides this parameter.
- The IdcCommandUser specifies the user permitted to run the command. The command line option -u overrides this parameter.
- The IdcCommandLog specifies the path to the log file. The command line option -l overrides this parameter.
- The ConnectionMode specifies the connection mode for executing the commands. The command line option -c overrides this parameter.
  - If this option is not specified, the default connection mode of auto is used. In the default connection node, IdcCommand attempts to connect to the server.
  - If this fails, the commands are executed in stand-alone mode.

There are certain commands that cannot be executed in stand-alone mode. In general, these commands are performed asynchronously by the server in a background thread. This happens in the update or rebuild of the search index.

Command File Syntax uses precedence to resolve conflicts among the name/value pairs within the LocalData section. Special characters specific to the command file syntax are used.

The IdcCommand utility returns information about only the success and failure of the command. To retrieve information from the server in an interactive session, use the Java COM wrapper IdcCommandX available on NT.

Calling Services Remotely

To use services remotely, you must have these two files on the remote machine:

- <instal_dir>\bin\intradoc.cfg (same file as on the content server).
- <instal_dir>\config\config.cfg

In addition, these configuration entries must be defined in the #Additional Variables section of the config.cfg file on the remote machine:

- IntradocServerPort=4444
- IntradocServerHostName=IP or DNS

Executing the IdcCommand Utility

You run the IdcCommand utility from the command line. In general, You must specify the user name and the command file. These options are specified on either the command line or in an intradoc.cfg configuration file.

Note: See the *IdcCommand Reference Guide* for a list of available commands.

Perform the following steps to run IdcCommand from the command line.

1. Create a new directory.

   For example:

C:/idccommand

This directory will be used as the IdcCommand working directory., in which you store your configuration and command files.

2. Copy the intradoc.cfg configuration file from /stellent/bin/ directory into the newly created directory.

   The intradoc.cfg configuration file contains *Directory Variable* information and any *Additional Variable* information.

Important: Do not delete the *WebBrowserPath* or *IntradocDir* information. This directory variable information is required.

The intradoc.cfg configuration file will contain *Directory Variable* information (and optionally, *Additional Variable* information).

3. In the copied intradoc.cfg configuration file located in the newly created directory, define the configuration file option parameters.

Note: Refer to “Configuration File Options” in the *IdcCommand Reference Guide* for additional information.

In this example, IdcCommandFile references *newfile.hda* in the newly created *idccommand* directory. The IdcService must be defined in that file.

```
IdcCommandFile=newfile.hda
IdcCommandUser=sysadmin
IdcCommandLog=C:/idccommand/newlog.txt
```

```
# Directory Variables
WebBrowserPath=C:/Program Files/Plus!/Microsoft Internet/Iexplore.exe
IntradocDir=C:/stellent/

# IdcCommand Variables
IdcCommandFile=newfile.hda
IdcCommandUser=sysadmin
IdcCommandLog=C:/idccommand/newlog.txt
```

Figure 3-1 Configuration File

4. Create a command file in the newly created directory and add the desired command.

   In this example, a command file named newfile.hda. is created and a user called “Jennifer” is defined within that file. For the ADD_USER service, IdcService, dName, and dUserAuthType are required parameters.

```
@Properties LocalData
IdcService=ADD_USER
dName=Jennifer
dUserAuthType=Local
dFullName=Jennifer Smith
dEmail=jsmith@stellent.com
@end
```

Figure 3-2 Web Application Descriptor File

5. Run IdcCommand. Follow the steps to run IdcCommand on NT or to run IdcCommand on Solaris provided on the following pages.

Run IdcCommand on NT

To run IdcCommand on NT, follow steps 1 - 5 in the previous section and continue with these steps:

6. From the command line, change to the IdcCommand working directory created in step 1 (this directory contains the edited intradoc.cfg configuration file and the new HDA command file).

   For example:

   ```
   cd idccommand
   ```

7. From the command line, change the *jview* command to define the correct path of the virtual machine.

   For example:

   ```
   jview /cp:a C:/stellent/shared/classes/server.zip
   IdcCommand
   ```

8. Define the command options.

   For example:

   ```
   -f newfile.hda -u sysadmin -l C:/idccommand/newlog.txt
   ```

9. Execute the command.

Review the log file output. Common errors are syntax, failing to define the command options, or incorrect directory information in the .hda or .cfg files.

```
C:\idccommand>jview /cp:a C:/stellent/shared/server.zip
IdcCommand -f newfile.hda -u sysadmin -l C:/icdcommand/newlog.txt
```

Figure 3-3 NT Command Line Example

Run IdcCommand on Solaris

To run IdcCommand on Solaris, follow steps 1 - 5 in the previous section and continue with these steps:

6. From the command line, change to the IdcCommand working directory (created in step 1). This directory contains the edited

   For example:

```
cd <working directory>
```

7. From the command line, change the /bin/java command to define the correct path of the virtual machine and define the command options.

```
bin/java –classpath ./classes:./classes/idcserver3.zip:
<Oracle-JDBC-Driver-Path> IdcCommand –f newfile
```

Figure 3-4 Solaris Command Line Example

8. Execute the command.

Review the log file output. Common errors are syntax, failing to define the command options, or incorrect directory information in the .hda or .cfg files.

Related Documentation

Refer to this documentation for additional information:

- Idoc Script Reference Guide
- IdcCommand—Java Command Utility Reference Guide
- Customizing Content Server

Chapter

4

COM INTEGRATION

INTRODUCTION

The Stellent Content Server utilizes a Component Object Model based API which provides the capability to call Stellent functionality from within a Microsoft Component Object Model (COM) environment.

You can use a COM interface to integrate Stellent Content Management with Microsoft environments and applications. Stellent provides an ActiveX control and an OCX component as interface options to gain access to the content and content management functions within Stellent Content Server.

Tech Tip: Calling services from a command line on the local server using the Stellent IdcCommandX ActiveX Command Utility provides faster execution of commands than calling services remotely using the Stellent IntradocClient OCX component

Active Extension Control (ActiveX)

Use the Stellent ActiveX utility to access Stellent content management functions and to provide interoperability with other types of components and services.

Note: See *ActiveX Interface* (page 4-2) for information on calling services from a command line on the local server.

Object Linking and Embedding Control Extension (OCX)

Use the Stellent IntradocClient OCX component within a Windows Visual Basic development environment to gain access to the content and content management functions within Stellent Content Server.

Note: See *OCX Interface* (page 4-5) for information on calling services in a visual development environment, or to connect to a remote content server.

ACTIVEX INTERFACE

The Stellent IdcCommandX ActiveX Command Utility is an ActiveX control that allows a program to execute a service and retrieve file path information. You can call the IdcCommandX utility from both a Microsoft Visual Basic and Microsoft Visual C++ environment.

IdcCommandX Setup

Run the IdcCommandX ActiveX Command Utility setup executable located in the Extras/IDCCommandX sub-directory on the Stellent Content Server CD-ROM.

Executing Services

When executing services using the Stellent IdcCommandX ActiveX Command Utility keep these notes in mind:

- IdcCommandX must be initialized with a valid user and the intradoc.cfg directory. Outside of the `init` and `connection` managing methods, all methods use the serialized HDA format for communication.
- IdcCommandX attempts to establish a connection to a running server. If a connection is not made it fails.
- The returned serialized HDA format string contains information about the success or failure of the command. The `StatusCode` will be negative if a failure occurs, and `StatusMessage` will indicate the error.

Note: Refer to the *IdcCommand Reference Guide* for additional information.

Activex Integration Example

Calling IcdCommandX

Visual Basic

Add IdcCommandX as a control to the Visual Basic project and code the following lines to create and initialize the control:

```
Set idcCmd=CreateObject("Idc.CommandX")
idcCmd.init("sysadmin", "c:\stellent\bin")
```

Visual C++

Add the IdcCommandX control to the project and call the desired IdcCommandX class.

IdcCommandX Methods

Note: Refer to the *IdcCommand Reference Guide* for a compete description of IdcCommand and IdcCommandX methods and required parameters.

Public Function init(StellentDir As String) As Boolean

Initializes the system. You must call this function before using IdcCommandX and also specify the username and Stellent directory as parameters.

Public Sub executeCommand(Data As String)

Executes a service from the content server. This method automatically handles the connection to the server. It will not establish a connection if a connection has already been established with a *connectToServer* call. On completion of the command, the connection is closed.

Public Function ConnectToServer() As Boolean

Allows the calling program to establish a connection to the server. The connection is held open until a command is executed. Once a command is executed the connection is closed. This method does not need to be called. It is provided only as a convenience for managing the state of the connection.

Public Sub closeServerConnection()

Closes the server connection. This method is provided as a convenience for managing the state of the connection. It does not need to be called after executing a command.

Public Function computeWebFilePath(Data As String) As String

Computes the web file path from the serialized HDA and returns the value as a string (returns the serialized HDA containing StatusCode and StatusMessage). Returns the same data that was passed in as the parameter along with the additional value *WebFilePath* set in the local data containing the web file path.

Public Function computeNativeFilePath(Data As String) As String

Computes the native file path from the serialized HDA and returns the value as a string (returns the serialized HDA containing StatusCode and StatusMessage). Returns the same data that was passed in as the parameter along with the additional value *NativeFilePath* set in the local data containing the native file path.

Public Function computeURL(Data As String, IsAbsolute As Boolean) As String

Computes the URL path from the serialized HDA and returns the value as a string (returns serialized HDA containing StatusCode and StatusMessage). Returns the same data that was passed in as the parameter along with the additional value URL set in the local data containing the URL path.

OCX INTERFACE

The Stellent IntradocClient OCX component is used within a Windows Visual Basic development environment to gain access to the content and content management functions within Stellent Content Server. The OCX integration is designed to call services in a visual development environment, or to connect to a remote content server.

The IntradocClient OCX component provides functionality that you can access with a method call. Methods perform actions and often return results. Information is passed to methods using parameters. Some functions do not take parameters; some functions take one parameter; some take several.

The IntradocClient OCX component requires a username and password to execute the commands. The user must have the appropriate permissions to execute the commands. Some commands will require an administrative access level, other commands may require only write permission.

Outside of the `init` and `connection` managing methods, all methods use the serialized HDA format for communication. The returned serialized HDA format

string contains information about the success or failure of the command. The `StatusCode` will be negative if a failure occurs, and `StatusMessage` will indicate the error.

Note: Refer to the *IdcCommand Reference Guide* for additional information.

Note: Refer to the *IntradocClient OCX Reference Guide* for the IntradocClient OCX API specifications listing the properties, methods, and events.

OCX Integration Example

This integration example provides a description of the setup process for the IntradocClient OCX component.

IntradocClient OCX Component Setup

Follow these steps to set up the IntradocClient OCX component in the Microsoft Visual Basic development environment:

1. Select **Project—Components**.
2. Browse to the IntradocClient.ocx file on your system and click **Open**.

   The IntradocClient module is added to the Component Controls list.
3. Ensure that the check box for *IntradocClient ActiveX Control* module is enabled and click **OK**.

   An OCX control is placed in the list of controls.
4. Select the control and draw it on the Visual Basic form.
5. Add the following lines to create and initialize the control:

```
Set idcCmd=CreateObject("Idc.CommandX")
idcCmd.init("sysadmin", "c:\stellent\bin")
```

6. Use the Visual Basic development environment to create a visual interface. See Figure 4-1 for an example.

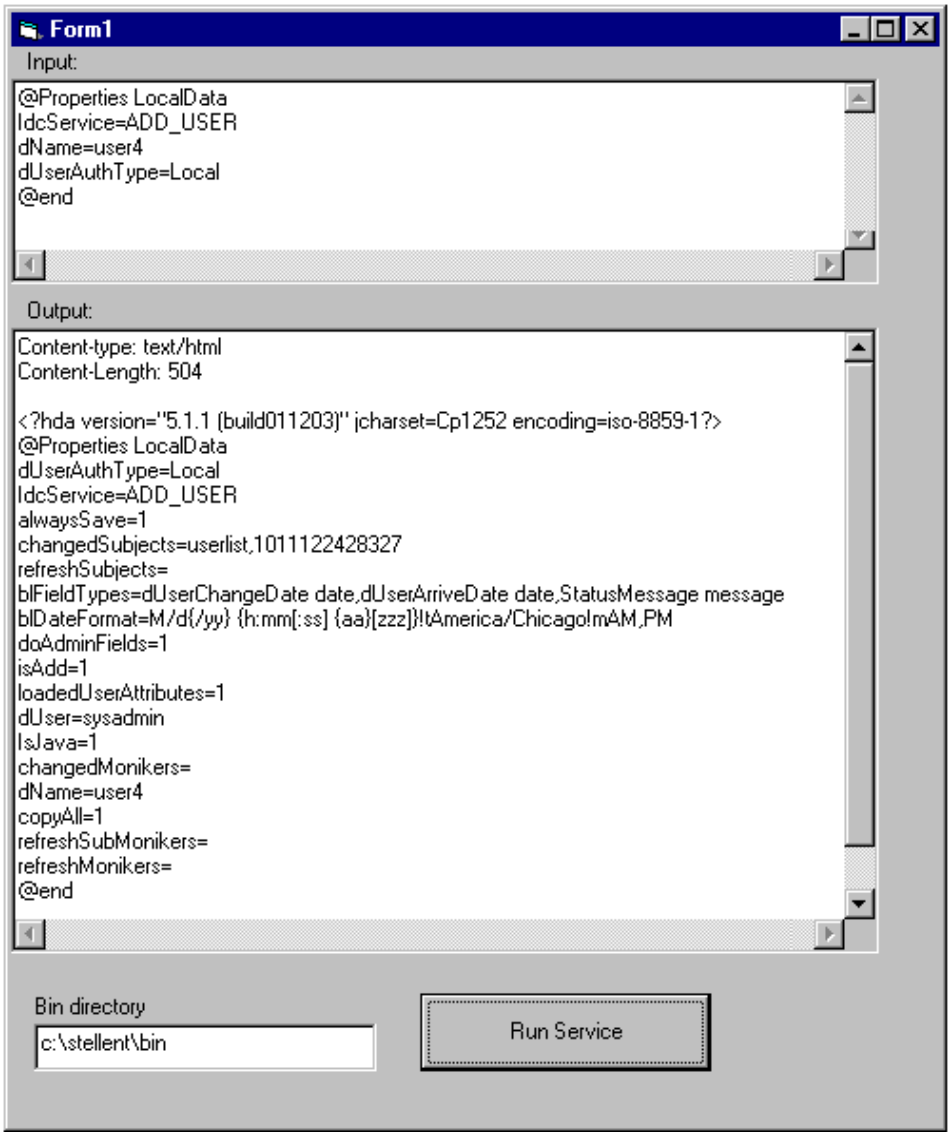

Figure 4-1 Visual Interface Example

INTEGRATION NOTES

Related Documentation

Refer to this documentation for addtional information:

- IntradocClient OCX Reference Guide
- IdcCommand—Java Command Utility Reference Guide

Chapter

5

JSP INTEGRATION

INTRODUCTION

Stellent Content Server core functionality can be accessed from a Java Server Page by any of these methods:

- Through the JSP Execution functionality using the Apache Jakarta Tomcat Server in a seamless integration with the Stellent Tomcat Integration JavaBean called *ServerBean.*
- Through the Stellent Content Server JavaBean called *IdcServerBean.*
- Through the Stellent Content Server Enterprise JavaBean (EJB) called *ContentServerBean.*

The Stellent JSP Tag Library is also provided to simplify Java Server Page development. By using the Stellent JSP Tag Library, application developers can focus on presentation issues rather than on how to access Stellent Content Server services.

Note: For information on portal servers, portals, and portlets, see *Chapter 6: EJB Integration*.

INTEGRATION METHODS

This section details the several methodologies that can be employed to access Content Server core functionality from a Java Server Page.

JSP Execution

The Stellent JSP Execution functionality uses the Apache Jakarta Tomcat Servlet/JSP Server in a seamless integration with the Stellent Tomcat Integration JavaBean to access the content and content management functions within Stellent Content Server. This JavaBean, customized for Tomcat Server, provides the public class *ServerBean* as the main entry point to interact with Stellent Content Server from a Java Server Page.

The Apache Jakarta Tomcat Server is a free, open-source server of Java Servlet and Java Server Pages that is run inside of Content Server when the feature is enabled. The integration of Tomcat Server with Stellent Content Server provides the benefit of increased performance for content delivery.

Using JSP Execution functionality enables developers to access and modify Stellent Content Server content, ResultsSets, personalization and security definitions, and predefined variables and configuration settings through Java Server Pages rather than through standard Stellent component architecture. Stellent services and IdocScript functions can also be executed from JSP pages which reside as executable content in the Content Server.

Important: JSP pages can execute IdocScript functions only when the JSP page is being server on the content server as part of the Stellent JSP Execution functionality. JSP pages served on a separate JSP server do not have this functionality. In those cases, checking a JSP page into the content server provides revision control but does not provide dynamic execution of IdocScript functions on the presentation tier (JSP server).

Note: Refer to the *Stellent JSP and JavaBean Guide* for additional information on JSP Execution functionality on Tomcat Server.

Note: Refer to the JavaDoc API in the Extras/JSPExecution/docs sub-directory on the Stellent Content Server CD-ROM for information on the Class/Interface, Field, and Method descriptions of the ServerBean.

Content Server JavaBean

You can access Stellent Content Server core functionality from a Java Server Page through the Stellent Content Server JavaBean called *IdcServerBean.* The IdcServerBean is accessed directly from a Java application or Java Server Page. The IdcServerBean provides a direct entry point to interact with the Content Server. This direct integration method using the IdcServerBean eliminates the need for a socket-based interface while still enabling access to all Stellent Content Server core capabilities.

Note: Refer to the *Stellent JSP and JavaBean Guide* for additional information on using Stellent Content Server JavaBean with a JSP Server.

Note: Refer to the JavaDoc API in the Extras/J2EE/docs sub-directory on the Stellent Content Server CD-ROM for information on the Class/Interface, Field, and Method descriptions of the IdcServerBean.

Enterprise JavaBean (EJB)

You can access Stellent Content Server core functionality from a Java Server Page through the Stellent Content Server Enterprise JavaBean (EJB) called *ContentServerBean.* You can deploy the ContentServerBean EJB in any J2EE compliant Enterprise JavaBean container such as WebLogic, WebSphere, or EAServer.

Note: For additional information on Enterprise JavaBean integration, see *Chapter 6: EJB Integration*.

Note: Refer to the *Stellent JSP and JavaBean Guide* for information on deploying the Stellent Content Server Enterprise JavaBean on a J2EE compliant application server.

Note: Refer to the JavaDoc API in the Extras/J2EE/docs sub-directory on the Stellent Content Server CD-ROM for information on the Class/Interface, Field, and Method descriptions of the ContentServerBean EJB.

JSP TAG LIBRARY

The Stellent JSP Tag Library is provided to simplify Java Server Page development. By using the Stellent JSP Tag Library, you can focus on presentation issues rather than being concerned with how to access Stellent Content Server services.

Understanding Tags

A JSP Tag is used on a JSP page to invoke a custom action. These custom actions encapsulate recurring tasks so that they can be reused across more than one application. A collection of custom tags is called a JSP Tag Library.

Tags from the Stellent JSP Tag Library reference a Stellent JavaBean as a wrapper to access the Stellent Content Server core functionality. Depending on the integration methodology used, this can be a JavaBean seamlessly integrated with Tomcat Server, a JavaBean providing a direct entry point to interact with the Content Server, or an Enterprise JavaBean (EJB) deployed on an application server.

Note: For additional information on integration methodologies, see *Integration Methods*, page 5-2.

Associated Reference Files

These associated files reference important naming and directory information:

- taglib.properties
- web.xml

The tag library properties file is a text file that stores the Uniform Resource Locator (URL) and the port number used by Stellent Content Server. These parameters are defaulted to *localhost* and *4444* respectively. This file is named *taglib.properties* and is located in the web application /WEB-INF/classes directory.

```
stellentURL=localhost
stellentPort=4444
```

Figure 5-1 Tag Library Properties File

The web application descriptor file is an XML file that defines the Uniform Resource Identifier (URI) and the location of the Tag Library Descriptor (.tld) file for the tag library. This file is named *web.xml* and is located in the web application /WEB-INF directory.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
 //DTD Web Application 2.3//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">

<web-app>
 <taglib>
    <taglib-uri>/stellent-taglib.tld</taglib-uri>
    <taglib-location>/WEB-INF/stellent-taglib.tld
    </taglib-location>
 </taglib>
</web-app>
```

Figure 5-2 Web Application Descriptor File

Declaring Tag Libraries

To use the Stellent JSP Tag Library from a Java Server Page, you must reference the tag library using one of these coding methods:

- Call the Tag Library Descriptor of an unpacked JSP Tag Library from a Java Server Page. For example:

  ```
  <%@ taglib uri="/WEB-INF/stellent-taglib.tld"
  prefix="scs" %>
  ```

- Call the JAR file containing the JSP Tag Library from a Java Server Page. For example:

  ```
  <%@ taglib uri="/WEB-INF/stellent-tags.jar"
  prefix="scs" %>
  ```

- Call the JSP Tag Library by reference. This requires editing the web application descriptor and defining a short name for the Java Server Page to use for referencing this JSP Tag Library. For example:

  ```
  <%@ taglib uri="SCSTags" prefix="scs" %>
  ```

The Uniform Resource Identifier (URI) attribute uniquely identifies the tag library. The URI can define the JSP Tag Library directly or define the JSP Tag Library by reference. If the URI is defined by reference, it must be mapped to an absolute location in the *taglib-location* element of a web application deployment descriptor (See figure 7-2). The prefix attribute defines the prefix that distinguishes tags provided by a given tag library from those provided by other tag libraries.

Tech Tip: Defining a reference to the Tag Library Descriptor file from the web application descriptor provides the convenience of not being required to change coded Java Server Pages at a later time if you decide to JAR the JSP Tag Library.

Calling the Tag Library

The Uniform Resource Identifier (URI) and the location of the Tag Library Descriptor (.tld) file is defined in the *web.xml* file (See figure 7-2). The Tag Library Descriptor (.tld) file defines each tag and the associated attribute.

To use the Stellent JSP Tag Library, code this entry on the selected Java Server Pages:

```
<%@ taglib uri="/stellent-taglib.tld" prefix="scs" %>
```

This entry points the JSP container to the location of the Stellent JSP Tag Library.

Calling the Tag Library by Reference

To call the Stellent JSP Tag Library by reference, you must edit the web application descriptor and provide a short name for the Stellent JSP Tag Library. This short name will be used on the Java Server Page to reference the Stellent JSP Tag Library.

Follow these steps to call the JSP Tag Library by reference:

1. In a text-only editor, open the *web.xml* file found in the web application /WEB-INF directory.
2. Define a short name for the JSP page to use for referencing the Stellent Tag Library (in this example, we have used SCSTags). Add the following lines just before the </web-app> entry:

   ```
   <taglib>
      <taglib-uri>SCSTags</taglib-uri>
      <taglib-location>/WEB-INF/stellent-taglib.tld
      </taglib-location>
   </taglib>
   ```

   Notice that the stellent-taglib.tld file is referenced rather than a JAR file.
3. Save the web.xml file.
4. Code this entry on the selected Java Server Pages:

   ```
   <%@ taglib uri="SCSTags" prefix="scs" %>
   ```

This entry points the JSP container to the location of the Stellent Tag Library. If the Tag Library Descriptor (.tld) file is packaged as a JAR file (for example, stellent-tags.jar), the tag library location entry could be changed to read */WEB-INF/stellent-tags.jar* without impacting the code on each Java Server Page.

INTEGRATION NOTES

Related Documentation

Refer to this documentation for addtional information:

- Stellent Java Server Page and JavaBean Guide
- Installing and Configuring Stellent Portlets on WebSphere Portal Server
- Installing and Configuring Stellent Portlets on WebLogic Portal Server
- Stellent EJB Integration for WebLogic Personalization Server

Chapter

6

J2EE API INTEGRATION

INTRODUCTION

Use the J2EE API to gain access to the content and content management functions within Stellent Content Server by deploying the Stellent Content Server Enterprise JavaBean on your J2EE compliant application server.

The Stellent Content Server Enterprise JavaBean can be deployed on any J2EE compliant application server such as BEA WebLogic, IBM WebSphere, Sybase Application Server, and iPlanet Application Server.

The Stellent Content Server Enterprise JavaBean provides robust connectivity to Stellent Content Server using the J2EE-EJB architecture. As a result, Stellent Content Server can be integrated into extended multiple-tier applications and Stellent services can be called from within a J2EE application server environment. While the Stellent Content Server Enterprise JavaBean is called from within the application server environment, it executes functions completely in the Stellent Content Server environment.

Note: For more information on Stellent Content Server Enterprise JavaBean deployment refer to the *Stellent JSP and JavaBean Guide*.

Note: For information on the Class/Interface, Field, and Method descriptions of the Content Server Enterprise JavaBean see the JavaDoc in the Extras/J2EE/docs sub-directory on the Stellent Content Server CD-ROM.

INTEGRATION NOTES

Portals and Portlets

Stellent has the capability to interface with any J2EE compliant portal server such as IBM's WebSphere Portal Server and BEA's WebLogic Personalization Server to populate portlets with content managed by Stellent Content Server.

Companies increasingly use portals, which provide secure, personalized points of access to a company's resources and information. Portlets (also called "portal applications") are dynamic components that end users see within their portal pages. They are basically small "windows" within the portal that serve as content channels. A portlet could, for example, show the latest headlines or provide personalized search capability. Figure 5-1 on the next page shows an example of a website with a number of individual portlets, each providing different, personalized pieces of information.

Note: For more information on integrating Stellent with portal servers see *Installing and Configuring Stellent Portlets on WebSphere Portal Server* and *Stellent WebLogic Personalization Server Integration Guide*.

Figure 6-1 Example Portlet Implementation

Stellent Content Server Enterprise JavaBean

The Stellent Content Server Enterprise JavaBean can be deployed on any J2EE compliant application server supporting EJB 1.1. Refer to the *Stellent JSP and JavaBean Guide* for deployment instructions concerning the following servers:

- BEA WebLogic® Server v.5.1 with Service Pack v.6 (or greater)

- BEA WebLogic® Server v.6.0
- IBM WebSphere® Server v.4.0 (or greater)
- Sybase EAServer® Enterprise Application Server v.3.6.1 (or greater)

The Stellent Content Server Enterprise JavaBean software and documentation is included on the Stellent Content Server CD-ROM in the *extras/J2ee* folder. Of particular interest:

- The Stellent Content Server Enterprise JavaBean (ContentServerBean) documentation is located in the *extras/J2ee/docs* sub-directory.
- The Stellent Content Server Enterprise JavaBean (ContentServerBean) JAR file (*IdcEjb.jar)* is located in the *extras/J2ee/ejb* sub-directory.
- The Stellent Content Server Enterprise JavaBean (ContentServerBean) Test Client is located in the *extras/J2ee/ejb* sub-directory.

Basic Deployment Steps

The deployment of the Stellent Content Server Enterprise JavaBean (ContentServerBean) requires several basic steps that are common to all application servers. While these steps may vary depending on the specific application server, these basic steps are still performed.

Important: Before deploying the ContentServerBean EJB ensure that the correct Java environment exists, and both the application server and Stellent Content Server are installed.

These basic steps are required for generation and deployment:

1. Copy the ContentServerBean to a location on your primary system.

2. Compile the Java classes and interfaces.
3. Generate the container classes. This creates the container-generated EJBObject implementing the Remote interface.
4. Run RMI against the Remote classes. This generates the Stub and Skeleton structure of the Remote Method Invocation architecture.
5. JAR the re-complied ContentServerBean along with the associated files (such as the deployment descriptor).
6. Deploy the re-compiled ContentServerBean to the application server.

Generally, an application server provides a utility or wizard to automate steps 2-6 or steps 3-5. The application server utility or wizard may perform these steps with a series of dialog screens or automatically when invoked. For example, each basic step in generation and deployment may be performed in this manner:

- Step two is often performed by the utility or wizard using the SDK `javac` Java compiler or a similar complier.
- Step three is performed by the utility or wizard by referencing the SDK `javax` extension classes to extend `EJBObject` for the Remote session interface and to extend `EJBHome` for the Home session interface.
- Step four is performed by the utility or wizard referencing the SDK `rmic` Remote Method Invocation compiler.
- Step five is often performed by the utility or wizard using the Sun `jar.exe` utility or a similar JAR utility
- Step six is often performed by the utility or wizard but may involve manually updating several application server specific files. For example, updating a file to reference the re-compiled ContentServerBean or to ensure that the JNDI name is properly defined.

Note: The `javax` extension classes, the `javac` compiler, the `rmic` compiler, and the `jar.exe` utility are all provided with Java 2 SDK 1.3.

Related Documentation

Refer to this documentation for addtional information:

- Stellent Java Server Page and JavaBean Guide
- Installing and Configuring Stellent Portlets on WebSphere Portal Server
- Installing and Configuring Stellent Portlets on WebLogic Portal Server
- Stellent EJB Integration for WebLogic Personalization Server

Chapter

7

CORBA INTEGRATION

INTRODUCTION

Use the Common Object Request Broker Architecture (CORBA) API to reference the Stellent Content Server Enterprise JavaBean and gain access to the content and content management functions within Stellent Content Server.

Stellent Content Server supports accessing the CORBA environment through our J2EE compliant Enterprise Java Bean (EJB). The CORBA distributed object architectures provide the network communications layer to the J2EE application server and the Stellent Content Server Enterprise JavaBean provides the connection to Stellent Content Server.

INTEGRATION NOTES

A CORBA integration is performed by implementing Remote Method Invocation (RMI) as the method-passing protocol and Internet Inter-ORB Protocol (IIOP) as the transport-level protocol. The IIOP protocol works across any TCP/IP implementation such as the Internet. Using these protocols (RMI over IIOP) the Stellent Content Server Enterprise JavaBean can be used to gain access to the Stellent Content Server.

Using a CORBA integration, the Stellent Content Server Enterprise JavaBean can be invoked with clients developed in various programming languages. For example:

- ActiveX clients
- CORBA C++ clients
- CORBA Java clients

The Stellent Content Server Enterprise JavaBean is accessed by using a proxy for the EJB Home interface, and calling business methods using a proxy for the EJB Remote interface.

Note: The Stellent Content Server Enterprise JavaBean (ContentServerBean) public interface *ContentServerHome* extends javax.ejb.EJBHome.

Note: Refer to the JavaDoc API in the Extras/J2EE/docs folder on the Content Server CD-ROM for information on the Class/Interface, Field, and Method descriptions of the Stellent Content Server Enterprise JavaBean (ContentServerBean).

CORBA Architecture

CORBA uses the Object Request Broker (ORB) architecture as the protocol for making requests. The ORB makes requests to objects and resides on the host between the data and the application layer.

The ORB handles the interactions between objects by performing object-oriented remote procedure calls. Requests are made on the data through the CORBA object, the ORB handles the message passing and accesses object data through the object definition.

- The ORB is the distributed service that implements the request to the remote object. It locates the remote object on the network, communicates the request to the object, waits for the results and communicates those results back to the client.
- The ORB implements location transparency. The same request mechanism is used by the client and the CORBA object regardless of where the object is located.
- The ORB implements programming language independence for the request. The client issuing the request can be written in a different programming language from the implementation of the CORBA object. The ORB does the necessary translation between programming languages. Language bindings are defined for all popular programming languages.

CORBA Interoperability

The ability for CORBA clients and servers to successfully interoperate with Enterprise JavaBeans (EJB) is defined by the EJB architecture and provided by the J2EE technologies. In addition to the RMI over IIOP method-passing and transport-level protocols, CORBA provides for these mapping methodologies.

- Mapping of IDL Interfaces
- Mapping of Naming Services
- Mapping of Transaction Services
- Mapping of Security Services

These mapping methodologies are discussed in more detail below.

Mapping of IDL Interfaces

This mapping defines how the CORBA Interface Definition Language (IDL) provides the constructs needed to specify interfaces. The interface defines the services that an object provides. These interfaces are mapped to language-specific interfaces. Mappings exist for a variety of languages, including Java, C, C++, COBOL, Ada, Smalltalk, TCL, Perl, Delphi, and Python.

- CORBA objects are accessed through the use of an interface.
- The CORBA IDL mapping defines interfaces, their attributes, methods, and parameters to those methods within the interface.
- The CORBA IDL interface for each object allows an object to interact by communicating object methods and parameters to other objects through the ORB.

- CORBA IDL interpreters map the IDL constructs to the corresponding programming language constructs. Table 8-1 shows how the IDL constructs are mapped to the Java programming language and the C++ programming language.

Table 7-1 IDL Construct Mapping

IDL	Java	C++
module	package	namespace
interface	interface	abstract class
operation	method	member function
attribute	pair of methods	pair of functions
exception	exception	exception

Mapping of Naming Services

This mapping defines how the CORBA naming services locate the EJB Home object. In a CORBA/EJB integration, the CORBA Common Object Service (COS) Naming Service resides on top of the ORB and the CORBA Interoperable Naming Service (INS) resides on top of the CORBA Naming Service.

- The CORBA COS Naming Service resides on top of the ORB as a standard CORBA object with a IDL interface. This allows client applications to obtain references to the object using a standard API.
- The CORBA INS is a URL-based naming system that allows applications to share a common initial naming context.

Application servers use the Java Naming and Directory Interface (JNDI) implementation layered on top of the CORBA INS. This architecture enables EJB/CORBA clients to access the naming service by using the JNDI API as required by the EJB specifications.

Mapping of Transaction Services

This mapping defines how EJB transactions map to the CORBA Object Transaction Service (OTS). An application server J2EE/EJB runtime environment uses an implementation of the CORBA OTS for transaction support. The J2EE Java Transaction Service (JTS) is the Java mapping for the IDL interfaces of CORBA OTS.

Mapping of Security Services

This mapping defines how EJB security maps to CORBA security. The application server controls access to the EJB by determine the identity of the client requesting access. The EJB/CORBA mapping specifies that the CORBA client ORB adds the client principal to each client request. This communication between the client and the server propagates the client identity to the server.

Related Documentation

Refer to this documentation for additional information:

- Stellent Java Server Page and JavaBean Guide
- IdcCommand—Java Command Utility Reference Guide

Chapter

8

ODMA INTEGRATION

INTRODUCTION

The Open Document Management Application Program Interface (ODMA) is a standard used to interface between desktop applications and file management software. Use the Stellent ODMA-based plug-in to gain access to the content and content management functions within Stellent Content Server (for ODMA-compliant desktop applications).

You can publish files to your Stellent Web repository directly from any ODMA-compliant application, such as Microsoft Word, Corel WordPerfect, and Adobe FrameMaker. With Stellent's Web centric adoption of ODMA, you can check in and publish information directly to the Web. This is a significant advancement over traditional ODMA client-server implementations, where information is published first to a server and is not immediately available on the Web for consumption.

INTEGRATION NOTES

This section discusses the Stellent ODMA Client, including the supported desktop applications, and the Stellent ODMA interfaces.

Stellent ODMA Client

The Stellent ODMA Client is used to check in and publish information directly to the Web from your desktop applications. Stellent ODMA Client surpasses traditional ODMA client–server models, which publish information to a server and not immediately to the Web for consumption.

You can use Stellent ODMA Client from within your desktop application to perform many tasks which interact with the Content Server.

For example:

- Save a file and immediately check it into the Content Server.
- Save a file to check in later to the Content Server.
- Check a file out of the Content Server.
- Update a file's Content Server metadata (content information).
- Save the file to your local file system and bypass the Stellent ODMA Client system.

Supported Desktop Applications

The Stellent ODMA Client can be used with the following ODMA-compliant desktop applications.

- Microsoft Word 97, 2000, and XP
- Microsoft PowerPoint 97, 2000 (with Service Release 1), and XP

- Microsoft Excel 2000 and XP (with the Stellent ODMA Client Excel plug-in)
- WordPerfect 8 (with Service Pack 7), except on Windows 2000
- Adobe FrameMaker 5.5 and 6.0 (with the FMBookMan plug-in)
- Microsoft Visio 5.5 and 2000
- Lotus WordPro 9
- Adobe Acrobat Capture 3.0

Note: If you install a new ODMA-supported application that you would like to use with ODMA, run "Update ODMA Apps" so Stellent ODMA Client recognizes the application. See the Stellent ODMA Client 5.0 Installation and Configuration Guide in the Documentation directory of the Stellent Desktop CD for instructions. You do not need to run the update when you upgrade a supported application.

Stellent ODMA Interfaces

This section describes accessing theses interfaces:

- Stellent ODMA Client Interface
- Stellent ODMA Desktop Shell Interface
- Stellent Content Server Interface

Stellent ODMA Client Interface

The Stellent ODMA Client - Select Document screen with the Recent Files option selected displays a list of files that you recently used through ODMA. This screen displays instead of the typical Open dialog box. If a file does not display on this screen, you can search for it in the Content Server or the local file system.

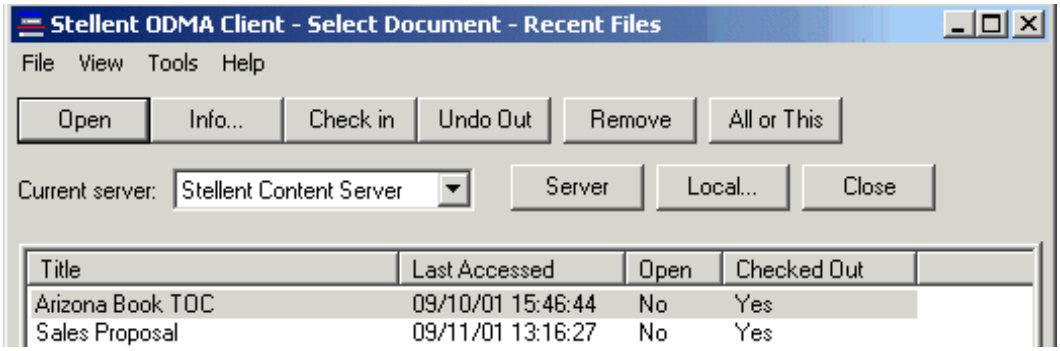

Figure 8-1 Stellent ODMA Client Select Document Screen

Stellent ODMA Desktop Shell Interface

The Stellent Client Desktop Shell provides "drag and drop" check-in functionality, and access to the Stellent ODMA Client - Select Document screen from outside of your desktop application.

Through the Desktop Shell, you can:

- Select a file from your desktop or a Windows Explorer window and drag it to the Desktop Shell to check it into the Content Server.
- Select and open a file from the Recent Files list or from the Content Server.

To access the Stellent Desktop Shell, from the Windows Start menu select *Programs—Stellent Client—Client*. Use this feature to access the Stellent ODMA Client program from outside of ODMA-compliant applications.

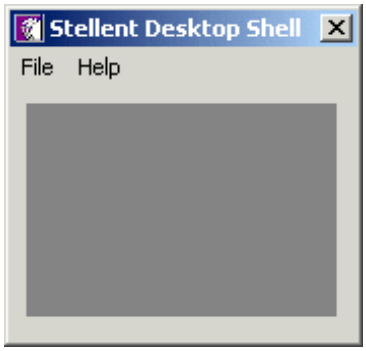

Figure 8-2 Stellent ODMA Client Desktop Shell

Stellent Content Server Interface

You can open and check out an ODMA file directly from the Stellent Content Server Content Information page if you wish. When you open a file from the Content Server, it opens in its native application so you can edit it and quickly check the file back into the Content Server.

Follow these steps to open a file from Stellent Content Server:

1. Open the Content Server as usual in your browser window.
2. Locate the file you want to work on, and click the Info button to open the Content Information page. A Security Warning dialog may display asking if you want to install and run a Java applet, distributed by Stellent. This applet enables the Check out and open feature. Click Yes to install and run the application and display the button, or No to cancel. If you click No, the Check out and Open button will not display during that browser session.

   If the file is checked in and you have ODMA on your machine (which is set up for that Content Server), Check out and Open displays as shown in Figure 9-2:

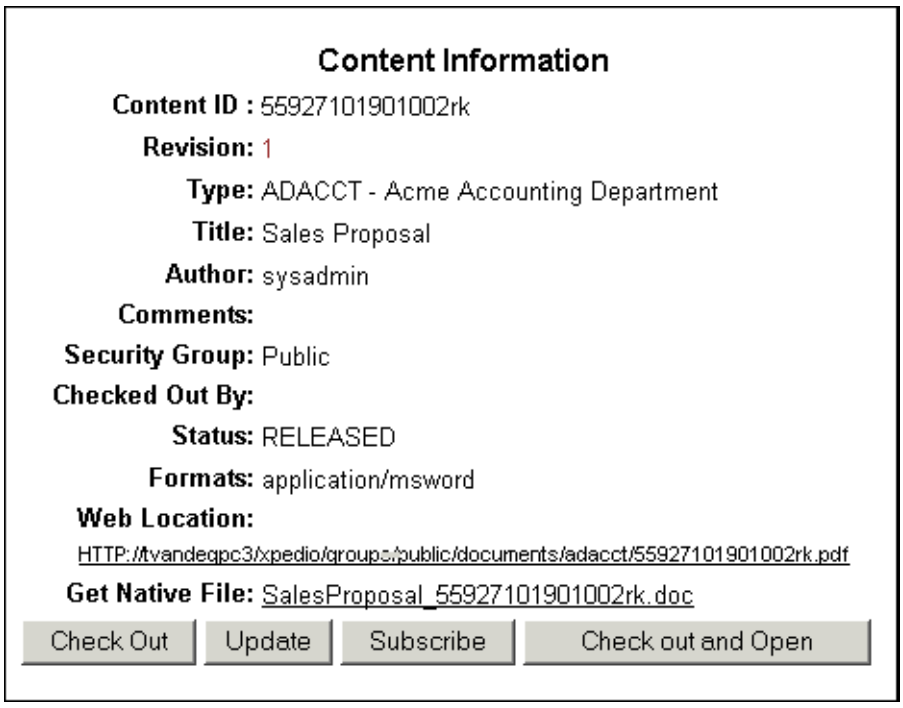

Figure 8-3 Stellent Content Server Content Information Screen

Note: You can use Check out and Open for all ODMA-compliant applications except Microsoft Excel, which you open through an ODMA-compliant application, the Select Document screen, or the Desktop Shell instead.

3. Click Check out and Open to open the file in its native application (if the application is installed on your machine).
4. Edit and save the file as usual.
5. When you close the file, the Content Check In Form displays:
   - To check the file into the Content Server, click Check In.

- To keep the file checked out so you can open it at a later time from the "Stellent ODMA Client - Select Document - Recent Files" screen, click Close. You may want to do this if you are in the process of editing a file when you need to close the document, and you only want to check the file in and create a revision in the Content Server when you complete your changes.

Note: You can also open and check out a file from within an ODMA-compliant application, and you can open a copy of a file instead of checking it out. Refer to the Stellent Online Help “ODMA Client Basics” for more information.

Related Documentation

Refer to this documentation for addtional information:

Intradoc Client OCX Reference Guide

Stellent Online Help (ODMA)

Stellent Online Help (ODMA / FrameMaker)

Chapter

9

SOAP INTEGRATION

INTRODUCTION

Use a Simple Object Access Protocol (SOAP) interface to access the content and content management functions within Stellent Content Server and to deploy your content management capabilities as a Web service. The SOAP messaging protocol integrates .NET servers, J2EE application servers, or other systems with Extensible Markup Language (XML) based interfaces.

Employing a SOAP integration provides a standardized interface for executing Stellent services using the Java API (IdcCommand) and provides Content Server managed XML and non-XML content.

Because SOAP uses the Hypertext Transfer Protocol (HTTP) for data transmission, it can be invoked across the Web and enables content to be accessible over a network in a platform-independent and language neutral way.

THE SOAP PROTOCOL

Employing a SOAP integration provides a standardized interface for executing Stellent services using the Java API (IdcCommand) and provides content server managed XML and non-XML content.

Because SOAP uses the Hypertext Transfer Protocol (HTTP) for data transmission, it can be invoked across the Web and enables content to be accessible over a network in a platform-independent and language-neutral way.

SOAP is an XML based messaging protocol consisting of these parts:

- an envelope that defines what is in a message and how to process it
- a set of encoding rules for defining application data types
- a convention for representing remote procedure calls and responses

Using SOAP to access content management capabilities as a Web service enables real-time programmatic interaction between applications and enables the integration of business processes and facilitates information exchange.

Web services are modular components that are contained in an XML wrapper and defined by the Web Services Description Language (WSDL) specifications. The Universal Description Discovery and Integration (UDDI) Web-based registry system is used to locate these services.

Tech Tip: While .NET servers support WSDL and integrate with the SOAP Toolkit, is must be specified that the SOAP packet is sending a Remote Procedure Call (RPC). The default is to evaluate SOAP messages as document-style rather than Remote Procedure Call (RPC) style SOAP messages. Using the SOAP Toolkit client with a .NET developed Web service returns an error reading the WSDL document. To permit the SOAP Toolkit to read the generated WSDL and call your .NET Web service, the `SoapRpcService()` attribute must be specified in your Web service class.

Note: The Stellent SOAP Component is located in in the Extras/Soap sub-directory on the Stellent Content Server CD-ROM.

Note: Refer to *Using SOAP to Connect to Stellent Content Server* for SOAP Component installation steps.

INTEGRATION NOTES

Stellent SOAP Clients

The Stellent SOAP Clients include a Visual Basic client and three Java programs. These tools are provided with the installed SOAP component.

For example (Visual Basic SOAP Client):

```
C:/stellent/custom/Soap/VBSoapClient/
```

For example (Java SOAP Clients):

```
C:/stellent/custom/Soap/JavaSamples/
```

Visual Basic SOAP Client

A Visual Basic SOAP Client is provided with the SOAP component. The sample Visual Basic SOAP Client is for the testing of basic services only. To check in a content item or retrieve a content item from the content server, use the Java SOAP Client.

To run the Visual Basic SOAP Client, you must have Microsoft Visual Studio installed or the provided MSINET.OCX file registered.

Note: The ZIP file for the SOAP component contains a number of sample files with pre-defined services.

Note: Refer to *Using SOAP to Connect to Stellent Content Server* for additional

information on using the Visual Basic SOAP Client.

Java SOAP Client

These Java SOAP programs are provided with the SOAP component:

Java Program	Description
SoapClient	This Java program allows you to call services in the content server using the SOAP interface.
SoapClientUpload	This Java program allows you to upload files to the content server using the SOAP interface.
SoapClientDownload	This Java program allows you to download files from the content server using the SOAP interface.

Note: A number of sample XML files with pre-defined services are provided. In many cases, these services will need to be edited to meet your particular application.

Note: Refer to *Using SOAP to Connect to Stellent Content Server* for additional information on using the Java SOAP Client.

Simple Soap Request

A SOAP request is a XML-based Remote Procedure Call (RPC) sent using the HTTP transport protocol. The payload of the SOAP packet is an XML document that specifies the call being made and the parameters being passed.

The Stellent Soap component allows a user to call an IdcService on a Content Server by constructing a Soap-XML formatted request and pass this via HTTP to the Content Server. The Content Server will then pass back a response in Soap-XML format.

Note: Refer to *Using SOAP to Connect to Stellent Content Server* for additional information and sample SOAP service calls.

The PING_SERVER service evaluates whether a connection to the server exists.

- Returns status information.
- If this service is unable to execute, this message is displayed to the user: *Unable to establish connection to the server.*

Tech Tip: Execute a PING_SERVER request before calling other services to ensure that there is a connection to the content server and that you are logged in as a user authorized to execute commands.

Note: Refer to the *IdcCommand Reference Guide* for a list of avalable services and the required parameters.

SOAP Request

```
<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<idc:service
xmlns:idc="http://www.stellent.com/IdcService/"
IdcService="PING_SERVER">
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
<?xml version='1.0' ?>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<idc:service
xmlns:idc="http://www.stellent.com/IdcService/"
IdcService="PING_SERVER">
<idc:document>
<idc:field name="changedSubjects">

</idc:field>
<idc:field name="refreshSubjects">

</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="StatusMessage">
You are logged in as &#39;sysadmin&#39;.
</idc:field>
<idc:field name="changedMonikers">

</idc:field>
<idc:field name="refreshSubMonikers">

</idc:field>
<idc:field name="refreshMonikers">

</idc:field>
```

```
</idc:document>
<idc:user dUser="sysadmin">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Related Documentation

Refer to this documentation for addtional information:

- Using SOAP to Connect to Stellent Content Server
- Stellent Java Server Page and JavaBean Guide

Chapter

10

WEBDAV INTEGRATION

INTRODUCTION

Use a Web-Based Distributed Authoring and Versioning (WebDAV) enabled desktop or business application to connect directly to the Stellent Content Management system and avoid the need for additional client-side software.

The WebDAV component provides a way to remotely author and manage your Stellent content using clients that support the WebDAV protocol. For example, you can use Microsoft Windows Explorer or Microsoft Office applications to check in, check out, and modify content in the Stellent repository rather than using Stellent's web browser interface.

INTEGRATION NOTES

WebDAV server functionality in Stellent Content Server is implemented using Java Servlets technology. To perform a WebDAV integration, you will need either an HTTP Server with support for Java Servlet technology, or a Servlet engine with HTTP Server capabilities.

Most Servlet engines available today are capable of providing at least basic HTTP server capabilities. Using a Servlet engine in “standalone” configuration eliminates the need for a standard HTTP server such as Apache, Netscape iPlanet, or Microsoft IIS. However, you should consider that standalone Servlet engines may be suitable only for:

- Testing WebDAV Server support for SCS
- Serving a few thousand requests per day
- Serving content when the server is behind a firewall

Note: We recommend that you perform internal site tests before deciding on a configuration. Tomcat sometimes delivers better performance as a standalone configuration.

For all other situations, you should consider using an HTTP server such as Apache, iPlanet, or IIS in front of the Servlet engine. These HTTP servers are much faster in handling connections and doing basic processing on URLs, and their IO is well optimized. Such HTTP servers may also provide the performance, scalability, reliability, and security mechanisms you need when your server is on the Internet or in a production environment with a large user base.

Note: Regarding security mechanisms, you must install the Servlet engine to the same web server that the content server uses to get the benefit of the security filter for NT security

System Configurations

WebDAV support can be set up in a number of different ways, depending on your content server configuration and available hardware. This section describes typical integration scenarios:

- Standalone Configuration
- HTTP Server with Servlet Engine Configuration
- iPlanet HTTP Server Configuration

Standalone Configuration

A standalone configuration requires only the Stellent Content Server and a Servlet engine with a WebDAV server—a separate HTTP server is not required. There are two typical hardware configurations for a standalone system:

- Single box: Both the Content Server and the Servlet engine are installed on the same machine (see Figure 11-1).
- Separate boxes: The Content Server and Servlet engine are installed on different machines (see Figure 11-2).

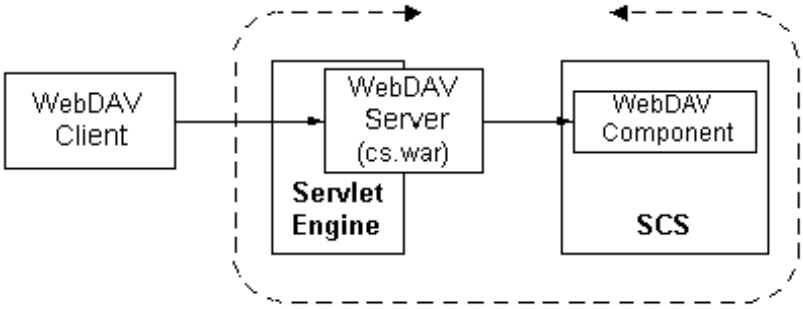

Figure 10-1 Standalone System on a Single Box

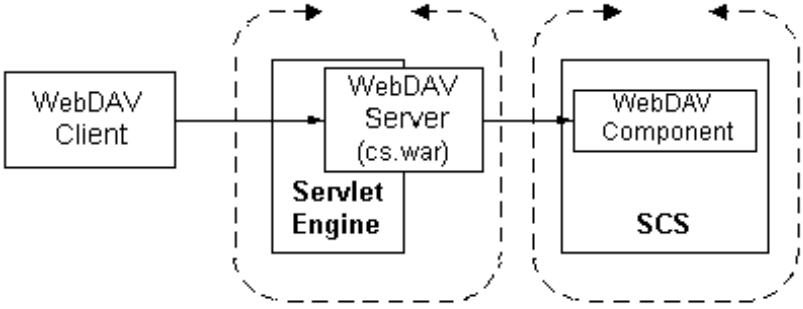

Figure 10-2 Standalone system on Separate Boxes

HTTP Server with Servlet Engine Configuration

Note: Refer to the *WebDAV Installation Guide* for additional information.

An HTTP Server with Servlet Engine configuration includes an HTTP server as an interface between the client and the Servlet engine. There are two typical hardware configurations for an HTTP server with Servlet engine integration:

- One box: The Content Server, HTTP server, and Servlet engine/WebDAV server are installed on the same machine (see Figure 11-3).
- Two boxes: The Content Server is installed on one machine and the HTTP Server and Servlet Engine/WebDAV Server are installed on another machine (see Figure 11-4).

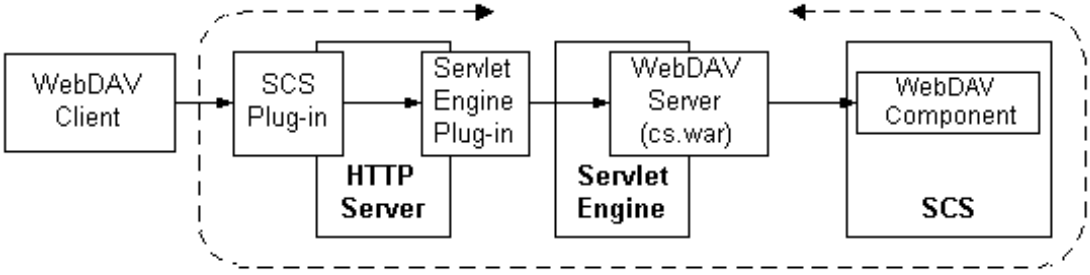

Figure 10-3 HTTP Server and Servlet Engine on Same Box

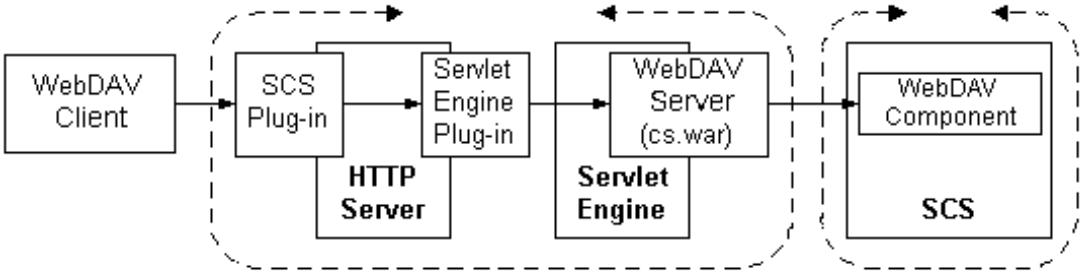

Figure 10-4 HTTP Server and Servlet Engine on Separate Box

The HTTP Server with Servlet Engine integration configurations listed in the following table have been tested (see Table 11-1).

Table 10-1 Tested HTTP Server with Servlet Engine Configurations

HTTP Server	Servlet Engine	Servlet Engine Plug-in for HTTP Server	Stellent Plug-in for HTTP Server
Apache 1.3.1x	Tomcat 3.2.x/3.3	mod_jk	N/A
iPlanet	JRun 3.1	JRun plug-in	Win32: ns_rqheader.dll Solaris: ns_rqheader.so
IIS 5.0	Tomcat 3.2.x/3.3	iis_redirect.dll	iis_translate.dll
IIS 5.0	JRun 3.1	JRun plug-in	iis_translate.dll

Note: Servlet engine plug-ins are installed during Servlet engine installation. Plug-ins required for SCS WebDAV support are installed separately.

iPlanet HTTP Server Configuration

If an iPlanet 6.0 HTTP server is used, a separate Servlet engine is not required.

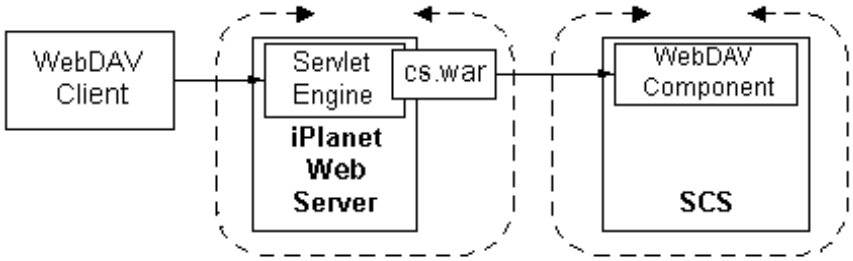

Figure 10-5 iPlanet HTTP Server Integration

Integration Requirements

The following items must be installed and configured for Stellent Content Server to provide WebDAV support:

- Stellent Content Server (SCS) with WebDAV component
- Servlet engine with WebDAV server
- HTTP server (optional)
- WebDAV client

Note: Refer to the *WebDav Installation Guide* for complete installation instructions.

This table lists the specific software that must be installed for Stellent Content Server to provide WebDAV support.

Table 10-2 WebDAV Integration Requirements

System	Required Software
Content Server	❖ Stellent Content Server 5.0 or higher ❖ Stellent WebDAV component
Servlet Engine	❖ WebDAV server (cs.war file) ❖ JAXP API v1.1 (Java API for XML Processing) ❖ Servlet 2.2-compliant Servlet engine, such as one of the following: • Apache Tomcat 3.2 and higher • Allaire JRun 3.1 • New Atlanta ServletExec 4.0 • iPlanet Web Server 6.0
HTTP Server (Optional)	One of the following web servers with the WebDAV plug-in installed: • Apache 1.3.1x • IIS 5.0 • iPlanet 6.0 • Netscape Enterprise Server

System	Required Software
Client Machine	Microsoft Windows 2000 or Microsoft Windows NT operating system and at least one of the following software packages: • Microsoft Office 2000 (SP1 or later) • Microsoft Office XP • Microsoft Internet Explorer 5.x with the Web Folders option enabled (this option is enabled during a “full” install of IE 5.x) One or more WebDAV client applications: • Microsoft Windows Explorer • Microsoft Word 2000 or XP • Microsoft Excel 2000 or XP

Note: Windows 95 and Windows 98 will support WebDAV functionality through Windows Explorer if Internet Explorer 5.x is installed with the Web Folders option enabled. Versions of client applications earlier than Office 2000 and/or other browser versions may not work with these client operating systems

Note: Stellent WebDAV has been tested using Tomcat, JRun, and ServletExec Servlet engines. However, these Servlet engines are not supported as part of Stellent Content Server, and the results with other Servlet engines are unknown.

Related Documentation

Refer to this documentation for addtional information:

- Folders Component Installation Guide
- WebDAV Support Installation Guide
- README.html (for WebDAV)

Index

T

U

V

W

X