Using WSDL Generator and SOAP
10*g* Release 3 (10.1.3.3.1)

May 2010

ORACLE

Using WSDL Generator and SOAP, 10*g* Release 3 (10.1.3.3.1)

Contributing Authors: Jean Wilson

Contributors: Scott Nelson, Sam White

Table of Contents

Chapter

1

OVERVIEW

OVERVIEW

The information contained in this guide is based on Content Server 7.5. 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, Oracle, Inc.. cannot warrant compatibility with all versions and features of third-party products.

This chapter contains these topics:

ABOUT THIS GUIDE

This guide discusses using Web Services Definition Language (WSDL) files and SOAP (Simple Object Access Protocol) to manage Content Server. SOAP is a lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network. The WSDL Generator component allows for creating WSDLs for the services of the Content Server. Users can then take the WSDLs and plug them into APIs to create web services that can be used with the Content Server.

Some SOAP functionality has been built into the core Content Server as of the 7.5 release. The WSDL Generator component is not essential to use SOAP; administrators can still

write or call Content Server service calls in SOAP if needed. The WSDL Generator provides flexibility in altering existing client applications.

Content Server has a WSDL 1.1 implementation that exposes the Content Server IDCService (Internet Distributed Content Service), which in turn extends all of the capabilities of the Unversal Content Management. Using the IDCService, content can be checked out and checked in, workflows can be created, run and approved, content can be made available for publishing, and content can be searched by category (metadata), content (full-text), or a combination of both.

You can use WSDL files to map to Content Server and the SOAP to access the content and content management functions within Content Server and to deploy your content management capabilities as a web service.

Important: The WSDL Generator component is available on version 7.5 and later of the Content Server. It is not supported on earlier versions. Customers needing SOAP and Web Services functionality on earlier versions should use the SOAP component, version 7.1.

WEB SERVICES AND CONTENT MANAGEMENT

These web services integration technologies are provided to access the functionality of the Content Server:

- WSDL Generator Component:
  - Includes custom WSDL files.
  - Sample applications in Java, VB and C#
- WSDL Files:
  - CheckIn.wsdl
  - DocInfo.wsdl
  - GetFile.wsdl
  - MetaData.wsdl
  - PortalInfo.wsdl
  - Search.wsdl
  - Subscription.wsdl
  - Workflow.wsdl
- Ability to expose any Content/Collaboration Manager service as a web service

Web services provide an open architecture that can be used to integrate the content server. Exposed services within the Unversal Content Management system (search, check-in, file

retrieval, content information) are available to be used as web services out-of-the-box. Oracle's service-based architecture also enables you to expose any core or custom internal service as a web service.

Audience

This guide is intended for application developers who need to access content server services using WSDL and SOAP. It is assumed that the reader is familiar with network protocols, the eXtensible Markup Language (XML), and a programming language such as Java or C#.

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 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.
	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.

Chapter

2

USING WEB SERVICES

OVERVIEW

This chapter provides an overview of web services, general information on WSDL files, and general information on the SOAP protocol and installation information.

This chapter covers the following topics:

WEB SERVICES FRAMEWORK

Web services reside as a layer above existing software systems such as application servers, .NET servers, and the content server. Web services can be used as a bridge to dissimilar operating systems or programming languages. Web services are adapted to the Internet as the model for communication and rely on the HyperText Transfer Protocol (HTTP) as the default network protocol. Thus, using web services, you can build applications using a combination of components.

The core enabling technologies for web services are XML, WSDL, SOAP, and UDDI. This section provides a general overview of these technologies.

XML: Data

The eXtensible Markup Language (XML) is a bundle of specifications that provides the foundation of all web services technologies. Using the XML structure and syntax as the foundation allows for the exchange of data between differing programming languages, middleware, and database management systems.

The XML syntax incorporates instance data, typing, structure, and semantic information associated with data. XML describes data independently and also provides information for mapping the data to software systems or programming languages. Because of this flexibility, any software program can be mapped to web services.

When web services are invoked, the underlying XML syntax provides the data encapsulation and transmission format for the exchanged data. The XML elements and attributes define the type and structure information for the data. It is XML that provides the capability to model data and define the structure specific to the programming language (e.g, Java, C#, Visual Basic, etc.), the database management system, or software application. web services use the XML syntax to specify how data is represented, how the data is transmitted, and how the service interacts with the referenced application.

WSDL: Interface

The Web Services Description Language (WSDL) provides the interface that is exposed to web services. It is the WSDL layer that enables web services to be mapped to underlying programs and software systems. A WSDL file is an XML file that describes how to connect to and use a web service.

SOAP: Communication

The Simple Object Access Protocol (SOAP) provides communication for web services interfaces to communicate to each other over a network. SOAP is an XML-based communication protocol used to access web services. Web services receive requests and return responses using SOAP packets encapsulated within an XML document.

UDDI: Registry

The Universal Description Discovery and Integration (UDDI) service provides registry and repository services for storing and retrieving web services interfaces. UDDI is a public or private XML-based directory for registration and lookup of web services.

Content Server currently does not publish to any public or private UDDI sources. However this does not prevent users from integrating Content Server with other applications using web services.

DIME: Message format

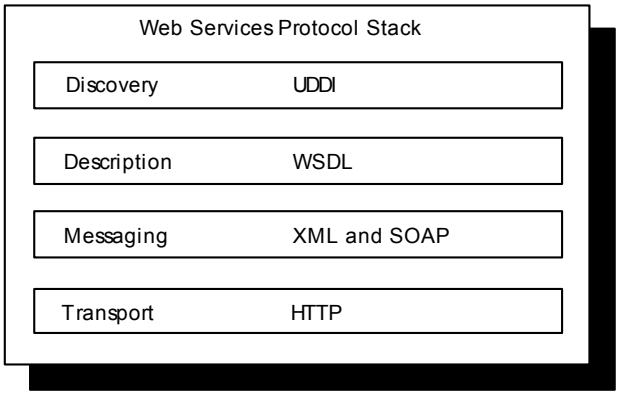
DIME is a lightweight, binary message format that can be used to encapsulate one or more application-defined groups of arbitrary type and size into a single message construct. This format can be used when uploading or downloading content. The payloads consist of the SOAP message and one or more groups of file content.

How the Enabling Technologies Work Together

The XML, WSDL, SOAP, and UDDI technologies work together as layers on the web services protocol stack. The web services protocol stack consists of these layers.

- The service *transport* layer between applications (HTTP).
- The *messaging* layer that provides a common communication method (XML and SOAP).
- The service *description* layer that describes the public interface to a specific web service (WSDL).
- The service *discovery* layer that provides registry and repository services for storing and retrieving web services interfaces (UDDI).

Figure 2-1 Web service protocol stack



Note: While several protocols are available as a transport layer (e.g., HTTP, SMTP, FTP, BEEP), the HTTP protocol is most commonly used. The WSDL Generator component relies on the HTTP protocol as the transport layer.

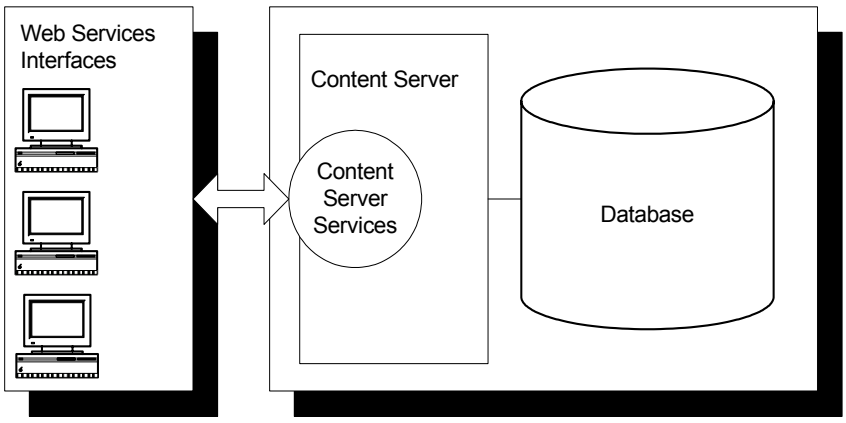
To help grasp the connection between these technologies, consider this simple analogy: Think of HTTP as the telephone wire (transport between applications) and UDDI as a telephone book (where a developer can browse a UDDI registry to locate a registered service). SOAP could be described as the voices of the people talking on the telephone (the exchange of information), and XML as the language they are speaking in (the underlying structure for the exchange of data). To continue with the telephone analogy, WSDL would be the phone number that calls a specific web service (of course, WSDL is more than just a "telephone number" as it includes information such as the available functions and data types).

IMPLEMENTATION ARCHITECTURES

Web services are not executable, but rather exchange data within the development environment. Thus, web services are a means to exchange information with an application server or software package that is performing the communication between the programs exchanging data.

The following scenario provides a web services implementation architecture for the content server application. The primary value of this architecture remains in the features and functions of the content server. Web services access the content server via the WSDL Generator component and use the exposed Content Server services to execute actions and provide data transaction between the user employing web services and the content server.

Figure 2-2 Web service implementation architecture



IMPLEMENTATION ON .NET

The Microsoft .NET products including the .NET platform, .NET Framework, and Visual Studio .NET all support the XML schema, WSDL, and SOAP specifications:

- The .NET platform is a designed as a programming model that enables developers to build XML web services and applications. The platform provides a set of servers that integrates, executes, and manages XML web services and applications.
- The .NET Framework product enables developers to build and deploy web services and applications. It provides a structured environment for integrating web services and consists of a common language run-time, unified class libraries, and includes the ASP .NET server.
- The Visual Studio .NET product provides the tools for developers to write application software according to the XML-based web service specifications.

Using the .NET architecture, development and deployment of a web service are integrated as a single step. Because every program written in a .NET language is designed to function as a web service, the .NET server is able to easily create and deploy the program as a web service.

THE SOAP PROTOCOL

Employing a SOAP integration provides a standardized interface for executing Content Server 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 enable 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, it 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, you must specify the `SoapRpcService()` attribute in your web service class.

WSDL GENERATOR COMPONENT INSTALLATION

This section provides information on how to install the WSDL Generator component. The WSDL Generator component enables 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 passes back a response in SOAP-XML format.

The WSDL Generator component can be installed on your Content Server instance using the Component Wizard or the Component Manager.

Installation Using Component Manager

Follow these steps to install the component using the Component Manager:

1. Select **Admin Server** from the Administration Menu.

   The Content Admin Server page is displayed.

2. Click on the instance name of the server where the component will be installed.

   The Content Server <*instance-name*> page is displayed.

3. Click **Component Manager**.

   The Component Manager page is displayed.

4. Click the **Browse** button and find the zip file that was saved previously.

5. Highlight the component name and click **Open**.

6. Click **Install**. A message is displayed, detailing what will be installed.

7. Click **Continue** to continue with installation or **Cancel** to stop installation.
8. If you select **Continue**, a message appears after successful installation. You can choose one of two options:
   - To enable the component and restart the Content Server. You are returned to the Content Server <*instance-name*> page, where you can restart the server.
   - To return to the Component Manager, where you can continue adding components. When done, highlight the components you want to enable and click **Enable**. When finished enabling components, return to the Content Server <*instance-name*> page and restart the server.

Installation Using Component Wizard

Follow these steps to install the component using the Component Wizard:

1. Start the Component Wizard on Windows systems by selecting **Start**—**Programs**—**Content Server**—<*install_dir*>—**Utilities**—**Component Wizard**. On UNIX systems, navigate to the <*install_dir*>/bin directory and run the ComponentWizard.exe file.

   The Component Wizard main screen and the Component List screens are displayed.
2. On the Component List screen, click **Install**.

   The Install screen is displayed.
3. Click **Select**.
4. Navigate to the zip file that was saved previously and select it.
5. Click **Open**.

   The zip file contents are added to the Install screen list.
6. Click **OK**. You are prompted to enable the component.
7. Click **Yes**. The component is listed as enabled on the Component List screen.
8. Exit the Component Wizard.
9. Restart the Content Server.

Chapter

3

SOAP CLIENTS

OVERVIEW

Note: Users developing SOAP client implementations must ensure that "chunking" is disabled in their client API code.

You can use SOAP to access the content and content management functions within the content server and to deploy your content management capabilities as a web service.

This chapter discusses the SOAP packet format used to call Content Server services and describes how to use the provided SOAP Clients to execute those services.

This section contains these topics:

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

- For example (Visual Basic SOAP Client):
  `<install_dir>/custom/WsdlGenerator/samples/VisualBasic/`
- For example (Java SOAP Clients):
  `<install_dir>/custom/WsdlGenerator/samples/Java/`

Note: See the *Services Reference Guide* for a list of available services and the required parameters. See Chapter 4 (*Soap Service Calls)* for additional information on defining services and parameters in an XML document.

USING THE VISUAL BASIC SOAP CLIENT

This section describes the Visual Basic SOAP Client provided with the installed WSDL Generator 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. The MSINET.OCX file is located in the <*install_dir*>/custom/WsdlGenerator/samples/VisualBasic/ directory.

Follow these steps to register the OCX and use the VBSoapClient:

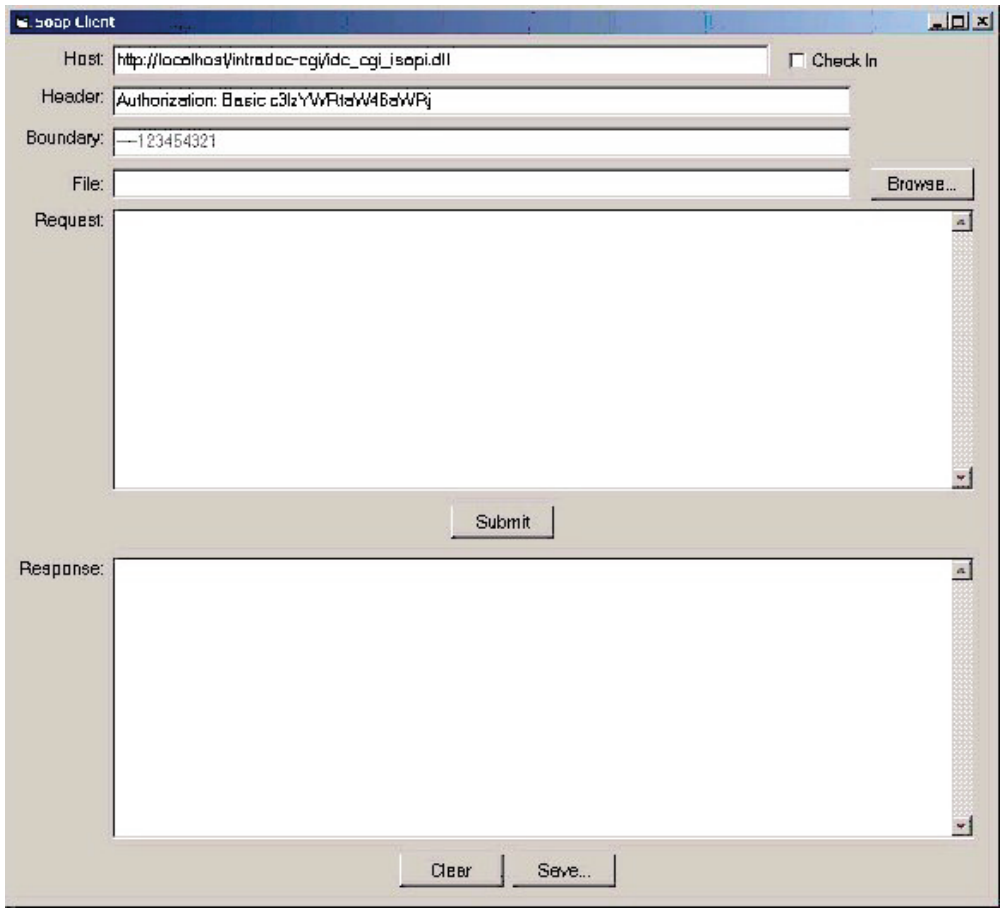
1. From a command prompt, define the complete path to the MSINET.OCX file and run the command to register the OCX:

   ```
   regsvr32 <install_dir>/custom/WsdlGenerator/samples/VisualBasic/msinet.ocx
   ```

2. Launch the SoapClient.exe executable file.

   The Soap Client interface is displayed.

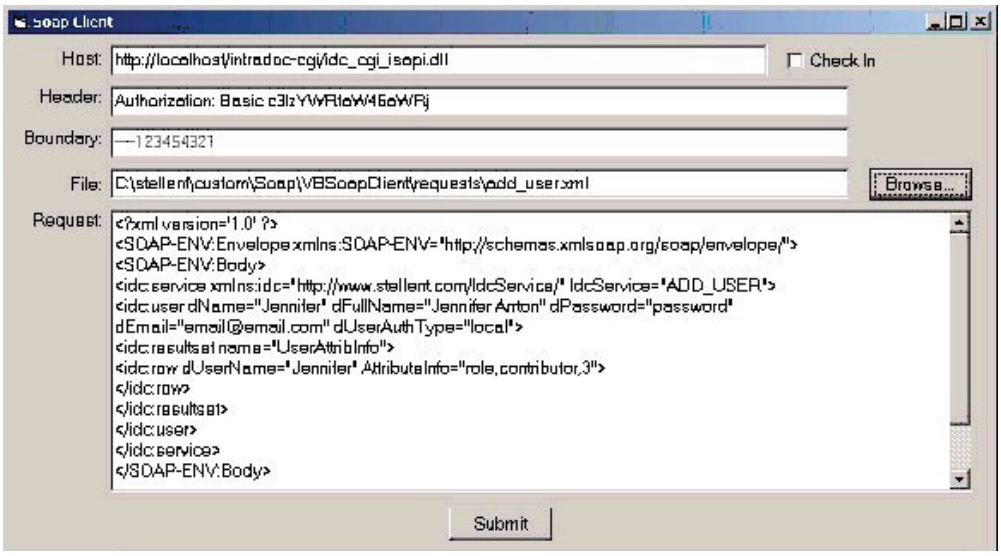
Figure 3-3 Soap Client interface



3. In the Request text area, define a SOAP packet in an XML-formatted document.
   - ❖ Define the content server service to execute and the required (and optional) parameters.
   - ❖ Alternatively, you can open an XML document containing service and parameter information by clicking **Browse** and navigating to the file.

Note: The unpackaged ZIP file for the WSDL Generator component contains a number of sample files containing pre-defined services. These files are located in the <*install_dir*>/custom/WsdlGenerator/samples/VisualBasic/requests/ directory.

Figure 3-4 Soap Client definition



4. Click **Submit** to execute the command.

USING THE JAVA SOAP CLIENT

This section references the three Java SOAP programs provided with the installed WSDL Generator component. These are the provided sample Java programs stored in the <*install_dir*>/custom/WsdlGenerator/samples/Java/ directory:

Java Program	Description
SoapClient (page 3-5)	Enables you to call services in the content server using the SOAP interface. The files for this program are stored in the /SoapClient/ directory.
SoapClientUpload (page 3-7)	This Java program enables you to upload files to the content server using the SOAP interface. The .java and .class files for this program are stored in the /SoapClientUpload/ directory.
SoapClientDownload (page 3-8)	This Java program enables you to download files from the content server using the SOAP interface. The .java and .class files for this program are stored in the SoapClientDownload/ directory.

Note: Each directory contains a number of sample XML files with pre-defined services. In many cases, you will need to edit these services to meet your specific needs.

These are the command line parameters used as arguments for the Java programs:

Parameters	Description
-c <*config file*>	The configuration file containing server settings (host, port, etc.).
-x <*xml file*>	The XML file containing the SOAP request to pass to the content server.
-p <*primary file*>	The filename to upload as the primary file.
-a <*alternate file*>	The filename to upload as the alternate file (optional).
-l <*log file*>	The filename containing the request and response data (optional).

For example, you would run SoapClientUpload from the command line by referencing the Java program, and defining the command line parameters:

```
SoapClientUpload -c soap.cfg -x checkin_universal.xml
-p soaptest.doc -a soaptest.pdf -l logfile.txt
```

The complete directory path to the file must be specified if the referenced file is not located in the same directory as the Java program executed.

SoapClient

Services such as PING_SERVER, ADD_USER, and DOC_INFO can be executed using this program.

Define the Configuration File Properties

Define the correct configuration properties for your content server instance. Specify a User/Password with the appropriate permissions to execute the desired command. Many commands require administrative access:

```
Host=testserver
Port=80
CgiPath=/stellent/idcplg
```

```
User=sysadmin
Password=idc
```

Define the XML File Properties

Define the content server service to execute and the required (and optional) parameters as a SOAP packet in an XML-formatted document. This XML document is referenced from the command line when the command is executed.

```
<?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.oracle.com/IdcService/" IdcService="DOC_INFO">
<idc:document dID="6">
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Run the Command

Execute the service from the command line by changing to the *SoapClient/* directory, referencing the Java program, and defining the command line parameter:

```
<install_dir>/custom/WsdlGenerator/samples/Java/SoapClient>
java SoapClient -c soap.cfg -x doc_info.xml -l logfile.txt
```

SoapClientUpload

Use the CHECKIN_UNIVERSAL service (or related service) to perform a content server controlled check in.

Define the Configuration File Properties

Define the correct configuration properties for your content server instance. Specify a User/Password with the appropriate permissions to execute the desired command. Many commands require administrative access:

```
Host=testserver
Port=80
CgiPath=/stellent/idcplg
User=sysadmin
Password=idc
```

Define the XML File Parameters

Define the content server service to execute and the required (and optional) parameters as a SOAP packet in an XML-formatted document. This XML document is referenced from the command line when the command is executed:

```
<?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.oracle.com/IdcService/"
IdcService="CHECKIN_UNIVERSAL">
<idc:document dDocName="SoapUpload2" dDocAuthor="sysadmin" dDocTitle="Soap
Upload 2 Document" dDocType="ADACCT" dSecurityGroup="Public" dDocAccount="">
<idc:file name="primaryFile"
href="C:/stellent/custom/WsdlGenerator/samples/Java/SoapClientUpload/soaptest.
doc">
</idc:file>
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Run the Command

Execute the service from the command line by changing to the SoapClientUpload/ directory, referencing the Java program, and defining the command line parameters:

```
<install_dir>/custom/WsdlGenerator/samples/Java/SoapClientUpload>
java SoapClientUpload -c soap.cfg -x checkin_universal2.xml -p
```

```
<install_dir>/custom/WsdlGenerator/samples/Java/SoapClientUpload/
soaptest.doc
```

SoapClientDownload

Use the GET_FILE service to return a specific rendition of a content item, the latest revision, or the latest released revision.

When the GET_FILE service is executed the content item requested (defined in the XML file) is saved to the current directory.

For example:
`<install_dir>/custom/WsdlGenerator/samples/Java/SoapClientDownload/`

Note: The CHECKOUT_BY_NAME service only marks the content item as locked. It does not perform a download.

Define the CLASSPATH Entry

To execute the GET_FILE service the SoapClientDownload application uses some content server classes. To access these classes you must include the *client.zip* file as a CLASSPATH environment variable. In most configurations the *client.zip* file is located in the <*install_dir*>/ weblayout/common/ directory.

- If a CLASSPATH environment variable is defined, append the complete path of the *client.zip* file to the existing entries.
- If a CLASSPATH environment variable is not defined, create a new system variable named CLASSPATH that defines the complete path to the *client.zip* file.

Define the Configuration File Properties

Define the correct configuration properties for your content server instance. Specify a User/Password with the appropriate permissions to execute the desired command. Many commands require administrative access:

```
Host=testserver
Port=80
CgiPath=/stellent/idcplg
User=sysadmin
Password=idc
```

Define the XML File Properties

Define the content server service to execute and the required (and optional) parameters as a SOAP packet in an XML-formatted document. This XML document is referenced from the command line when the command is executed:

```
<?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.oracle.com/IdcService/"
IdcService="GET_FILE">
<idc:document dID="10">
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Run the Command

Execute the service from the command line by changing to the *SoapClientDownload/* directory, referencing the Java program, and defining the command line parameters:

```
<install_dir>/custom/WsdlGenerator/samples/Java/SoapClientDownload>
java SoapClientDownload -c soap.cfg -x get_file.xml
```

Chapter

4

SOAP SERVICE CALLS

OVERVIEW

This chapter discusses executing various Content Server IdcCommand services using the SOAP interface. Content Server IdcCommand services can be executed using the SOAP interface. The user must have appropriate permissions to execute the commands. Some commands require administrative access, other commands may require only write permission.

You must install and enable the WSDL Generator component to call services. See WSDL Generator Component Installation (page 2-6) for details. See the *Services Reference Guide* for a list of available services and the required parameters.

This chapter contains these topics:

SOAP PACKET FORMAT

A SOAP request is an 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.

HTTP Headers

This entry is required in a SOAP request HTTP header:

```
Content-Type: text/xml; charset="utf-8"
```

This SOAPAction header is suggested, but not required:

```
SOAPAction: "http://www.oracle.com/IdcService"
```

Namespaces

Within the body of a SOAP message the SOAP message XML namespaces are used to qualify element and attribute names within the parts of the document. Element names can be global (referenced throughout the SOAP message) or local. The local element names are provided by namespaces and are used in the particular part of the message where they are located. Thus, SOAP messages use namespaces to qualify element names in the separate parts of a message. Application specific namespaces are used to qualify application specific element names. Namespaces also identify the envelope version and encoding style.

Content Server defines a namespace called “idc” that explains the schema and allowable tags for the SOAP content.

Nodes

A SOAP node is the entity that processes a SOAP message according to the rules for accessing the services provided by the underlying protocols through the SOAP bindings. Thus, message processing involves mapping to the underlying services. The SOAP specification defines a correlation between the parts of a SOAP message and the software handlers that will process each part of the message.

The following nodes may be required for a service request or may be returned in the response:

- Service Node (page 4-3)
- Document Node (page 4-3)
- User Node (page 4-4)
- Optionlist Node (page 4-4)
- Option Sub-Node (page 4-4)
- Resultset Sub-Node (page 4-5)
- Row Sub-Node (page 4-5)
- Field Sub-Node (page 4-5)

Note: On requests, content server services are lenient as to where data is specified. If you specify a data field in a field node and it is supposed to be a document attribute, or vice versa, the service still processes the data correctly. The response puts the data in the correct node.

Service Node

This is the main node of the IDC namespace.

- This node must exist to process a request.
- The required attribute *IdcService* defines the service you are requesting.
- It is not required that the sub-nodes of <service> carry the namespace in their tags. For example, <document> can be used rather than <idc:document>. However if you do define the namespace identifier in the child nodes, it must match the one specified in the service tag.

For example:

```
<idc:service xmlns:idc="http://www.oracle.com/IdcService/"
IdcService="PING_SERVER">
</idc:service>
```

Document Node

This node contains all content item information and is the parent node of all data nodes.

Attributes that are valid for your content items are defined by your particular content server instance. For example, *dID*, *dDocTitle*, and *dDocType* are common attributes.

- Custom content item information such as *xSpec* is valid if defined as metadata.

- All known document fields can be used as attributes.

In the following document node example, the CHECKOUT_BY_NAME service is used:

```
<idc:service xmlns:idc="http://www.oracle.com/IdcService/"
IdcService="CHECKOUT_BY_NAME">
<idc:document dDocName="soap_sample">
</idc:document>
</idc:service>
```

User Node

This is the node to contain all user information:

- Attributes that are valid for users are defined by your specific content server instance. For example, *dName*, *dFullName*, and *dEmail* are common attributes.
- Custom user information is valid if defined as metadata.
- All known user fields may be used as attributes.

For example:

```
<idc:service xmlns:idc="http://www.oracle.com/IdcService/"
IdcService="GET_USER_INFO">
<idc:user dUser="sysadmin">
</idc:user>
</idc:service>
```

Optionlist Node

This is the node to contain any option lists. The *name* attribute specifies the name of the option list. Each option sub-node contains a value in the optionlist node.

For example:

```
<idc:optionlist name="Users_UserLocaleList">
<idc:option>
English-US
</idc:option>
</idc:optionlist>
```

Option Sub-Node

This sub-node is specified within the <optionlist> node. The *option* attribute specifies the name of the option for the option list.

For example:

```
<idc:optionlist name="dDocType">
<idc:option>ADACCT</idc:option>
```

```
<idc:option>ADHR</idc:option>
<idc:option>ADSALES</idc:option>
</idc:optionlist>
```

Resultset Sub-Node

This sub-node can be specified within a <document> or <user> node.

- This sub-node contains result set information in a request or response.
- The *name* attribute specifies the name of the result set.

For example:

```
<idc:resultset name="REVISION_HISTORY">
<idc:row dFormat="text/plain" dInDate="4/12/02 1:27 PM" dOutDate=""
dStatus="RELEASED" dProcessingState="Y" dRevLabel="1" dID="6"
dDocName="stellent" dRevisionID="1">
</idc:row>
</idc:resultset>
```

Row Sub-Node

This sub-node is specified within a <resultset> sub-node.

- This sub-node can appear multiple times within <resultset> and specifies each row in the result set.
- Attributes that are valid are defined by your specific content server instance. These are the same fields that can appear as attributes in a document and/or user node.

For example:

```
<idc:resultset name="UserAttribInfo">
<idc:row dUserName="jsmith" AttributeInfo="role,contributor,15">
</idc:row>
</idc:resultset>
```

Field Sub-Node

This sub-node can be specified within a <document>, <user>, or <row> node. The *name* attribute specifies the name of the field.

Often represents data such as *refreshSubjects* or *dSubscriptionID.* For example:

```
:<idc:field name="dSubscriptionID">
stellent
</idc:field>
```

- May represent document or user metadata that is user configurable or custom metadata such as *xComments*.

- Used to pass search result values such as *QueryText* and *OriginalQueryText.* For example:

```
<idc:field name="QueryText">
dDocType <Substring> "ADSALES&"
</idc:field>
```

SPECIAL CHARACTERS

When passing characters such as '<' or '>' to Content Server, you must use the XML-encoding format:

Standard Format	XML-Encoding
<	<
>	>
"	"
&	&
\	'

Note: Some search result values such as *QueryText* and *OriginalQueryText* are URL-encoded in the response.

This example passes a string submitted for a Content Server content item query as both a standard formatted string and an XML-encoded format:

- Parameter with standard formatted string.

```
QueryText=dDocType <Substring> "ADSALES"
```

- Parameter with XML-encoded string.

```
<idc:field name="QueryText">
dDocType &lt;Substring&gt; &quot;ADSALES&quot;
</idc:field>
```

Chapter 5

USING ACTIVE SERVER PAGES

OVERVIEW

You can execute Content Server IdcCommand services from an Active Server Page by encapsulating a SOAP packet that defines the service to execute and the required parameters. You must have appropriate permissions to execute the commands. Some commands require administrative access, other commands may require only write permission.

This chapter discusses these topics:

Note: See Special Characters (page 4-6) for information on formatting XML-encoded strings.

SAMPLE SOAP REQUEST

This section provides an example Active Server Page that calls a service from the content server. A description of the service is provided including the required and optional parameters. This section also provides an XML-formatted version of the embedded SOAP request.

This example XML-formatted SOAP request uses the GET_SEARCH_RESULTS service to retrieve the search results for the passed query text.

Note: See Appendix A (*Sample Service Calls*) for additional information on service calls including required and optional parameters.

```
<?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.oracle.com/IdcService/"
IdcService="GET_SEARCH_RESULTS">
<idc:document>
<idc:field name="QueryText">
dDocType <Substring> "ADSALES"
</idc:field>
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SAMPLE ACTIVE SERVER PAGE

The embedded SOAP request forms the basis of the Active Server Page. This sample executes the GET_SEARCH_RESULTS.

```
<%
```

Note: See Appendix A (*Sample Service Calls*) for additional information on service calls and examples of SOAP response/request messages.

```
' Sample ASP page of sending a DOC_INFO Soap request.

Option Explicit

Response.Write("Search Results")

%>
<br><br>
<%

' Construct the Soap request.
Dim strSoapRequest, strQueryText

strQueryText = Request.Form("QueryText")
strQueryText = Server.HtmlEncode(strQueryText)

strSoapRequest = "<?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.oracle.com/IdcService/""
IdcService=""GET_SEARCH_RESULTS"">" _
& "<idc:document>" _
& "<idc:field name=""QueryText"">" & strQueryText & "</idc:field>" _
& "<idc:field name=""SortField"">" & Request.Form("SortField") & "</idc:field>" _
& "<idc:field name=""SortOrder"">" & Request.Form("SortOrder") & "</idc:field>" _
& "<idc:field name=""ResultCount"">" & Request.Form("ResultCount") &
"</idc:field>" _
& "<idc:field name=""Auth"">Internet</idc:field>" _
& "</idc:document>" _
& "</idc:service>" _
& "</SOAP-ENV:Body>" _
& "</SOAP-ENV:Envelope>"

' Send the Soap request.
Dim objXmlHttp
Set objXmlHttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
objXmlHttp.open "POST", "http://localhost/stellent/idcplg", False, "sysadmin",
"idc"
objXmlHttp.setRequestHeader "Content-Type", "text/xml; charset=utf-8"
objXmlHttp.send(strSoapRequest)

' Parse the Soap response.
Dim objXmlDoc
Set objXmlDoc = Server.CreateObject("Msxml2.DOMDocument")
objXmlDoc.async = False
objXmlDoc.Load objXmlHttp.responseXml

' Check for errors.
Dim strResponseError
strResponseError = objXmlDoc.parseError.reason
If strResponseError <> "" Then
Response.Write(objXmlHttp.ResponseText)
DisplayBackButton()
Response.End
End If

' Check for a fault string.
Dim objXmlFaultNode
Set objXmlFaultNode = objXmlDoc.documentElement.selectSingleNode("//SOAP-
ENV:Fault/faultstring")
If (Not (objXmlFaultNode Is Nothing)) Then
Response.Write(objXmlFaultNode.Text)
DisplayBackButton()
Response.End
End If

' Check the status code.
```

```
Dim objXmlStatusCodeNode, objXmlStatusMessageNode, strStatusCode, nStatusCode,
strStatusMessage
Set objXmlStatusCodeNode =
objXmlDoc.documentElement.selectSingleNode("//idc:field[@name='StatusCode']")
If (Not objXmlStatusCodeNode Is Nothing) Then
nStatusCode = CInt(objXmlStatusCodeNode.Text)
If (nStatusCode < 0) Then

Response.Write(objXmlDoc.documentElement.selectSingleNode("//idc:field[@name='Stat
usMessage']").Text)
DisplayBackButton()
Response.End
End If
End If

' Display search results
Dim strDocName, strDocTitle, strDocType, strInDate, strComments, nCurRow,
nTotalRows
Dim objXmlResultNodeList, objXmlCommentNode

Set objXmlResultNodeList =
objXmlDoc.documentElement.selectNodes("//idc:resultset[@name='SearchResults']/idc:
row")
nTotalRows = objXmlResultNodeList.Length

%>
<table>
<tr>
<td><b>Content ID</b></td>
<td> </td>
<td><b>Title</b></td>
<td> </td>
<td><b>Type</b></td>
<td> </td>
<td><b>Release Date</b></td>
<td> </td>
<td><b>Comments</b></td>
</tr>

<%
For nCurRow = 0 To (nTotalRows - 1)
strDocName = GetXmlNodeValue(objXmlResultNodeList.Item(nCurRow), "dDocName")
strDocTitle = GetXmlNodeValue(objXmlResultNodeList.Item(nCurRow), "dDocTitle")
strDocType = GetXmlNodeValue(objXmlResultNodeList.Item(nCurRow), "dDocType")
strInDate = GetXmlNodeValue(objXmlResultNodeList.Item(nCurRow), "dInDate")
strComments = GetXmlNodeValue(objXmlResultNodeList.Item(nCurRow), "xComments")

%>
```

```
<tr>
<td><%=strDocName%></td>
<td> </td>
<td><%=strDocTitle%></td>
<td> </td>
<td><%=strDocType%></td>
<td> </td>
<td><%=strInDate%></td>
<td> </td>
<td><%=strComments%></td>
</tr>
<%
Next
%>

</table>

<%

DisplayBackButton()
'----------------------------
Function GetXmlNodeValue(objXmlRowNode, strNodeName)
'----------------------------
Dim objXmlNode, objXmlNodeValue

Set objXmlNode = objXmlRowNode.selectSingleNode("@" & strNodeName)
If (objXmlNode Is Nothing) Then
Set objXmlNode = objXmlRowNode.selectSingleNode("idc:field[@name='" & strNodeName
& "']")
End If

If (Not (objXmlNode Is Nothing)) Then
GetXmlNodeValue = objXmlNode.Text
End If
'----------------------------
End Function
'----------------------------

'----------------------------
Sub DisplayBackButton()
'----------------------------
%>
<form method=POST action="request.asp">
<table>
<tr>
<td><input type=submit value="Back"></td>
</tr>
</table>
</form>
```

```
<%
'----------------------------
End Sub
'----------------------------
%>
```

Chapter

6

USING WSDL FILES

OVERVIEW

This section provides an overview of the Content Server WSDL files. In addition, information on generating WSDL files for interfacing with Content Server services is provided.

This section contains these topics:

UNDERSTANDING WSDL FILES

WSDL files provide the ability to pass data that can be understood by content server services. This enables access to the content and content management functions within Content Server. After the component is installed, the provided WSDL files are stored in the <*install_dir*>/weblayout/groups/secure/wsdl/custom/ directory.

These WSDL files are provided with the WSDL Generator component:

- CheckIn.wsdl
- DocInfo.wsdl
- GetFile.wsdl
- MetaData.wsdl

- PortalInfo.wsdl
- Search.wsdl
- Subscription.wsdl
- Workflow.wsdl

Additional WSDL files can be generated using the Soap Custom WSDL administrative pages. See Chapter 7 (*WSDL Administration Tutorial)* for additional information.

Note: IIS 6.0 does not serve files with an extension that is not registered in the MIME types list. Add the `.wsdl` extension with a MIME type of text/xml in order for it to be served by IIS 6.0.

WSDL File Structure

WSDL files are formally structured with elements that contain a description of the data to be passed to the web service. This enables both the sending application and the receiving application to interpret the data being exchanged.

WSDL elements contain a description of the operation to perform on the data and a binding to a protocol or transport. This permits the receiving application to both process the data and interpret how to respond or return data. Additional sub-elements may be contained within each WSDL element.

The WSDL file structure includes these major elements:

- Data Types—generally in the form of XML schema to be used in the messages.
- Message—the definition of the data in the form of a message either as a complete document or as arguments to be mapped to a method invocation.
- Port Type—a set of operations mapped to an address. This defines a collection of operations for a binding.
- Binding:—the actual protocol and data formats for the operations and messages defined for a particular port type.
- Service and Port—the service maps the binding to the port and the port is the combination of a binding and the network address for the communication exchange.

Note: The code fragments in this section are from the *DocInfo.wsdl* file provided with the WSDL Generator component. See Sample WSDL File (page 6-4) for a complete WSDL file.

Data Type

The Data Type <types> defines the complex types and associated elements. Web services supports both simple data types (such as string, integer, or boolean) and complex data types. A complex type is a structured XML document that contains a number of simple types or an array of sub-elements.

The following fragment for the ContentInfo set defines the Name, Title, Author, and Group elements and specifies that they are strings.

```
<s:complexType name="ContentInfo">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="dDocName" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="dDocTitle" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="dDocType" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="dDocAuthor" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="dSecurityGroup" type="s:string"/>
</s:sequence>
</s:complexType>
```

Message

The Message <message> defines the data as arguments to be mapped to a method invocation.

```
<message name="DocInfoByIDSoapIn">
<part name="parameters" element="s0:DocInfoByID" />
</message>
<message name="DocInfoByIDSoapOut">
<part name="parameters" element="s0:DocInfobyIDResponse" />
</message>
```

Port Type

The Port Type <portType> defines a collection of operations for a binding. The DocInfo.wsdl file provides the DocInfoSoap and the DocInfo operation name (method name) with input/output information for processing the message.

```
<portType name="DocInfoSoap">
<operation name="DocInfoByID">
<input message="s0:DocInfoByIDSoapIn" />
<output message="s0:DocInfoByIDSoapOut" />
</operation>
</portType>
```

Note: While a port type is a collection of operations (like classes in Java), WSDL is an independent data abstraction that provides more functionality than simply mapping to .NET, EJB, or CORBA objects.

Binding

The binding <binding> defines the actual protocol and data formats for the operations and messages for the particular port type.

```
<binding name="DocInfoSoap" type="s0:DocInfoSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
<operation name="DocInfoByID">
<soap:operation soapAction="http://wwww.oracle.com/Soap/DocInfo/"
style="document" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
</binding>
```

Service and Port

The service <service> maps the binding to the port. The port is the combination of a binding and the network address for the communication exchange. The port is used to expose a set of port types (operations) on the defined transport.

```
<service name="DocInfo">
<port name="DocInfoSoap" binding="s0:DocInfoSoap">
<soap:address location="HTTP://testserver/stellent/idcplg" />
</port>
</service>
```

Tech Tip: You can add `&IsSoap=1` to the URL of a Content Server browser window to view the underlying SOAP code for that page.

SAMPLE WSDL FILE

This sample code presents the complete *DocInfo.wsdl* file. This file along with the CheckIn.wsdl, GetFile.wsdl, and Search.wsdl files are found in the weblayout/groups/secure/wsdl/custom directory of the Content Server <*install_dir*>.

```
<?xml version='1.0' encoding='utf-8' ?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://wwww.oracle.com/DocInfo/"
targetNamespace="http://wwww.oracle.com/DocInfo/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
<s:schema elementFormDefault="qualified"
targetNamespace="http://www.oracle.com/DocInfo/">
<s:element name="DocInfoByID">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="dID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="extraProps" type="s0:IdcPropertyList"
/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="DocInfoByIDResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="DocInfoByIDResult"
type="s0:DocInfoByIDResult" />
</s:sequence>
</s:complexType>
</s:element>
<s:complexType name="DocInfoByIDResult">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="ContentInfo"
type="s0:ContentInfo" />
<s:element minOccurs="0" maxOccurs="unbounded" name="Revisions"
type="s0:Revisions" />
<s:element minOccurs="0" maxOccurs="unbounded" name="WorkflowInfo"
type="s0:WorkflowInfo" />
<s:element minOccurs="0" maxOccurs="1" name="StatusInfo" type="s0:StatusInfo" />
</s:sequence>
</s:complexType>
<s:element name="DocInfoByName">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="dDocName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="extraProps" type="s0:IdcPropertyList"
/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="DocInfoByNameResponse">
<s:complexType>
```

```
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="DocInfoByNameResult"
type="s0:DocInfoByNameResult" />
</s:sequence>
</s:complexType>
</s:element>
<s:complexType name="DocInfoByNameResult">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="ContentInfo"
type="s0:ContentInfo" />
<s:element minOccurs="0" maxOccurs="unbounded" name="Revisions"
type="s0:Revisions" />
<s:element minOccurs="0" maxOccurs="unbounded" name="WorkflowInfo"
type="s0:WorkflowInfo" />
<s:element minOccurs="0" maxOccurs="1" name="StatusInfo" type="s0:StatusInfo" />
</s:sequence>
</s:complexType>
<s:complexType name="ContentInfo">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="dDocName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dDocTitle" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dDocType" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dDocAuthor" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dSecurityGroup" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dDocAccount" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="dRevClassID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="dRevisionID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="dRevLabel" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dIsCheckedOut" type="s:boolean" />
<s:element minOccurs="0" maxOccurs="1" name="dCheckoutUser" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dCreateDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dInDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dOutDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dStatus" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dReleaseState" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dFlag1" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWebExtension" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dProcessingState" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dMessage" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dReleaseDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dRendition1" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dRendition2" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dIndexerState" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dPublishType" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dPublishState" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dDocID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="dIsPrimary" type="s:boolean" />
<s:element minOccurs="0" maxOccurs="1" name="dIsWebFormat" type="s:boolean" />
```

```
<s:element minOccurs="0" maxOccurs="1" name="dLocation" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dOriginalName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dFormat" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dExtension" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dFileSize" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="CustomDocMetaData"
type="s0:IdcPropertyList" />
</s:sequence>
</s:complexType>
<s:complexType name="Revisions">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="dFormat" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dInDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dOutDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dStatus" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dProcessingState" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dRevLabel" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="dDocName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dRevisionID" type="s:int" />
</s:sequence>
</s:complexType>
<s:complexType name="WorkflowInfo">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="dWfID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="dDocName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWfDocState" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWfComputed" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWfCurrentStepID" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="dWfDirectory" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dClbraName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWfName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWfDescription" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dCompletionDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dSecurityGroup" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWfStatus" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dWfType" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dProjectID" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="dIsCollaboration" type="s:boolean" />
</s:sequence>
</s:complexType>
<s:complexType name="StatusInfo">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="statusCode" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="statusMessage" type="s:string" />
</s:sequence>
</s:complexType>
<s:complexType name="IdcPropertyList">
<s:sequence>
```

```
<s:element minOccurs="0" maxOccurs="unbounded" name="property"
type="s0:IdcProperty" />
</s:sequence>
</s:complexType>
<s:complexType name="IdcProperty">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="name" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="value" type="s:string" />
</s:sequence>
</s:complexType>
</s:schema>
</types>
<message name="DocInfoByIDSoapIn">
<part name="parameters" element="s0:DocInfoByID" />
</message>
<message name="DocInfoByIDSoapOut">
<part name="parameters" element="s0:DocInfoByIDResponse" />
</message>
<message name="DocInfoByNameSoapIn">
<part name="parameters" element="s0:DocInfoByName" />
</message>
<message name="DocInfoByNameSoapOut">
<part name="parameters" element="s0:DocInfoByNameResponse" />
</message>
<portType name="DocInfoSoap">
<operation name="DocInfoByID">
<input message="s0:DocInfoByIDSoapIn" />
<output message="s0:DocInfoByIDSoapOut" />
</operation>
<operation name="DocInfoByName">
<input message="s0:DocInfoByNameSoapIn" />
<output message="s0:DocInfoByNameSoapOut" />
</operation>
</portType>
<binding name="DocInfoSoap" type="s0:DocInfoSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<operation name="DocInfoByID">
<soap:operation soapAction="http://www.oracle.com/DocInfo/" style="document" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="DocInfoByName">
<soap:operation soapAction="http://www.oracle.com/DocInfo/" style="document" />
<input>
<soap:body use="literal" />
```

```
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
</binding>
<service name="DocInfo">
<port name="DocInfoSoap" binding="s0:DocInfoSoap">
<soap:address location="http://snelsonpc/idcplg75/idc_cgi_isapi.dll" />
</port>
</service>
</definitions>
```

GENERATING WSDL FILES

Note: The WSDL Generator component must be installed to generate WSDL files. See WSDL Generator Component Installation (page 2-6) for additional information.

When the Content Server is restarted after installing and enabling the WSDL Generator component, several folders and related HDA files are generated that expose several Services as web services. Two directories are created in the *<install_dir>*/data/soap/ directory. The /generic directory contains a generic.hda file and the /custom directory contains a wsdl_custom.hda file. Administrators can customize or add WSDL files using the *Soap Wsdl* administration pages. These pages are accessed by clicking the **Soap WSDL** link from the Administration section of the Admin Applet page.

Note: See Chapter 7 (*WSDL Administration Tutorial)* for step-by-step instructions on creating and editing a custom WSDL using the *Soap Custom Wsdl* administration pages.

GENERATING PROXY CLASS FROM WSDL FILES

Note: In addition to the WSDL files provided with the WSDL Generator component, you can generate additional WSDL files for any Content Server service. See Generating WSDL Files (page 6-9) for additional information.

Using the WSDL files, developers may choose to create proxy classes to plug into a development tool. A number of software products and tool kits are available for converting WSDL files to programming class files in languages such as Java, Visual Basic, and C#. For example, Apache AXIS provides a SOAP to Java toolkit, and

Microsoft .NET Development Environment provides functionality to convert WSDL files to C#.

If using Microsoft .NET, you can use the utilitywsdl.exe to generate the proxy classes:

```
wsdl /l:CS DocInfo.wsdl
```

This utility generates the file DocInfoService.cs (C# class) which contains the class DocInfoService and the function DocInfo with the parameters specified. The return value is the DocInfoSet class, which is all the response parameters specified, along with ErrorCode and ErrorMessage values. If the ErrorCode is less than zero, an error has occurred in the service call, and you can see the specifics of it in the value ErrorMessage.

Chapter

7

WSDL ADMINISTRATION TUTORIAL

OVERVIEW

The Soap Custom Wsdl administration pages provide an administrator with the ability to edit and customize WSDL files. This chapter provides an administrative tutorial that gives step-by-step instructions on creating and editing a custom WSDL.

The WSDL Generator component must be installed to generate WSDL files. See WSDL Generator Component Installation (page 2-6) for details. In addition to the WSDL files provided with the WSDL Generator component, you can generate additional WSDL files for any Content Server service. See Generating WSDL Files (page 6-9) for additional information.

See the *Services Reference Guide* for a list of available services and the required parameters.

TUTORIAL

This section provides step-by-step instructions on creating and editing a custom WSDL using the Soap Custom WSDL administration pages.

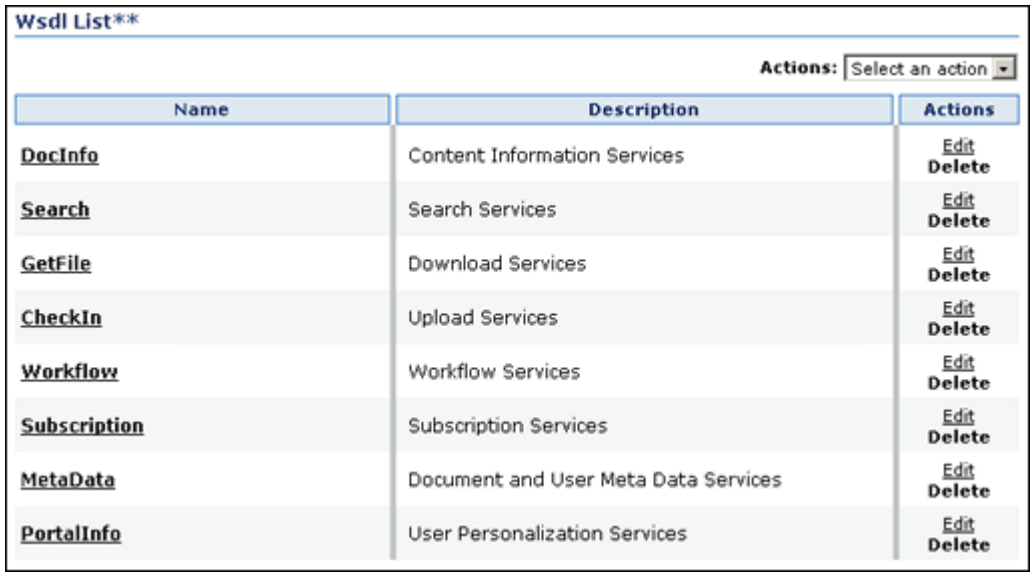
1. Access the Admin Applets page.

Figure 7-5 Admin Applets

2. Click the **Soap Wsdls** link.

   The Wsdl List page is displayed.

Figure 7-6 The Wsdl List page

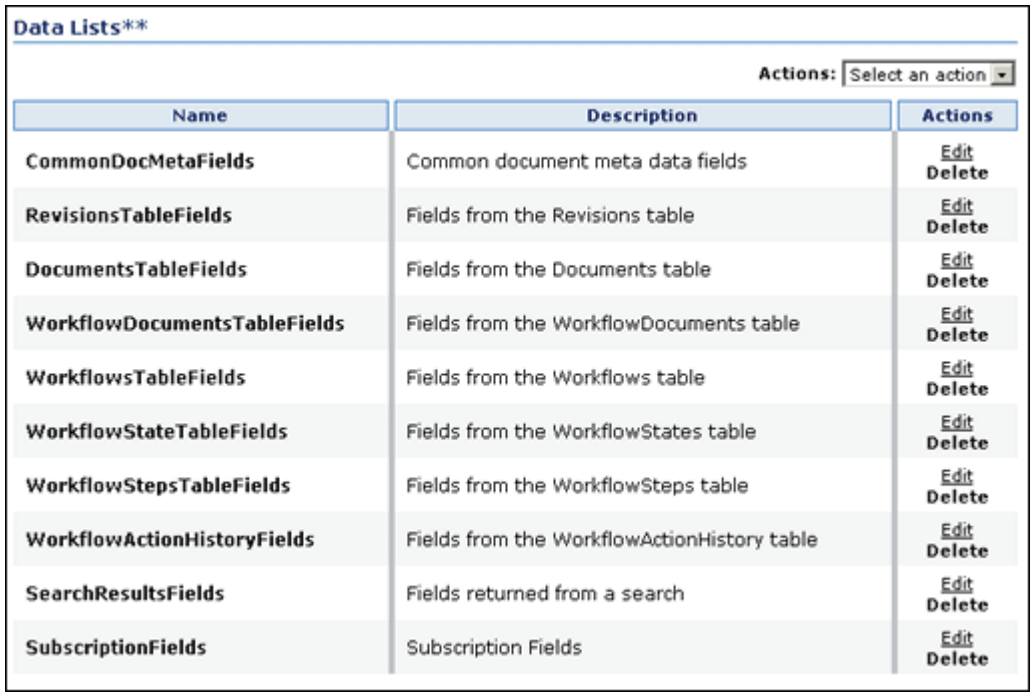


3. Click **Data Lists** from the drop-down Action menu.

   The Data Lists page is displayed.

   Data Lists are global lists of data that can be used with complex types, service parameters, or other DataLists. When a DataList is specified as a parameter or a sub-type of a complex type, all the sub-types of the DataList will appear as data types. DataLists are defined once but can be referenced multiple times with different WSDLs and services. All the DataLists have a prefix of "d:" in the data type list.

Figure 7-7 DataList page



Note: System-specific WSDLs cannot be deleted. You can, however, edit the WSDL and enable or disable the complex type elements for that WSDL.

4. Select **Add Data List** from the drop-down Actions menu.

   The Add Data List page is displayed.

   Enter the following information:

   Name: **UserMetaFields**

   Description: **User Metadata Fields**

5. Click **Add**.

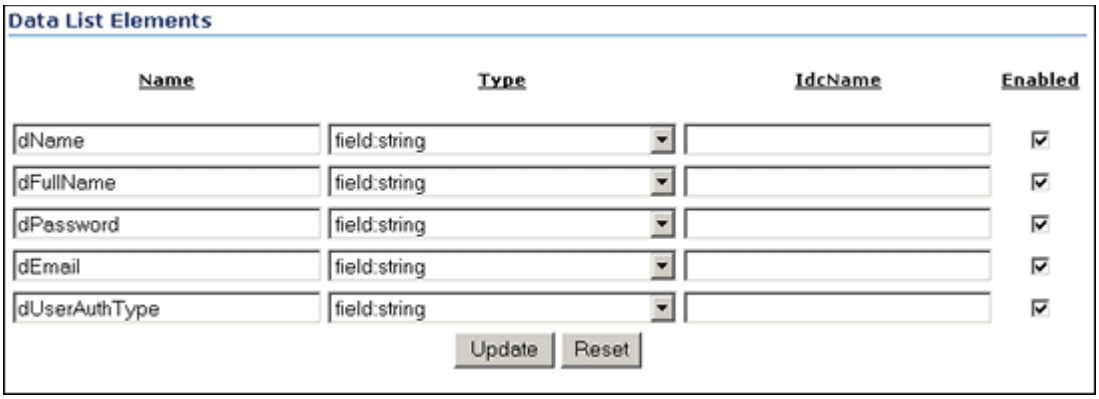
   The Data List Information / Data List Elements page is displayed.

6. Enter the following Data Elements information, selecting the Type from the pulldown menu:

Name	Type	Idc Name
dName	field:string	
dFullName	field:string	

Name	Type	Idc Name
dPassword	field:string	
dEmail	field:string	
dUserAuthType	field:string	

Figure 7-8 DataList Elements



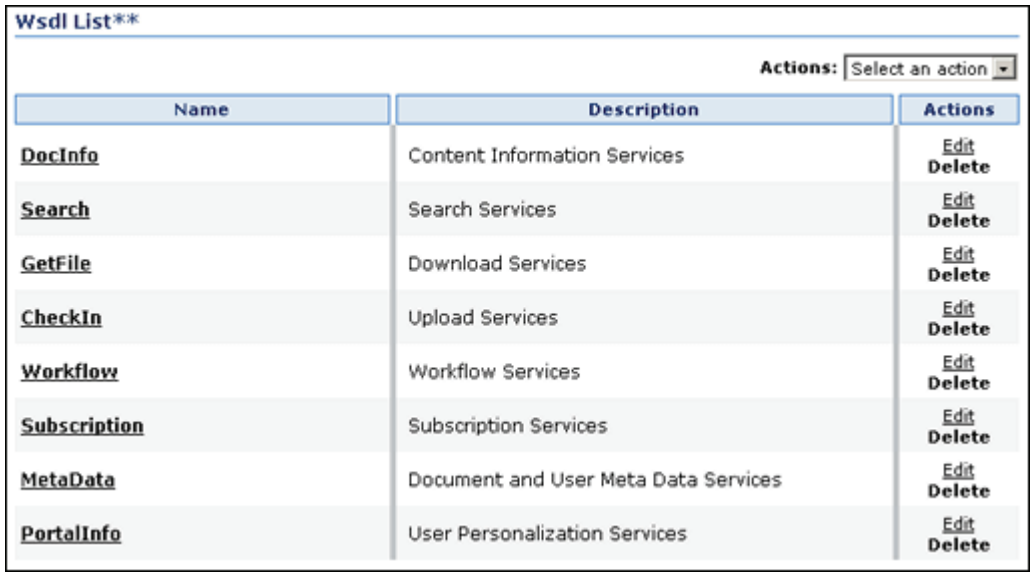
7. Click **Update**.

   You are returned to the updated Data Lists page. Note that UserMetaFields now appears at the bottom of the list.

8. Select **Wsdls List** from the drop-down Actions menu.

   The Wsdls List page is displayed:

Figure 7-9 Wsdls List page re-displayed



9. Select **Add Wsdl** from the drop-down Actions menu.

   The Add Wsdl page is displayed.

   Enter the following information:

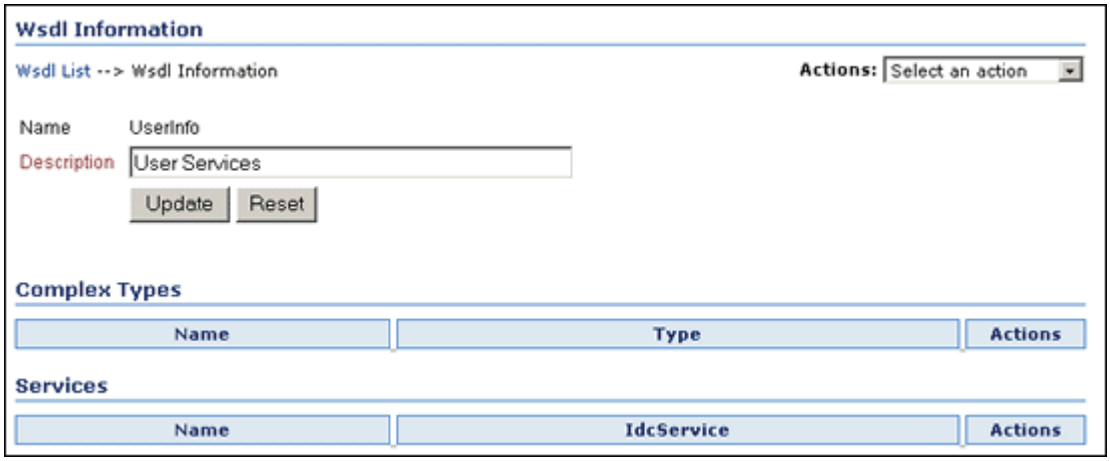
   Name: **UserInfo**

   Description: **User Services**

10. Click **Add**.

    The Wsdl Information page is displayed.

Figure 7-10 Wsdl Information page



11. Select **Add Complex Type** from the drop-down Actions menu.

    The Add Complex Type page is displayed.

Note: Complex types contain other data types as sub-types. Once these are created, any service in the WSDL can use these complex types as parameters.

Enter the following Complex Type information:

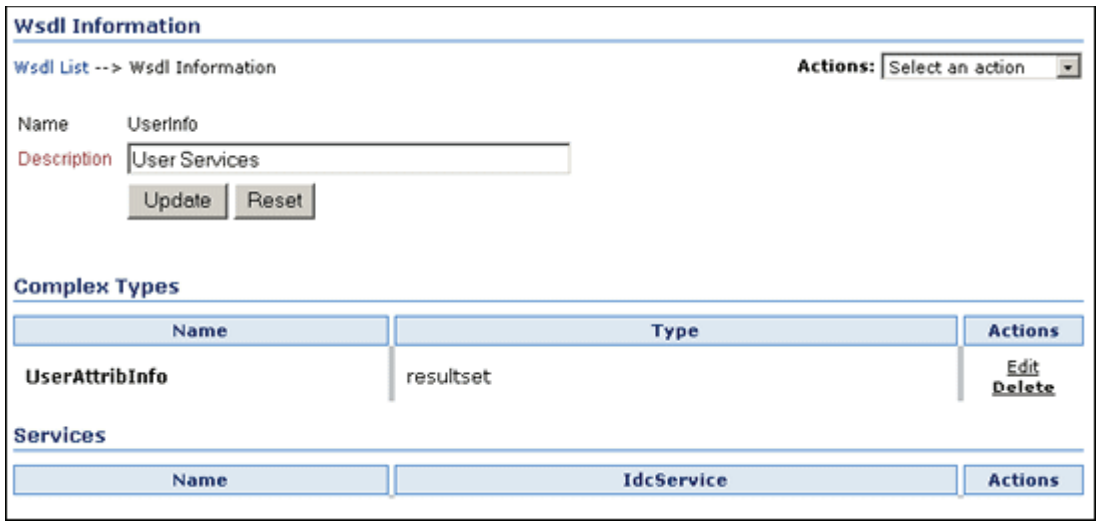
Name: **UserAttribInfo**

Type: select **resultset** from the pulldown menu

12. Click **Add**.

    The Wsdl Information page is displayed.

Figure 7-11 Wsdl Information page re-displayed



13. Select **Edit** from the UserAttribInfo line.

    The Complex Type Information/Complex Type Elements page is displayed.

14. Enter the following Complex Type Elements, selecting the Type from the pulldown menu:

Name	Type	Idc Name
dUserName	field:string	
AttributeInfo	field:string	

15. Click **Update** in the Complex Type Elements section.

You are returned to the updated Wsdl Information page. Note that User AttribInfo now appears as a complex type.

16. Select **Add Service** from the drop-down Actions menu.

    The Add Service page is displayed.

    Enter the following information:

    Name: **AddUser**

    IdcService: **ADD_USER**

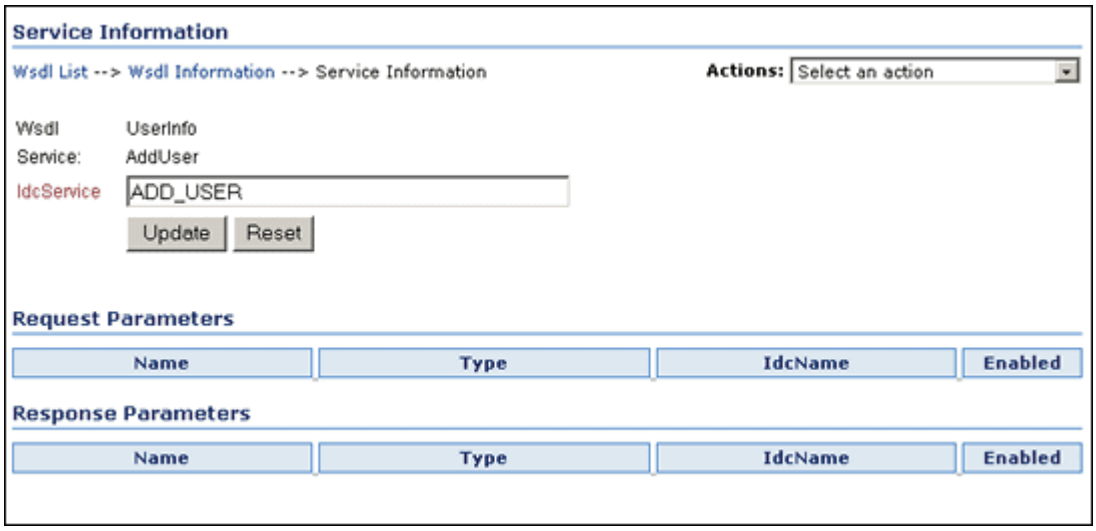
17. Click **Add**.

    The Wsdl Information page is displayed.

18. Select **Edit** from the AddUser Service line.

    The Service Information page is displayed.

Figure 7-12 Service Information page



Note: When you create a WSDL, you create services that correspond to the IdcServices of the content server. You also specify the request and response parameters you want the service to pass and receive from the Web Service call.

19. Select **Update Request Parameters** from the drop-down Actions menu.

    The Request Parameters page is displayed.

    Enter the following information, selecting the Type from the pulldown menu:

Name	Type	Idc Name
DataList	d:UserMetaFields	
CustomUserData	propertylist:CustomUserMeta	

20. Click **Update**.

    You are returned to the updated Service Information page. Note that DataList and CustomUserData now appears in the Request Parameters section.

21. Click **Update**.

    You are returned to the updated Wsdl Information page, showing the service that you just added.

22. Click **Update** again.

    You are returned to the updated Wsdls List page. UserInfo appears at the bottom of the list.

23. Select **Generate Wsdls** from the drop-down Actions menu.

    A confirmation message displays after the Wsdls are generated successfully.

24. Click **Back**.

    You are returned to the Wsdl List page.

25. Click the **UserInfo** link in the Name column.

    The source code for the generated Wsdl file is displayed (a portion is shown below).

Figure 7-13 Source code, Wsdl file

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://www.stellent.com/UserInfo/"
    targetNamespace="http://www.stellent.com/UserInfo/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://www.stellent.com/UserInfo/">
      - <s:element name="AddUser">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="dName"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="dFullName"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="dPassword"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="dEmail"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="dUserAuthType"
                type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="CustomUserData"
                type="s0:IdcPropertyList" />
              <s:element minOccurs="0" maxOccurs="1" name="extraProps"
                type="s0:IdcPropertyList" />
            </s:sequence>
          </s:complexType>
        </s:element>
```

26. Click the browser **Back** button.

    You are returned to the Soap Custom Wsdl page.

Tech Tip: You can “right-click” the **View** link and save the Wsdl file to your desktop (for use with .NET, etc.). However, be sure to save the file with a .wsdl file extension rather than the default .XML file extension.

Appendix

A

SAMPLE SERVICE CALLS

INTRODUCTION

This appendix provides sample Service Calls with SOAP Response/Request (page A-1) and presents information on executing content server services in a SOAP request. See the *Services Reference Guide* for a list of available services and the required parameters.

SERVICE CALLS WITH SOAP RESPONSE/REQUEST

These IdcCommand services are used as SOAP request examples:

IdcCommand	Description
PING_SERVER (see page A-2)	This service evaluates whether a connection to the server exists.
ADD_USER (see page A-4)	This service adds a new user to the system.
EDIT_USER (see page A-8)	This service edits an existing user.
GET_USER_INFO (see page A-11)	This service retrieves the user list.
DELETE_USER (see page A-14)	This service deletes an existing user.
CHECKIN_UNIVERSAL (see page A-15)	This service performs a content server controlled check in.

IdcCommand	Description
CHECKOUT_BY_NAME (see page A-21)	This service marks the latest revision of the specified content item as locked.
UNDO_CHECKOUT_BY_NAME (see page A-23)	This service reverses a content item checkout using the Content ID.
DOC_INFO (see page A-26)	This service retrieves content item revision information.
GET_FILE (see page A-28)	This service retrieves a copy of a content item without performing a check out.
GET_SEARCH_RESULTS (see page A-32)	This service retrieves the search results for the passed query text.
GET_TABLE (see page A-36)	This service exports the specified table in the content server database.
GET_CRITERIA_WORKFLOWS_FOR_GROUP (see page A-38)	This service returns criteria workflow information.

Ping the Server

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

- This service returns status information on the content server.
- 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.

Required Parameters

These parameters must be specified:

Parameter	Description
IdcService	Must be set to PING_SERVER.

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.oracle.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.oracle.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>
```

Add a New User

The ADD_USER service adds a new user to the system.

- Given a user name, the service determines if the user is in the system. If the user does not exist, the service will add the user.
- The most likely error is when the user name is not unique. If this service is unable to execute, an error message is displayed to the user.

Required Parameters

These parameters must be specified:

Parameter	Description
dName	The unique name.
dUserAuthType	The user authorization type. This value must be set to either LOCAL or GLOBAL.
IdcService	Must be set to ADD_USER.

Optional Parameters

These optional parameters may be specified:

Parameter	Description
dEmail	The email address for the user.
dFullName	The full name of the user.
dPassword	The password for the user.

Optional Attribute Information

This optional data defines the user's attribute information, the roles the user belongs to, and the accounts the user has access to. Attribute information consists of a list of three comma-separated strings. The first string indicates the type of attribute, the second the name of the attribute, and the third is the access number.

Important: The user attribute information is not pre-defined. The user by default will belong to no roles or accounts, and will become a guest in the system.

Attribute Information	Description
Access Number	The access number determines the level of access or privileges assigned to the user
Attribute Name	The attribute name is the name of the *role* or *account* to be assigned. For example, *admin*, *contributor*, or *editor* may be assigned.
Attribute Type	The attribute types consists of *role* or *account.*

Access Number

These access numbers can be assigned to the user.

Access Level Flags	Description
1	Read only.
3	Read and write.
7	Read, write, and delete.
15	Administrative privileges.

Attribute Name

A user can belong to multiple roles and accounts, there may be multiple role and account information strings separated by commas in the attribute information column.

- If the user is to have the admin role, define the user attribute information as follows:

```
<idc:resultset name="UserAttribInfo">
<idc:row dUserName="jsmith" AttributeInfo="role,contributor,15">
```

- If the user is to belong to both the contributor and editor roles and has read privilege on the account books, define the user attribute information as:

```
<idc:resultset name="UserAttribInfo">
<idc:row dUserName="jsmith"
AttributeInfo="role,contributor,15,role,editor,15,account,books,1">
```

Attribute Type

When defining a role, the first string specifies that this is a role attribute, the second string is the name of the role, and the third is the default entry of 15.

When defining an account, the first string specifies that this is an account attribute, the second string is the name of the account, and the third is the access level.

- For an attribute role, the information is in the form:

```
role,contributor,15
```

- For an attribute account where the access level determines the users rights to the named account, the information is in the form:

```
account,books,1
```

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.oracle.com/IdcService/" IdcService="ADD_USER">
<idc:user dName="Jennifer" dFullName="Jennifer Anton" dPassword="password"
dEmail="email@email.com" dUserAuthType="local">
<idc:resultset name="UserAttribInfo">
<idc:row dUserName="Jennifer" AttributeInfo="role,contributor,3">
</idc:row>
</idc:resultset>
</idc:user>
</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.oracle.com/IdcService/" IdcService="ADD_USER">
<idc:document>
<idc:field name="refreshMonikers">
```

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

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

</idc:field>
<idc:field name="isAdd">
1
</idc:field>
<idc:field name="copyAll">
1
</idc:field>
<idc:field name="alwaysSave">
1
</idc:field>
<idc:field name="dAttributeName">
contributor
</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="doAdminFields">
1
</idc:field>
<idc:field name="dAttributePrivilege">
3
</idc:field>
<idc:field name="dAttributeType">
role
</idc:field>
<idc:field name="changedMonikers">

</idc:field>
<idc:field name="changedSubjects">
userlist,1018884022874
</idc:field>
</idc:document>
<idc:user dUserAuthType="local" dEmail="email@email.com" dFullName="Jennifer
Anton" dUser="sysadmin" dPassword="password" dName="Jennifer">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Edit Existing User

The EDIT_USER service edits an existing user.

- Given a user name and user authorization type, the service determines if the user is in the system. If the user does not exist, the service fails. Otherwise the user information is updated and replaced.
- The most likely error is the user does not have the security level to perform this action. If this service is unable to execute, an error message is displayed to the user.

Note: The user attribute information replaces the current attributes. It does not add to the list. Consequently, if the user attribute information is not defined, the user will become a guest in the system.

Required Parameters

These parameters must be specified:

Parameter	Description
dName	The unique name.
dUserAuthType	The user authorization type. This value must be set to either LOCAL or GLOBAL.
IdcService	Must be set to EDIT_USER.

Optional Parameters

These optional parameters may be specified:

Parameter	Description
dEmail	The email address of the user.
dFullName	The full name of the user.
dPassword	The password for the user.
dUserLocale	The locale designation such as English-US, English-UK, Deutsch, Français, Español.

Parameter	Description
dUserType	The defined user type.

Optional Attribute Information

A result set containing the user's attribute information and referencing the roles the user belongs to and the accounts the user has access to. Attribute information consists of a list of three comma-separated strings. The first string indicates the type of attribute, the second the name of the attribute, and the third is the access number.

Important: The user attribute information is not pre-defined. The user by default will belong to no roles or accounts, and will become a guest in the system

Attribute Information	Description
Access Number	The access number determines the level of access or privileges assigned to the user
Attribute Name	The attribute name is the name of the *role* or *account* to be assigned. For example, *admin*, *contributor*, or *editor* may be assigned.
Attribute Type	The attribute types consist of *role* or *account*.

Access Number

These access numbers can be assigned to the user.

Access Level Flags	Description
1	Read only.
3	Read and write.
7	Read, write, and delete.
15	Administrative privileges.

A user can belong to multiple roles and accounts, there may be multiple role and account information strings separated by commas in the attribute information column.

- If the user is to have the admin role, define the user attribute information as follows:

```
<idc:resultset name="UserAttribInfo">
<idc:row dUserName="jsmith" AttributeInfo="role,contribut
or,15">
```

- If the user is to belong to both the contributor and editor roles and has read privilege on the account books, define the user attribute information as:

```
<idc:resultset name="UserAttribInfo">
<idc:row dUserName="jsmith"
AttributeInfo="role,contributor,15,role,editor,15,account,books,1">
```

Attribute Type

When defining a role, the first string specifies that this is a role attribute, the second string is the name of the role, and the third is the default entry of 15.

When defining an account, the first string specifies that this is an account attribute, the second string is the name of the account, and the third is the access level.

- For an attribute role, the information is in the form:

```
role,contributor,15
```

- For an attribute account where the access level determines the users rights to the named account, the information is in the form:

```
account,books,1
```

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.oracle.com/IdcService/" IdcService="EDIT_USER">
<idc:user dName="Jennifer" dFullName="Jennifer Anton" dPassword="password"
dEmail="jennifer@email.com" dUserAuthType="local">
<idc:resultset name="UserAttribInfo">
<idc:row dUserName="Jennifer" AttributeInfo="role,guest,1">
</idc:row>
</idc:resultset>
</idc:user>
</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.oracle.com/IdcService/" IdcService="EDIT_USER">
<idc:document>
<idc:field name="refreshMonikers">

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

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

</idc:field>
<idc:field name="alwaysSave">
1
</idc:field>
<idc:field name="dAttributeName">
guest
</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="doAdminFields">
1
</idc:field>
<idc:field name="dAttributePrivilege">
1
</idc:field>
<idc:field name="dAttributeType">
role
</idc:field>
<idc:field name="changedMonikers">

</idc:field>
<idc:field name="changedSubjects">
userlist,1018884022877
</idc:field>
</idc:document>
<idc:user dUserAuthType="local" dEmail="jennifer@email.com" dFullName="Jennifer
Anton" dUser="sysadmin" dPassword="password" dName="Jennifer">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Get User Information

The GET_USER_INFO service retrieves the user list.

- Given a defined user, the service retrieves the user list.
- If this service is unable to execute, this message is displayed to the user: *Unable to retrieve user list.*

Required Parameters

These parameters must be specified:

Parameter	Description
dUser	The defined user.
IdcService	Must be set to GET_USER_INFO.

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.oracle.com/IdcService/"
IdcService="GET_USER_INFO">
<idc:user dUser="sysadmin">
</idc:user>
</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.oracle.com/IdcService/"
IdcService="GET_USER_INFO">
<idc:document>
<idc:field name="changedSubjects">

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

</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="changedMonikers">
```

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

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

</idc:field>
<idc:optionlist name="Users_UserLocaleList">
<idc:option>
English-US
</idc:option>
</idc:optionlist>
</idc:document>
<idc:user dUser="sysadmin" dName="sysadmin">
<idc:resultset name="UserMetaDefinition">
<idc:row umdName="dFullName" umdType="BigText" umdCaption="apTitleFullName"
umdIsOptionList="0" umdOptionListType="0" umdOptionListKey="" umdIsAdminEdit="0"
umdOverrideBitFlag="1">
</idc:row>
<idc:row umdName="dEmail" umdType="BigText" umdCaption="apTitleEmailAddress"
umdIsOptionList="0" umdOptionListType="" umdOptionListKey="" umdIsAdminEdit="0"
umdOverrideBitFlag="2">
</idc:row>
<idc:row umdName="dUserType" umdType="Text" umdCaption="apTitleUserType"
umdIsOptionList="1" umdOptionListType="combo"
umdOptionListKey="Users_UserTypeList" umdIsAdminEdit="0" umdOverrideBitFlag="4">
</idc:row>
<idc:row umdName="dUserLocale" umdType="Text" umdCaption="apTitleUserLocale"
umdIsOptionList="1" umdOptionListType="choice,locale"
umdOptionListKey="Users_UserLocaleList" umdIsAdminEdit="0" umdOverrideBitFlag="8">
</idc:row>
</idc:resultset>
<idc:resultset name="USER_INFO">
<idc:row dName="sysadmin" dFullName="System Administrator" dEmail=""
dPasswordEncoding="" dPassword="-----" dUserType="" dUserAuthType="LOCAL"
dUserOrgPath="" dUserSourceOrgPath="" dUserSourceFlags="0" dUserArriveDate=""
dUserChangeDate="" dUserLocale="" dUserTimeZone="">
</idc:row>
</idc:resultset>
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Delete User

The DELETE_USER service deletes an existing user.

- Given a user name, the service deletes the user from the system.
- The most likely error is when the user has been assigned to an alias. If this service is unable to execute, an error message is returned.

Required Parameters

These parameters must be specified:

Parameter	Description
dName	The unique name.
IdcService	Must be set to DELETE_USER.

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.oracle.com/IdcService/"
IdcService="DELETE_USER">
<idc:user dName="Jennifer" >
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
<idc:service xmlns:idc="http://www.oracle.com/IdcService/"
IdcService="DELETE_USER">
<idc:document>
<idc:field name="changedSubjects">
userlist,1018884022876
</idc:field>
<idc:field name="refreshSubjects">

</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
```

```
<idc:field name="changedMonikers">

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

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

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

Check In Content Item

The CHECKIN_UNIVERSAL service performs a content server controlled check-in.

- This service determines if the content item is new or already exists in the system by querying the database using the content ID (dDocName) as the key.
- If the content item exists in the system, the publish state (dPublishState) must be empty.
- If a revision label (dRevLabel) is specified, this service will check if the content revision exists in the system; an exception is thrown if the revision exists.
- This service will dispatch this request to:
  - CHECKIN_NEW SUB - If the content item does not exist in the server.
  - CHECKIN_SEL_SUB - If the content item exists on the system and no valid revision was specified and the content item is checked out.
  - WORKFLOW_CHECKIN_SUB - If the content item exists and is part of a workflow.
- The most likely errors are mismatched parameters or when the content item was not successfully checked in. If this service is unable to execute, this message is displayed to the user: *Content item "{dDocName}" was not successfully checked in.*

The `CHECKIN_UNIVERSAL` service is a content server controlled check in. The check in will fall into either a new, selected, or workflow check in process and follow the same

logic as a check in through the browser or Repository Manager application. If the content item to be checked in already exists in the system, the content item must be checked out for the check in to succeed.

These are essentially the same sub-services used during a content server controlled check in. However, these sub-services are not called during a BatchLoad or Archive import. This service will check security to determine if the user has sufficient privilege to perform a check in on the content item and if the content item (if it exists) has been checked out. Also, it will determine if the content item matches a workflow criteria or belongs to an active basic workflow.

If the content item is not found the content item is checked in using the `CHECKIN_NEW_SUB` sub-service. This sub-service validates the check in data and determines if this content item belongs to a criteria workflow. If the content item already exists in the system and the content item does not belong to a workflow, the `CHECKIN_SEL_SUB` is used. Otherwise the content item exists and belongs to a workflow and the `WORKFLOW_CHECKIN_SUB` is used.

Note: All paths use the forward slash ("/") as the file separator. This is because the backslash ("\") is an escape character. For example, `primaryFile=d:/temp/myfile.txt` should point to the primary file to check in.

Required Parameters

These parameters must be specified:

Parameter	Description
dDocAuthor	The content item author (contributor).
dDocName	The content item identifier (Content ID). ❖ This field is optional if the system has been configured with `IsAutoNumber` set to TRUE. In this scenario, if the `dDocName` is not specified, the check in will always be new, and the system will generate a new name for the content item. ❖ Otherwise, if `dDocName` is specified, the service will use this key to do a look up to determine what type of check in to perform.
dDocTitle	The content item title.

Parameter	Description
dDocType	The content item type.
dSecurityGroup	The security group such as PUBLIC or SECURE.
IdcService	Must be set to CHECKIN_UNIVERSAL.
primaryFile	The absolute path to the location of the file as seen from the server. Use the forward slash as the file separator. The primary file will not be uploaded as the native file, but instead will be converted. A primary file must be specified unless checking in metadata only. If an alternate file is specified with the primary file, the content refinery will convert the alternate file. Otherwise, the primary file will be converted. ❖ If a primary file is not specified, a metafile can be used in its place. Only one metafile can exist though for each content item (i.e. a primary AND alternate meta file cannot co-exist). ❖ If both a primary and alternate file is specified, their extensions must be different.

Important: Custom metadata fields that are defined must also be specified.

Additional Parameters

This parameter may be required:

Parameter	Description
dDocAccount	The security account for the content item. If you have accounts enabled, you must pass this parameter.

Optional Parameters

These optional parameters may be specified:

Parameter	Description
alternateFile	The alternate file for conversion. ❖ Only one metafile can exist though for each content item (a primary AND alternate meta file cannot co-exist.) ❖ If an alternate file is specified with the primary file, the content refinery will convert the alternate file. Otherwise, the primary file will be converted.
dCreateDate	The date the content item was created. By default, this is the current date.
dInDate	The content release date. The date the content item is to be released to the web. By default, this is the current date. If the content release date (dInDate) is not specified, the creation date (dCreateDate) is used. This value is auto generated if it is not supplied.
dOutDate	The content expiration date. By default, this is blank and does not specify an expiration date. If the content expiration date (dOutDate) is not entered, the value remains empty. This is a valid state.
dRevLabel	The revision label for the content item. If set, the label will be used to locate the specified revision.
isFinished	Set to TRUE (1) if this is a workflow check-in and you have finished editing it. See WORKFLOW_CHECKIN for additional information.

Note: Do not confuse the Content ID (dDocName) with the internal content item revision identifier (dID). The *dID* is a generated reference to a specific rendition of a content item

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.oracle.com/IdcService/"
IdcService="CHECKIN_UNIVERSAL">
<idc:document dDocName="SoapUpload2" dDocAuthor="sysadmin" dDocTitle="Soap Upload
2 Document" dDocType="ADACCT" dSecurityGroup="Public" dDocAccount="">
<idc:file name="primaryFile"
href="C:/stellent/custom/Soap/JavaSamples/SoapClientUpload/soaptest.doc">
</idc:file>
</idc:document>
</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.oracle.com/IdcService/"
IdcService="CHECKIN_UNIVERSAL">
<idc:document dDocAuthor="sysadmin" dDocName="SoapUpload2" dExtension="doc"
dDocAccount="" dIsPrimary="1" dRevisionID="1" dPublishType="" dInDate="4/22/02
1:31PM" dReleaseState="N" dRevClassID="12" dCreateDate="4/22/02 1:31 PM"
dIsWebFormat="0" dPublishState="" dLocation="" dStatus="DONE"
dOriginalName="12.doc" dOutDate="" dDocID="24" dRevLabel="1" dProcessingState="Y"
dDocTitle="Soap Upload 2 Document" dID="12" dDocType="ADACCT"
dSecurityGroup="Public" dFileSize="19456" dFormat="application/msword">
<idc:field name="primaryFile:path">
c:/stellent/vault/~temp/1230750423.doc
</idc:field>
<idc:field name="dRawDocID">
23
</idc:field>
<idc:field name="changedSubjects">
documents,1019482656706
</idc:field>
<idc:field name="StatusCode">
0
</idc:field>
<idc:field name="soapFile:path">
c:/stellent/vault/~temp/1230750422.xml
</idc:field>
<idc:field name="xComments">

</idc:field>
<idc:field name="soapStartContentID">
SoapContent
</idc:field>
<idc:field name="refreshSubMonikers">
```

```
</idc:field>
<idc:field name="changedMonikers">

</idc:field>
<idc:field name="dActionDate">
4/22/02 1:31 PM
</idc:field>
<idc:field name="dActionMillis">
30263
</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="WebfilePath">
c:/stellent/weblayout/groups/public/documents/adacct/soapupload2~1.doc
</idc:field>
<idc:field name="StatusMessage">
Successfully checked in content item &#39;SoapUpload2&#39;.
</idc:field>
<idc:field name="refreshSubjects">

</idc:field>
<idc:field name="dConversion">
PASSTHRU
</idc:field>
<idc:field name="primaryFile">
C:/stellent/custom/Soap/JavaSamples/SoapClientUpload/soaptest.doc
</idc:field>
<idc:field name="dAction">
Checkin
</idc:field>
<idc:field name="refreshMonikers">

</idc:field>
<idc:field name="VaultfilePath">
c:/stellent/vault/adacct/12.doc
</idc:field>
</idc:document>
<idc:user dUser="sysadmin">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Check Out Content Item

The CHECKOUT_BY_NAME checks out the latest revision of the specified content item.

- Given a content item revision ID, this service attempts to locate the content item in the system and undo the checkout.
- The service fails if the content item does not exist in the system, if the content item is not checked out, or the user does not have sufficient privilege to undo the checkout.
- The most likely error is a content item name that does not exist. If this service is unable to execute, an error message is displayed to the user.

Note: This service only marks the content item as locked. It does not perform a download.

Required Parameters

These parameters must be specified:

Parameter	Description
dDocName	The content item identifier (Content ID).
IdcService	Must be set to CHECKOUT_BY_NAME.

Note: Do not confuse the Content ID (dDocName) with the internal content item revision identifier (dID). The *dID* is a generated reference to a specific rendition of a content item.

Optional Parameters

This optional parameter may be specified:

Parameter	Description
dDocTitle	The content item title.

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.oracle.com/IdcService/"
IdcService="CHECKOUT_BY_NAME">
<idc:document dDocName="soap_sample">
</idc:document>
</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.oracle.com/IdcService/"
IdcService="CHECKOUT_BY_NAME">
<idc:document dDocTitle="soap_sample" dID="10" dRevLabel="1" dDocAccount=""
dRevClassID="10" dDocName="soap_sample" dOriginalName="soap_sample.txt"
dSecurityGroup="Public">
<idc:field name="dActionMillis">
39964
</idc:field>
<idc:field name="refreshMonikers">

</idc:field>
<idc:field name="dActionDate">
4/22/02 12:20 PM
</idc:field>
<idc:field name="latestID">
10
</idc:field>
<idc:field name="refreshSubMonikers">

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

</idc:field>
<idc:field name="CurRevID">
10
</idc:field>
<idc:field name="CurRevIsCheckedOut">
0
</idc:field>
<idc:field name="dAction">
Check out
</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="CurRevCheckoutUser">
```

```
sysadmin
</idc:field>
<idc:field name="changedMonikers">

</idc:field>
<idc:field name="changedSubjects">
documents,1019482656687
</idc:field>
<idc:resultset name="DOC_INFO">
<idc:row dID="10" dDocName="soap_sample" dDocType="ADACCT" dDocTitle="soap_sample"
dDocAuthor="sysadmin" dRevClassID="10" dRevisionID="1" dRevLabel="1"
dIsCheckedOut="1" dCheckoutUser="sysadmin" dSecurityGroup="Public"
dCreateDate="4/22/02 12:18 PM" dInDate="4/22/02 12:18 PM" dOutDate=""
dStatus="RELEASED" dReleaseState="Y" dFlag1="" dWebExtension="txt"
dProcessingState="Y" dMessage="" dDocAccount="" dReleaseDate="4/22/02 12:19 PM"
dRendition1="" dRendition2="" dIndexerState="" dPublishType="" dPublishState=""
dDocID="19" dIsPrimary="1" dIsWebFormat="0" dLocation=""
dOriginalName="soap_sample.txt" dFormat="text/plain" dExtension="txt"
dFileSize="12">
<idc:field name="xComments">

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

Undo Content Item Checkout

The UNDO_CHECKOUT_BY_NAME service reverses a content item checkout using the Content ID.

- Given a content item name, this service attempts to locate the content item in the system and undo the checkout.
- The service fails if the content item does not exist in the system, if the content item is not checked out, or if the user does not have sufficient privilege to undo the checkout.
- This service is used by an applet or application.
- If this service is unable to execute, this message is displayed to the user: *Unable to undo checkout for "{dDocName}".*

Required Parameters

These parameters must be specified:

Parameter	Description
dDocName	The content item identifier (Content ID).
IdcService	Must be set to UNDO_CHECKOUT_BY_NAME.

Note: Do not confuse the Content ID (dDocName) with the internal content item revision identifier (dID). The *dID* is a generated reference to a specific rendition of a content item.

Optional Parameters

This optional parameter may be specified:

Parameter	Description
dDocTitle	The content item title.

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.oracle.com/IdcService/"
IdcService="UNDO_CHECKOUT_BY_NAME">
<idc:document dDocName="soap_sample">
</idc:document>
</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.oracle.com/IdcService/"
IdcService="UNDO_CHECKOUT_BY_NAME">
```

```
<idc:document dCheckoutUser="sysadmin" dPublishState="" dDocTitle="soap_sample"
dID="10" dRevLabel="1" dDocAccount="" dDocName="soap_sample" dRevClassID="10"
dOriginalName="soap_sample.txt" dSecurityGroup="Public">
<idc:field name="dActionMillis">
5317
</idc:field>
<idc:field name="refreshMonikers">

</idc:field>
<idc:field name="dActionDate">
4/22/02 12:23 PM
</idc:field>
<idc:field name="latestID">
10
</idc:field>
<idc:field name="refreshSubMonikers">

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

</idc:field>
<idc:field name="CurRevID">
10
</idc:field>
<idc:field name="CurRevIsCheckedOut">
1
</idc:field>
<idc:field name="dAction">
Undo Checkout
</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="CurRevCheckoutUser">
sysadmin
</idc:field>
<idc:field name="changedMonikers">

</idc:field>
<idc:field name="changedSubjects">
documents,1019482656689
</idc:field>
</idc:document>
<idc:user dUser="sysadmin">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Get Content Item Information

The DOC_INFO service retrieves content item revision information.

- Given a content item revision ID, the service retrieves content item revision information
- The most likely errors are when the content item no longer exists in the system or when the user does not have the security level to perform this action. If this service is unable to execute, an error message is displayed to the user.

Required Parameters

These parameters must be specified:

Parameter	Description
dID	The generated content item revision ID.
IdcService	Must be set to DOC_INFO.

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.oracle.com/IdcService/" IdcService="DOC_INFO">
<idc:document dID="6">
</idc:document>
</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.oracle.com/IdcService/" IdcService="DOC_INFO">
<idc:document dStatus="RELEASED" dDocFormats="text/plain" dID="6"
DocUrl="HTTP://wharristest/stellent/groups/public/documents/adacct/stellent.txt"
dDocTitle="stellent">
<idc:field name="dSubscriptionAlias">
sysadmin
</idc:field>
```

```
<idc:field name="changedSubjects">

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

</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="changedMonikers">

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

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

</idc:field>
<idc:field name="dSubscriptionType">
Basic
</idc:field>
<idc:resultset name="REVISION_HISTORY">
<idc:row dFormat="text/plain" dInDate="4/12/02 1:27 PM" dOutDate=""
dStatus="RELEASED" dProcessingState="Y" dRevLabel="1" dID="6" dDocName="stellent"
dRevisionID="1">
</idc:row>
</idc:resultset>
<idc:resultset name="WF_INFO">
</idc:resultset>
<idc:resultset name="DOC_INFO">
<idc:row dID="6" dDocName="stellent" dDocType="ADACCT" dDocTitle="stellent"
dDocAuthor="sysadmin" dRevClassID="6" dRevisionID="1" dRevLabel="1"
dIsCheckedOut="0" dCheckoutUser="" dSecurityGroup="Public" dCreateDate="4/12/02
1:27 PM" dInDate="4/12/02 1:27 PM" dOutDate="" dStatus="RELEASED"
dReleaseState="Y" dFlag1="" dWebExtension="txt" dProcessingState="Y" dMessage=""
dDocAccount="" dReleaseDate="4/12/02 1:27 PM" dRendition1="" dRendition2=""
dIndexerState="" dPublishType="" dPublishState="" dDocID="11" dIsPrimary="1"
dIsWebFormat="0" dLocation="" dOriginalName="stellent.txt" dFormat="text/plain"
dExtension="txt" dFileSize="8">
<idc:field name="xComments">
stellent
</idc:field>
</idc:row>
</idc:resultset>
</idc:document>
<idc:user dUser="sysadmin">
```

```
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Get File

The GET_FILE service returns a specific rendition of a content item, the latest revision, or the latest released revision. A copy of the file is retrieved without performing a check out.

- This command computes the dID (content item revision ID) for the revision, and then determines the filename of a particular rendition of the revision with the computed dID. A specified dID or a dDocName (content item name) along with a RevisionSelectionMethod parameter can be used.
- Given a dID or a dDocName along with a RevisionSelectionMethod parameter, the service determines the filename of a particular rendition of the revision and returns that file to the client.
- The most likely errors are some form of mismatched parameters or a request for a revision or rendition that does not exist. If this service is unable to execute, an error message is displayed to the user.

Note: It is recommended that *dDocName* be present in all requests for content items where the requester knows the *dDocName*. Error messages in the content server assume that it is present, as do other features such as forms.

Required Parameters

Important: Either the content item revision ID (*dID*) must be specified or a content item name (*dDocName*) along with a *RevisionSelectionMethod* parameter must be defined.

Parameter	Description
dDocName	The content item identifier (Content ID). ❖ If dDocName is not present, dID must be present and RevisionSelectionMethod must not be present. ❖ If RevisionSelectionMethod is present, a rendition of a revision of the content item with this name will be returned, if it exists. ❖ If RevisionSelectionMethod is not present, dDocName may be used in error messages.
dID	The generated content item revision ID. ❖ If dID is not specified, dDocName and RevisionSelectionMethod must specified. ❖ A rendition of the revision of the content item with this ID will be returned, if it exists, and the RevisionSelectionMethod parameter does not exist or has the value *Specific*.
RevisionSelection Method	The revision selection method. If present, dDocName must be present. The value of this variable is the method used to compute a dID from the specified dDocName. Its value may be *Specific*, *Latest*, or *LatestReleased*. ❖ If the value is *Specific*, the dDocName is ignored, and dID is required and is used to get a rendition. ❖ If the value is *Latest*, the latest revision of the content item is used to compute the dID. ❖ If the value is *LatestReleased*, the latest released revision of the content item is used to compute the dID.
IdcService	Must be set to GET_FILE.

Optional Parameters

These optional parameters may be specified:

Parameter	Description
Rendition	The content item rendition. This parameter specifies the rendition of the content item and can be set to *Primary*, *Web*, or *Alternate*. If Rendition is not present, it defaults to *Primary*. ❖ If the value is *Primary*, the primary rendition of the selected revision is returned. ❖ If the value is *Web*, the web viewable rendition of the selected revision is returned. ❖ If the value is *Alternate*, the alternate rendition of the selected revision is returned.

Note: Do not confuse the Content ID (dDocName) with the internal content item revision identifier (dID). The *dID* is a generated reference to a specific rendition of a content item.

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.oracle.com/IdcService/" IdcService="GET_FILE">
<idc:document dID="10">
</idc:document>
</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.oracle.com/IdcService/" IdcService="GET_FILE">
<idc:document dID="10">
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Receving response...
```

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Connection: keep-alive
Date: Mon, 29 Apr 2002 16:09:42 GMT
Content-type: Multipart/Related; boundary=----------------4002588859573015789;
type=text/xml; start="<SoapContent>"
Content-Length: 1717

------------------4002588859573015789
Content-Type: text/xml; charset=utf-8
Content-ID:  <SoapContent>

<?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.oracle.com/IdcService/" IdcService="GET_FILE">
<idc:document dID="10" dExtension="txt">
<idc:field name="changedSubjects">

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

</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="changedMonikers">

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

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

</idc:field>
<idc:resultset name="FILE_DOC_INFO">
<idc:row dID="10" dDocName="soap_sample" dDocType="ADACCT" dDocTitle="soap_sample"
dDocAuthor="sysadmin" dRevClassID="10" dRevisionID="1" dRevLabel="1"
dIsCheckedOut="0" dCheckoutUser="" dSecurityGroup="Public" dCreateDate="4/22/02
12:18PM" dInDate="4/22/02 12:18 PM" dOutDate="" dStatus="RELEASED"
dReleaseState="Y" dFlag1="" dWebExtension="txt" dProcessingState="Y" dMessage=""
dDocAccount="" dReleaseDate="4/22/02 12:19 PM" dRendition1="" dRendition2=""
dIndexerState="" dPublishType="" dPublishState="" dDocID="19" dIsPrimary="1"
dIsWebFormat="0" dLocation="" dOriginalName="soap_sample.txt" dFormat="text/plain"
dExtension="txt" dFileSize="12">
<idc:field name="xComments">

</idc:field>
</idc:row>
```

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

------------------4002588859573015789
Content-Type: text/html
Content-ID: <soap_sample.txt>

...File content...
------------------4002588859573015789--
```

Get Search Results

The GET_SEARCH_RESULTS service retrieves the search results for the passed query text.

- Used to display the search results to a user making a content item query.
- You can append values for Title, Content ID, etc., on the *QueryText* parameter to refine this service.

  The QueryText parameter defines the query. For use in a SOAP message, this query must be XML-encoded. This example passes a string submitted for a content item query as both a standard formatted string and XML-encoded format:

  - Parameter with standard formatted string.

    ```
    QueryText=dDocType <Substring> "ADSALES"
    ```

  - Parameter with XML-encoded string

    ```
    <idc:field name="QueryText">
    dDocType <Substring> &quot;ADSALES&quot;
    </idc:field>
    ```

Note: See Special Characters (page 4-6) for additional information on formatting XML-encoded strings.

- If this service is unable to execute, this message is displayed to the user: *Unable to retrieve search results.*

Required Parameters

These parameters must be specified:

Parameter	Description
IdcService	Must be set to GET_SEARCH_RESULTS.
QueryText	The user supplied text submitted for the content item query.

Optional Parameters

These parameters may be specified:

Parameter	Description
resultCount	The number of results to return, defaults to "25"
sortField	The name of the metadata field to sort on. ❖ Examples: dInDate, dDocTitle, Score. ❖ Defaults to dInDate.
sortOrder	The sort order. Allowed values are ASC (ascending) and DES (descending).
startRow	The row to begin the search results. For example, if a result returns 200 rows, and resultCount is 25, set startRow to 26 to obtain the second set of results.

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.oracle.com/IdcService/"
IdcService="GET_SEARCH_RESULTS">
<idc:document>
<idc:field name="QueryText">
dDocType <Substring> "ADSALES"
</idc:field>
</idc:document>
</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.oracle.com/IdcService/"
IdcService="GET_SEARCH_RESULTS">
<idc:document StartRow="1" TotalDocsProcessed="6" TotalRows="0"
QueryText="dDocType+%3cSubstring%3e+%22ADSALES%22" EndRow="25"
SearchProviders="Master_on_wharristest" NumPages="0" PageNumber="1">
<idc:field name="refreshMonikers">

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

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

</idc:field>
<idc:field name="EnterpriseSearchMaxRows">
4
</idc:field>
<idc:field name="FullRequest">
&QueryText=dDocType+%3cSubstring%3e+%22ADSALES%22
</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="changedMonikers">

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

</idc:field>
<idc:field name="Text2">
&lt;$dDocTitle$&gt;
</idc:field>
<idc:field name="Text1">
<$dDocName$>
</idc:field>
<idc:field name="OriginalQueryText">
dDocType+%3cSubstring%3e+%22ADSALES%22
</idc:field>
<idc:resultset name="SearchResults">
</idc:resultset>
<idc:resultset name="NavigationPages">
</idc:resultset>
```

```
<idc:resultset name="Master_on_wharristest">
</idc:resultset>
<idc:resultset name="EnterpriseSearchResults">
<idc:row ProviderName="Master_on_wharristest" IDC_Name="Master_on_wharristest"
TotalRows="0" TotalDocsProcessed="6">
<idc:field name="ProviderDescription">
!csProviderLocalContentServerLabel
</idc:field>
<idc:field name="InstanceMenuLabel">
Master_on_wharristest
</idc:field>
<idc:field name="InstanceDescription">
Master_on_wharristest
</idc:field>
<idc:field name="IntradocServerHostName">
wharristest
</idc:field>
<idc:field name="HttpRelativeWebRoot">
/stellent/
</idc:field>
<idc:field name="IsImplicitlySearched">

</idc:field>
<idc:field name="UserAccounts">
#all
</idc:field>
<idc:field name="IsLocalCollection">
true
</idc:field>
<idc:field name="Selected">

</idc:field>
<idc:field name="StatusMessage">
Success
</idc:field>
<idc:field name="ResultSetName">
Master_on_wharristest
</idc:field>
<idc:field name="SearchCgiWebUrl">
/idcplg/idc_cgi_isapi.dll/stellent/pxs
</idc:field>
</idc:row>
</idc:resultset>
</idc:document>
<idc:user dUser="sysadmin">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Get Table Data

The GET_TABLE service exports the specified table in the content server database.

- Exports the specified table by creating a result set and adding it to the serialized *hda*. If the table is not found, the service will fail. It is up to the calling program receiving the serialized *hda* to store this result set for later usage.
- The most likely error is a table name that does not exist. If this service is unable to execute, an error message is displayed to the user.

Required Parameters

These parameters must be specified:

Parameter	Description
IdcService	Must be set to GET_TABLE.
tableName	The name of table to export.

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.oracle.com/IdcService/" IdcService="GET_TABLE">
<idc:document>
<idc:field name="tableName">
DocTypes
</idc:field>
</idc:document>
</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.oracle.com/IdcService/" IdcService="GET_TABLE">
<idc:document>
<idc:field name="tableName">
DocTypes
```

```
</idc:field>
<idc:field name="changedSubjects">

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

</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="changedMonikers">

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

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

</idc:field>
<idc:resultset name="DocTypes">
<idc:row dDocType="ADACCT" dDescription="Acme Accounting Department"
dGif="adacct.gif">
</idc:row>
<idc:row dDocType="ADCORP" dDescription="Acme Corporate Department"
dGif="adcorp.gif">
</idc:row>
<idc:row dDocType="ADENG" dDescription="Acme Engineering Department"
dGif="adeng.gif">
</idc:row>
<idc:row dDocType="ADHR" dDescription="Acme Human Resources Department"
dGif="adhr.gif">
</idc:row>
<idc:row dDocType="ADMFG" dDescription="Acme Manufacturing Department"
dGif="admfg.gif">
</idc:row>
<idc:row dDocType="ADMKT" dDescription="Acme Marketing Department"
dGif="admkt.gif">
</idc:row>
<idc:row dDocType="ADSALES" dDescription="Acme Sales Department"
dGif="adsales.gif">
</idc:row>
</idc:resultset>
</idc:document>
<idc:user dUser="sysadmin">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Get Criteria Workflow Information

The GET_CRITERIA_WORKFLOWS_FOR_GROUP service returns criteria workflow information.

- Given a named security group, this service returns a list of workflows and related steps.
- Returns the result sets WorkflowsForGroup and WorkflowStepsForGroup:
  - WorkflowsForGroup lists all of the workflows for this group (dWfID, dWfName).
  - WorkflowStepsForGroup lists all of the steps in all of the workflows for this group (dWfID, dWfName, dWfStepID, dWfStepName).
- Criteria workflows and sub-workflows can be added, edited, enabled, disabled, and deleted from the Criteria tab of the Workflow Admin administration applet.
- The most likely error is a named security group that does not exist or a user failing the security check. The service throws reasonable exceptions for display to the user in these situations.

Required Parameters

These parameters must be specified:

Parameter	Description
dSecurityGroup	The security group such as PUBLIC or SECURE.
IdcService	Must be set to GET_CRITERIA_WORKFLOWS_FOR_GROUPS.

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.oracle.com/IdcService/"
IdcService="GET_CRITERIA_WORKFLOWS_FOR_GROUP">
<idc:document dSecurityGroup="Public" />
</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.oracle.com/IdcService/"
IdcService="GET_CRITERIA_WORKFLOWS_FOR_GROUP">
<idc:document dSecurityGroup="Public">
<idc:field name="changedSubjects">

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

</idc:field>
<idc:field name="loadedUserAttributes">
1
</idc:field>
<idc:field name="changedMonikers">

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

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

</idc:field>
<idc:resultset name="WorkflowStepsForGroup">
<idc:row>
<idc:field name="dWfID">
1
</idc:field>
<idc:field name="dWfName">
TestWorkflow
</idc:field>
<idc:field name="dWfStepID">
1
</idc:field>
<idc:field name="dWfStepName">
contribution
</idc:field>
</idc:row>
<idc:row>
<idc:field name="dWfID">
1
</idc:field>
<idc:field name="dWfName">
TestWorkflow
</idc:field>
<idc:field name="dWfStepID">
```

```
2
</idc:field>
<idc:field name="dWfStepName">
StepOne
</idc:field>
</idc:row>
</idc:resultset>
<idc:resultset name="WorkflowsForGroup">
<idc:row>
<idc:field name="dWfID">
1
</idc:field>
<idc:field name="dWfName">
TestWorkflow
</idc:field>
</idc:row>
</idc:resultset>
</idc:document>
<idc:user dUser="sysadmin">
</idc:user>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Appendix

B

SAMPLE SOAP CALLS

OVERVIEW

This appendix provides samples of SOAP calls. These samples are provided with the SOAP component and are stored in <*install_dir*>/custom/WsdlGenerator/samples.

Note: The actual sample files are not detailed here. See the /samples directory for the content of each file.

This appendix covers the following topics:

APACHE AXIS TOOLKIT

This set of samples provides ways of making SOAP requests to the Content Server via the Apache Axis API. In order to run the samples, you need to perform the following steps:

1. Download the Apache Axis libraries found at http://ws.apache.org/axis/. Download the Axis 1.4 libraries.
2. Extract the following files to the <*install_dir*>/custom/WsdlGenerator/samples/ApacheAxis/ directory:

   axis.jar, commons_logging.jar, commons_discovery.jar, wsdl4j.jar, saaj.jar, jaxrpc.jar

   Some of these files may have a version number associated with it (for example, wsdl4j.jar). Rename the files to one of the files listed above.
3. Download the following files from the locations specified:

   mail.jar from http://java.sun.com/products/javamail/downloads/index.html

   activation.jar from http://java.sun.com/products/javabeans/glasgow/jaf.html

   Add these files to the same directory as the .jar files in Step 2.

If the WSDL has not been modified in the Content Server, skip to step 7.

4. Copy the WSDL file for the corresponding sample (DocInfo, Search, etc.) from the <*install_dir*>/weblayout/groups/secure/wsdl/custom/ directory to the <*install_dir*>/custom/WsdlGenerator/samples/ApacheAxis/<*sample_dir*>.
5. Run the wsdl.bat batch file. This regenerates the client stub classes in your classes path (for example, ./samples/ApacheAxis/DocInfo/docinfo.)
6. Run the compile.bat batch file to compile all the Java classes.
7. Update the configuration values in the soap.cfg file. The following values can be set:
   - *CgiPath:* Allows you to point to a Content Server different from the one where you retrieved the WSDL. Set value to *default* to use the Content Server where the WSDL was retrieved. If you specify a CgiPath to point to a Content Server, make sure the WSDLs have matching parameters or the request may fail.
   - *User:* The Content Server user.
   - *Password:* The Content Server password for the specified user.

   The rest of the configuration values are service specific. See Configuration Values (page B-3) for details. These configuration variables are also found in the readme.txt file for each service.

8. Execute the run.bat batch file to run the program.

Note: The samples have been compiled with Apache Axis 1.4. If you use a different version, you may need to recompile the Java classes.

In version 1.4, the java class IdcPropertyList is not auto-generated when running the WSDL batch file. In functions where IdcPropertyList was previously returned, IdcProperty() is not returned. This is only an issue if you are using a previous version of the TestClient.java from this component with Apache Axis 1.4.

Four directories contain the sample files for the Apache Axis services:

- Checkin
- DocInfo
- GetFile
- Search

Within each subdirectory are directories which contain the .java, .class and .wsdl files for each service. See the individual directories for each service for a copy of each file.

Configuration Values

The following configuration values are set for Apache Axis and Apache SOAP requests. These values are service-specific.

CheckIn Service Configuration Values

- dDocName: The content ID for the content item. If auto ID is enabled, this can be blank.
- dDocTitle: The content title.
- dDocType: The content type.
- dDocAuthor: The content author.
- dSecurityGroup: The security group associated with the content item.
- dDocAccount: The account associated with the content item.
- CustomFields: A comma-delimited list of custom metadata fields.
- primaryFile: The primary file to be checked in.
- alternateFile: (optional) An alternate file to be checked in.

DocInfo Service Configuration Values

- DocInfoByField: If the value specified is dDocName, the content information is derived from dDocName. Otherwise the content information is derived from dID.
- dDocName: Specifies the content if DocInfoByField has a value of dDocName.
- dID: Specifies the content if DocInfoByField is dID.

GetFile Service Configuration Values

- GetFileByField: If the value specified is dDocName, the content information is derived from dDocName. Otherwise the content information is derived from dID.
- dDocName: Specifies the content if GetFileByField has a value of dDocName.
- dID: Specifies the content if DocInfoByField is dID.
- RevisionSelectionMethod: Used if GetFileByField has a value of dDocName and possible values are *latest* and *latestReleased*.
- Rendition: Specifies the rendition to download. Possible values are *primary*, *alternate*, and *web*.
- SaveDir: Specifies the directory where the file will be saved.

Search Service Configuration Values

- queryText: The query text of the search.
- sortField: The field by which results will be sorted.
- sortOrder: Specifies ascending (ASC) or descending (DESC) sort order.
- resultCount: The number of content items to return.

APACHE SOAP

This set of samples provides ways of making SOAP requests to the Content Server with the Apache Soap Toolkit. In order to run the samples, you need to perform the following steps:

1. Download the Apache Soap libraries from the specified locations:

   mail.jar from http://java.sun.com/products/javamail/downloads/index.html

   activation.jar from http://java.sun.com/products/javabeans/glasgow/jaf.html

soap.jar from http://ws.apache.org/soap/

2. Copy these jar files to the <*install_dir*>/custom/WsdlGenerator/samples/ApacheSoap/ directory.
3. Run the compile.bat file to compile the java class.
4. Update the soap.cfg file configuration values. The following values can be set:
   - *CgiPath:* Specifies the Content Server to request.
   - *User:* The Content Server user.
   - *Password:* The Content Server password for the specified user.

   The rest of the configuration values are service specific.See CheckIn Service Configuration Values (page B-3) and GetFile Service Configuration Values (page B-4) for details. These configuration variables are also found in the readme.txt file for each service.
5. Execute the run.bat batch file to run the program.

Two directories contain the sample files for services used with Apache Axis:

- Checkin
- GetFile

Within each subdirectory are directories which contain the .java and .class files for each service. See the individual directories for each service for a copy of each file.

ASP

The ASP samples in the <*install_dir*>/custom/WsdlGenerator/samples/ASP directory allow you to call the Content Server with ASP pages. This sample uses the Microsoft MSXML 4.0 library, which should be installed if you have loaded Internet Explorer 4.0 or later.

Two directories contain the sample files for services used with ASP:

- DocInfo
- Search

Within each subdirectory are request.asp and response.asp files. See the individual directories for each service for a copy of each file.

ASP.NET

The ASP sample in the <*install_dir*>/custom/WsdlGenerator/samples/ASP.NET directory uses the Microsoft Soap Toolkit to download content from the Content Server using the DIME message format.

The comments in the file indicate what values must be set in order to call the services correctly. For example, you must specify the Content Server Cgi path, the user name and password, the request parameters and connector properties.

DIME

The DIME samples in the <*install_dir*>/custom/WsdlGenerator/samples/DME directory allow you to call the Content Server with ASP pages. It uses the Microsoft MSXML 4.0 library, which should be installed if you have loaded Internet Explorer 4.0 or later. You will need the Microsoft Soap Toolkit 3.0 installed in order to run these samples.

Two directories contain the sample files for services used with DIME:

- Checkin
- GetFile

Within each subdirectory are the specific `.asp` files for each service. See the individual directories for each service for a copy of each file.

The comments in each ASP file indicate what values must be set in order to call the services correctly. For example, you must specify the Content Server Cgi path, the user name and password, the flag used to determine if the download should be done by ID or docName. See each file for the individual comments which specify the information needed.

For more information about the DIME specification, see http://www.gotdotnet.com/team/xml_wsspecs/dime/draft-nielsen-dime-01.txt

JAVA

Three directories in the <*install_dir*>/custom/WsdlGenerator/samples/Java directory contain the sample files for services used with Java:

- SoapClient (page B-7)

- SoapClientDownload (page B-7)
- SoapClientUpload (page B-8)

Before using any of these samples, check the entries in the soap.cfg file to verify that the values are correct for your Content Server:

```
Host=localhost
Port=80
CgiPath=/stellent/idcplg
User=sysadmin
Password=idc
```

SoapClient

This Java sample can be used to call services in the Content Server with the SOAP interface. The following sample XML files are included for use:

- checkout_by_name.xml
- doc_info.xml
- get_search_results.xml
- get_table.xml

The following command line parameters are used:

- The configuration file containing server settings such as port number and host name.
- The XML file containing the SOAP request to be passed to the Content Server.
- (Optional) The name of the log file containing the request and response data.

To call this program, use the following command line:

```
java SoapClient -c config_file -x xml_file -l log_file
```

For example, to call the doc_info sample with a log file called logfile.txt, use the following command line:

```
java SoapClient -c soap.cfg -x doc_info.xml -l logfile.txt
```

SoapClientDownload

This Java sample can be used to download files into the Content Server with the SOAP interface. The get_file.xml sample file is included for use.

The following command line parameters are used:

- The configuration file containing server settings such as port number and host name.

- The XML file containing the SOAP request to be passed to the Content Server.
- (Optional) The name of the log file containing the request and response data.

When using the get_file sample, change the dID value to the corresponding dID in the Content Server.

To call this program with a log file named logfile.txt, use the following command line:

```
java SoapClientDownload -c soap.cfg -x get_file.xml -l logfile.txt
```

SoapClientUpload

This Java sample can be used to upload files into the Content Server with the SOAP interface. The following sample XML files are included for use:

- checkin_universal.xml
- checkin_universal2.xml
- checkin_universal3.xml
- checkin_universal4.xml

The following command line parameters are used:

- The configuration file containing server settings such as port number and host name.
- The XML file containing the SOAP request to be passed to the Content Server.
- The file to be uploaded, known as the primary file.
- (Optional) The file to be uploaded as an alternate file.
- (Optional) The name of the log file containing the request and response data.

To call this program, use the following command line:

```
java SoapClientUpload -c config_file -x xml_file -p primary_file -a
alternate_file -l log_file
```

For example, to call the checkin_universal.xml sample with a log file called logfile.txt, use the following command line:

```
java SoapClientUpload -c soap.cfg -x checkin_universal.xml
-p soapfile.txt -l logfile.txt
```

VISUAL BASIC

The Visual Basic samples allow you to specify an XML request and returns the XML response. When doing an upload, you must specify the multipart/related request. When

doing a download, the response is in multipart/related form. These examples only work with plain text files.

You may need to register one or both of the OCX files in order to run the sample. To register the OCX file, run the following command:

```
regsvr32 <path to Visual Basic directory>\msinet.ocx
```

VISUALSTUDIO.NET

Note: If you are using the .NET samples to connect to a Content Server using the iPlanet60 Web Server, you may need to update your filter.

The VisualStudio.NET samples provide a way to call Content Server services through Visual Studio.NET client applications.

The program executable files are located in the bin/Debug directory of each service. The following sample services are included:

- CheckIn
- CheckInDime
- DocInfo
- GetFile
- GetFileDime
- Search

The CheckIn sample includes the use of the TimeOut property. This allows you to set the amount of time before the service call times out. The default is 100000 milliseconds. When checking in large files, you may need to increase this value.

The CheckInDime and GetFileDime samples use Dime attachments to upload and download files, rather than encode them inside the SOAP message. Because no encoding and decoding needs to be done on the files, the service calls are faster.

You should download and install the Web Services Enhancement 1.0 Service Pack 1 from the Microsoft web site. If you need the web reference for the WSDL, you will need to change the class that the proxy class inherits in the reference.cs file for the CheckIn and GetFile WSDLs. This is explained further in the WSE documentation “Adding Attachments to a SOAP Message Using Dime”.

The reference.cs files are stored in the following directories:

- <install_dir>/custom/WsdlGenerator/samples/VisualStudio.NET/CheckIn/Web References/CheckInRef
- <install_dir>/custom/WsdlGenerator/samples/VisualStudio.NET/GetFile/Web References/GetFileRef.

If you do not change the web references, the sample program should run after installing WSE 1.0.

Note: These samples must be used with Microsoft .NET Framework 1.1 or later versions. The Microsoft .NET Framework Version 1.1 Redistributable Package can be downloaded from Microsoft's web site.

Appendix

C

THIRD PARTY LICENSES

OVERVIEW

This appendix includes a description of the Third Party Licenses for all the third party products included with this product.

APACHE SOFTWARE LICENSE

```
* Copyright 1999-2004 The Apache Software Foundation.

* Licensed under the Apache License, Version 2.0 (the "License");

* you may not use this file except in compliance with the License.

* You may obtain a copy of the License at

*      http://www.apache.org/licenses/LICENSE-2.0

*
```

```
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
```

W3C® SOFTWARE NOTICE AND LICENSE

```
* Copyright © 1994-2000 World Wide Web Consortium,
* (Massachusetts Institute of Technology, Institut National de
* Recherche en Informatique et en Automatique, Keio University).
* All Rights Reserved.  http://www.w3.org/Consortium/Legal/
*
* This W3C work (including software, documents, or other related items) is
* being provided by the copyright holders under the following license. By
* obtaining, using and/or copying this work, you (the licensee) agree that
* you have read, understood, and will comply with the following terms and
* conditions:
*
* Permission to use, copy, modify, and distribute this software and its
* documentation, with or without modification, for any purpose and without
* fee or royalty is hereby granted, provided that you include the following
* on ALL copies of the software and documentation or portions thereof,
* including modifications, that you make:
*
*   1. The full text of this NOTICE in a location viewable to users of the
*      redistributed or derivative work.
*
*   2. Any pre-existing intellectual property disclaimers, notices, or terms
```

```
*      and conditions. If none exist, a short notice of the following form
*      (hypertext is preferred, text is permitted) should be used within the
*      body of any redistributed or derivative code: "Copyright ©
*      [$date-of-software] World Wide Web Consortium, (Massachusetts
*      Institute of Technology, Institut National de Recherche en
*      Informatique et en Automatique, Keio University). All Rights
*      Reserved. http://www.w3.org/Consortium/Legal/"
*
*   3. Notice of any changes or modifications to the W3C files, including the
*      date changes were made. (We recommend you provide URIs to the location
*      from which the code is derived.)
*
* THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS
* MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
* NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR
* PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE
* ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
*
* COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR
* CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR
* DOCUMENTATION.
*
* The name and trademarks of copyright holders may NOT be used in advertising
* or publicity pertaining to the software without specific, written prior
* permission. Title to copyright in this software and any associated
* documentation will at all times remain with copyright holders.
*
```

ZLIB LICENSE

```
* zlib.h -- interface of the 'zlib' general purpose compression library
  version 1.2.3, July 18th, 2005

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler
This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly jloup@gzip.org
  Mark Adler madler@alumni.caltech.edu
```

GENERAL BSD LICENSE

Copyright (c) 1998, Regents of the University of California

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

"Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

"Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

"Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

GENERAL MIT LICENSE

Copyright (c) 1998, Regents of the Massachusetts Institute of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

UNICODE LICENSE

UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE

Unicode Data Files include all data files under the directories http://www.unicode.org/Public/, http://www.unicode.org/reports/, and http://www.unicode.org/cldr/data/ . Unicode Software includes any source code published in the Unicode Standard or under the directories http://www.unicode.org/Public/, http://www.unicode.org/reports/, and http://www.unicode.org/cldr/data/.

NOTICE TO USER: Carefully read the following legal agreement. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2006 Unicode, Inc. All rights reserved. Distributed under the Terms of Use in http://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, (b) both the above copyright notice(s) and this permission notice appear in associated documentation, and (c) there is clear notice in each modified Data File or in the Software as well as in the documentation associated with the Data File(s) or Software that the data or software has been modified.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

_____Unicode and the Unicode logo are trademarks of Unicode, Inc., and may be registered in some jurisdictions. All other trademarks and registered trademarks mentioned herein are the property of their respective owners

MISCELLANEOUS ATTRIBUTIONS

Adobe, Acrobat, and the Acrobat Logo are registered trademarks of Adobe Systems Incorporated.

FAST Instream is a trademark of Fast Search and Transfer ASA.

HP-UX is a registered trademark of Hewlett-Packard Company.

IBM, Informix, and DB2 are registered trademarks of IBM Corporation.

Jaws PDF Library is a registered trademark of Global Graphics Software Ltd.

Kofax is a registered trademark, and Ascent and Ascent Capture are trademarks of Kofax Image Products.

Linux is a registered trademark of Linus Torvalds.

Mac is a registered trademark, and Safari is a trademark of Apple Computer, Inc.

Microsoft, Windows, and Internet Explorer are registered trademarks of Microsoft Corporation.

MrSID is property of LizardTech, Inc. It is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

Oracle is a registered trademark of Oracle Corporation.

Portions Copyright © 1994-1997 LEAD Technologies, Inc. All rights reserved.

Portions Copyright © 1990-1998 Handmade Software, Inc. All rights reserved.

Portions Copyright © 1988, 1997 Aladdin Enterprises. All rights reserved.

Index

I

J

M

N

O

P

R

S

U

V

W

X