STELLENT™

Using SOAP to Connect to Stellent Content Server

SDK-EN10-610

Table of Contents

CHAPTER 1: OVERVIEW

CHAPTER 2: SOAP INTEGRATION

CHAPTER 3: STELLENT SOAP CLIENTS

CHAPTER 4: SOAP SERVICE CALLS

CHAPTER 5: ASP INTEGRATION

Chapter

1

OVERVIEW

INTRODUCTION

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

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

This chapter contains these topics:

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

ABOUT THIS GUIDE

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

This guide provides this information:

- The steps required to install and enable the SOAP component.
- Information on calling services.
- The procedures for using the Visual Basic SOAP Client.
- The procedures for using the Java SOAP Clients.
- Sample XML-formatted service calls.
- A sample Active Server Page with an encapsulated SOAP packet.

Audience

This guide is intended for application developers who need to access Stellent Content Server services using the Simple Object Access Protocol (SOAP) interface.

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 the Stellent Content Server instance 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.

STELLENT PRODUCT DISTINCTIONS

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

Stellent Content Management Product and Feature Distinction

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

If You Need Assistance

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

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

Support Options

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

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

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

Before Contacting Support

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

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

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

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

Telephone

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

E-Mail

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

Internet

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

Chapter

2

SOAP INTEGRATION

INTRODUCTION

This chapter provides general information on the SOAP protocol and describes how to install the Stellent SOAP Component version 6.1.

This chapter contains these topics:

- The SOAP Protocol
- SOAP Component Installation

THE SOAP PROTOCOL

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

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

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

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

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

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

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

SOAP COMPONENT INSTALLATION

Installation Steps

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

Note: The Stellent SOAP component is provided on the Stellent Content Server CD-ROM in the *extras/Soap/* directory and is bundled as the *Soap.zip* file.

The SOAP component can be installed on your Stellent Content Server instance using the Component Wizard or the Admin Server Component Manager web page.

Follow these steps to install the Soap.zip file from the Component Manager web page:

1. Log into the content server the system administrator and click **Administration**.
2. Under *Administration Pages*, click **Admin Server**.
3. Click the desired content server instance (e.g., Master_on_testserver).
4. Under *Options*, click **Component Manager**.

   The *Component Manager* screen is displayed.
5. In the Upload New Component section, click **Browse**.
6. Navigate to the Soap.zip component and click **Open**.
7. Click **Upload**.
8. Select **Soap** from the *Disabled Components* list and click **Enable**.

9. Under *Options*, click **Stop/Start Content Server**.

   The *Content Server Current Status* screen is displayed.

10. Stop and restart the content server.

Note: A WSDL file is included with the SOAP component. The <soap:address location> attribute is dependent on the location of the content service. A generic version is being used for this value.

Chapter

3

STELLENT SOAP CLIENTS

INTRODUCTION

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

This chapter contains these topics:

- Accessing the Stellent SOAP Clients
- Using the Visual Basic SOAP Client
- Using the Java SOAP Client

ACCESSING THE STELLENT SOAP CLIENTS

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

For example (Visual Basic SOAP Client):

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

For example (Java SOAP Clients):

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

USING THE VISUAL BASIC SOAP CLIENT

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

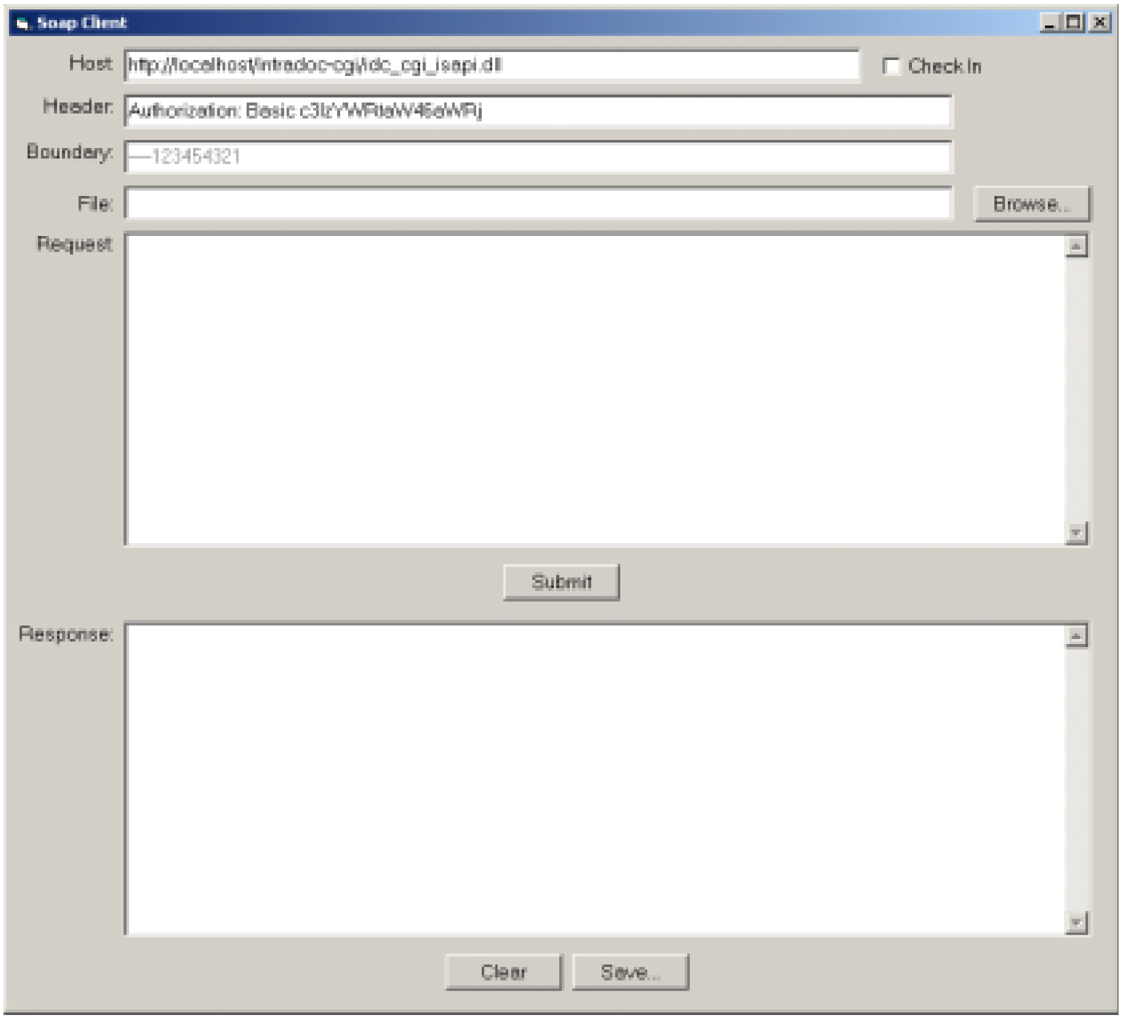
To run the Visual Basic SOAP Client, you must have Microsoft Visual Studio installed or the provided MSINET.OCX file registered. The file MSINET.OCX is located in the *VBSoapClient/* directory. Register the OCX by running this command:

```
regsvr32 <install_dir>/custom/SOAP/SoapClient/msinet.ocx
```

Follow these steps to use the VBSoapClient:

1. Launch the SoapClient.exe executable file (see Figure 3-1).

   The Soap Client interface is displayed.



Host: http://localhost/intradoc-cgi/idc_cgi_isapi.dll

□ Check In

Header: Authorization: Basic c3lzYWRtaW46aWRj

Boundary: ---123454321

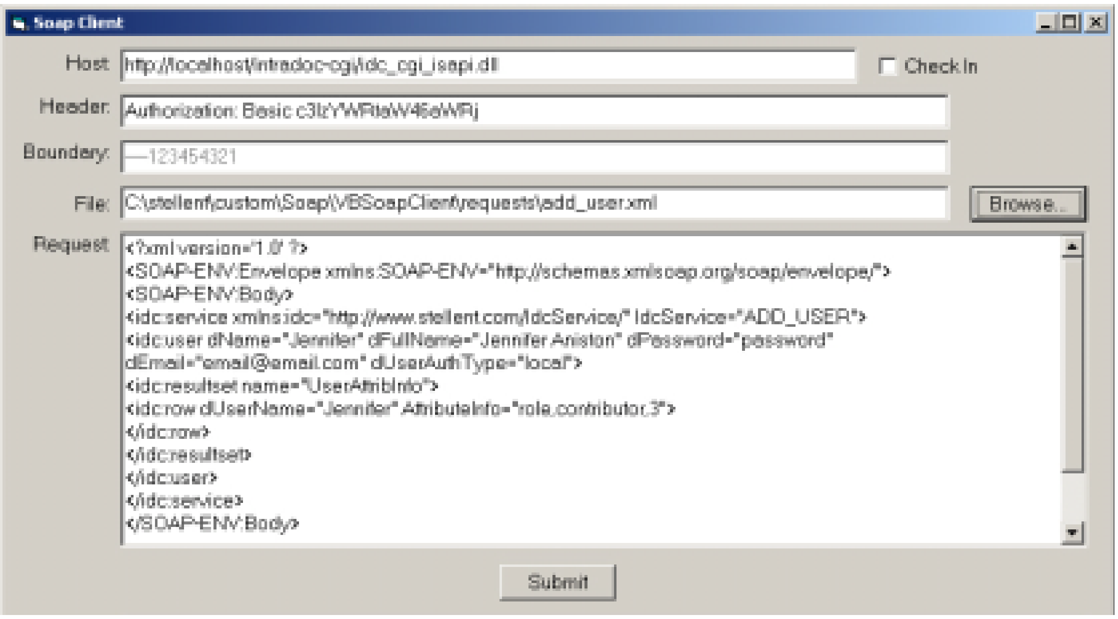
File:

Figure 3-1 The Soap Client interface.

2. In the *Request* text area, define a SOAP packet in an XML-formatted document (see Figure 3-2).

   ❖ Define the content server service to execute and the required (and optional) parameters.

❖ Alternatively, an XML document containing service and parameter information can be opened by clicking **Browse** and navigating to the file.



Host: http://localhost/intradoc-cgi/idc_cgi_isapi.dll

□ Check In

Header: Authorization: Basic c3lzYWRtaW46aWRj

Boundary: —123454321

File: C:\stellent\custom\Soap\VBSoapClient\requests\add_user.xml

Request:

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

Figure 3-2 The Soap Client interface—with defined service.

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

Note: The unpackaged ZIP file for the SOAP component contains a number of sample files containing pre-defined services. These files are located in the *VBSoapClient/requests/* directory.

Note: See *Chapter 4: SOAP Service Calls* for additional information on defining services and parameters in an XML document.

USING THE JAVA SOAP CLIENT

This section references the three Java SOAP programs provided with the installed SOAP component. These are the provided sample Java programs:

Java Program	Description
SoapClient (page 3-7)	This Java program allows you to call services in the content server using the SOAP interface. The .java and .class files for this program are located in the *JavaSamples/SoapClient/* directory.
SoapClientUpload (page 3-9)	This Java program allows you to upload files to the content server using the SOAP interface. The .java and .class files for this program are located in the *JavaSamples/SoapClientUpload/* directory.
SoapClientDownload (page 3-12)	This Java program allows you to download files from the content server using the SOAP interface. The .java and .class files for this program are located in the *JavaSamples/SoapClientDownload/* directory.

Note: Each directory contains a number of sample XML files with pre-defined services. In many cases, these services will need to be edited to meet your particular application.

Command Line Parameters

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 be passed to the content server.
-p <primary file>	The file name to be uploaded as the primary file.
-a <alternate file>	The file name to be uploaded as the alternate file (optional).
-l <log file>	The name of the file containing the request and response data (optional).

For example:

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

Note: 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 uing this program.

To run the SoapClient program:

- Define the onfiguration file properties.
- Define the XML file properties.
- Run the service from the command line.

Note: Refer to the *IdcCommand—Java Command Utility Reference Guide* for a list of available services and the required parameters.

Define 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 (see Figure 3-3).

```
Host=testserver
Port=80
CgiPath=/intradoc-cgi/idc_cgi_isapi.dll
User=sysadmin
Password=idc
```

Figure 3-3 The SoapClient configuration file.

Define 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 (see Figure 3-4).

```
<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://
schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<idc:service xmlns:idc="http://www.stellent.com/
IdcService/" IdcService="DOC_INFO">
<idc:document dID="6">
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 3-4 Example SoapClient XML file.

Note: See *Chapter 4: SOAP Service Calls* for additional information on defining services and parameters in an XML document.

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 parameters (see Figure 3-5).

```
C:/stellent/custom/Soap/JavaSamples/SoapClientDownload>
JAVA SoapClient -c soap.cfg -x doc_info.xml -l
logfile.txt
```

Figure 3-5 Example command line arguments.

SoapClientUpload

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

To run the SoapClientUpload program:

- Define the onfiguration file properties.
- Define the XML file properties.
- Run the service from the command line.

Note: Refer to the *IdcCommand—Java Command Utility Reference Guide* for a list of available services and the required parameters.

Define 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 (see Figure 3-6).

```
Host=testserver
Port=80
CgiPath=/intradoc-cgi/idc_cgi_isapi.dll
User=sysadmin
Password=idc
```

Figure 3-6 The SoapClientUpload configuration file.

Define 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 (see Figure 3-7).

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

Figure 3-7 Example SoapClientUpload XML file.

Note: See *Chapter 4: SOAP Service Calls* for additional information on defining services and parameters in an XML document.

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 (see Figure 3-8).

```
C:/stellent/custom/Soap/JavaSamples/SoapClientUpload>JAVA
SoapClientUpload -c soap.cfg -x checkin_universal2.xml -p
C:/stellent/custom/Soap/JavaSamples/SoapClientUpload/
soaptest.doc
```

Figure 3-8 Example command line arguments.

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:

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

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

To run the SoapClientDownload program:

- Define the CLASSPATH entry.
- Define the onfiguration file properties.
- Define the XML file properties.
- Run the service from the command line.

Define CLASSPATH Entry

To execute the GET_FILE service the SoapClientDownload application uses some content server classes. To access these classes the *client.zip* file must be included as a CLASSPATH environment variable (see Figure 3-9).

Note: In most configurations the *client.zip* file is located in the weblayout/ common/ directory.

A CLASSPATH entry can be defined from the command line or from the *Windows Control Panel—System Properties—Advanced tab—System Variables* section.

❖ If a CLASSPATH environment variable is defined, append the complete path of the *client.zip* file to the existing entries.

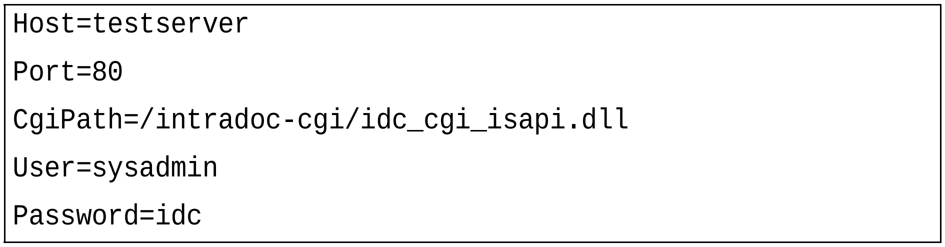
❖ If a CLASSPATH environment variable is not defined, create a new system variable named CLASSPATH that defines the complete path to the *client.zip* file (click **New** on the *Advanced tab—System Variables* section).

```
C:\stellent\weblayout\common\client.zip;
```

Figure 3-9 Example CLASSPATH entry.

Define 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 (see Figure 3-10).



```
Host=testserver
Port=80
CgiPath=/intradoc-cgi/idc_cgi_isapi.dll
User=sysadmin
Password=idc
```

Figure 3-10 The SoapClientDownload configuration file.

Define 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 (see Figure 3-11).

```
<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://
schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<idc:service xmlns:idc="http://www.stellent.com/
IdcService/" IdcService="GET_FILE">
<idc:document dID="10">
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 3-11 Example SoapClientDownload XML file.

Note: See *Chapter 4: SOAP Service Calls* for additional information on defining services and parameters in an XML document.

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 (see Figure 3-12).

```
C:/stellent/custom/Soap/JavaSamples/SoapClientDownload>
JAVA SoapClientDownload -c soap.cfg -x get_file.xml
```

Figure 3-12 Example command line arguments.

Chapter

4

SOAP SERVICE CALLS

INTRODUCTION

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

This chapter contains these topics:

- ❖ Calling Services with SOAP
- ❖ Sample Service Calls

CALLING SERVICES WITH SOAP

Stellent IdcCommand services can be executed using the SOAP interface. The user must have appropriate permissions to execute the commands. Some commands will require administrative access, other commands may require only write permission.

Note: The SOAP component must be installed and enabled to call services. See *Chapter 2: SOAP Integration* for additional information.

Note: Refer to the *IdcCommand—Java Command Utility Reference Guide* for a list of available services and the required parameters.

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.stellent.com/IdcService"
```

idc Namespace

We define a namespace called “idc” that explains the schema and allowable tags for our SOAP content. The following nodes may be required for a service request or may be returned in the response:

- service
- document
- user
- optionlist
- option
- resultset
- row
- field

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 will still process the data correctly. The response will put the data in the correct node.

Service Node

This is the main node of the idc namespace (see Figure 4-1).

- 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.stellent.com/
IdcService/" IdcService="PING_SERVER">

</idc:service>
```

Figure 4-1 Service node example using the PING_SERVER service.

Document Node

This snode contains all content item information (see Figure 4-2).

- 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 may be used as attributes.

For example:

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

Figure 4-2 Document node example using the CHECKOUT_BY_NAME service.

User Node

This is the node to contain all user information (see Figure 4-3).

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

For example:

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

Figure 4-3 User node example using the GET_USER_INFO service.

Optionlist Node

This is the node to contain any option lists (see Figure 4-4).

- The *name* attribute specifies the name of the option list.

For example:

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

Figure 4-4 Optionlist node example using the language localization option list.

Option Sub-Node

This sub-node is specified within the <optionlist> node (see Figure 4-5).

- 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>
```

Figure 4-5 Option sub-node example using the content item type option list.

Resultset Sub-Node

This sub-node can be specified within a <document> or <user> node (see Figure 4-6).

- 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>
```

Figure 4-6 Resultset sub-node example returning a revision history.

Row Sub-Node

This sub-node is specified within a <resultset> sub-node (see Figure 4-7).

- 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 particular 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>
```

Figure 4-7 Row sub-node example defining user attribute information.

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* (see Figure 4-8).
- 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* (see Figure 4-9).

For example (repesenting data):

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

Figure 4-8 Field sub-node example returning a subscrition ID.

For example (passing search result values):

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

Figure 4-9 Field sub-node example passing a content item query string.

Special Characters

To pass special characters such as ’<’ or ’>’ the XML-encoding format must be used:

Standard Format	XML-Encoding
<	&lt;
>	&gt;
"	&quot;
&	&amp;
\	&#39;

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 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 &lt;Substring&gt; &quot;ADSALES&quot;
  </idc:field>
  ```

SAMPLE SERVICE CALLS

This section presents information on executing specific content server services and provides examples of formatting the required and optional parameters for these services in a SOAP request.

Note: Refer to the *IdcCommand—Java Command Utility Reference Guide* for a list of available services and the required parameters.

These IdcCommand services are used as SOAP request examples:

IdcCommand	Description
PING_SERVER	This service evaluates whether a connection to the server exists (page 4-11).
ADD_USER	This service adds a new user to the system (page 4-13).
EDIT_USER	This service edits an existing user (page 4-19).
GET_USER_INFO	This service retrieves the user list (page 4-25).
DELETE_USER	This service deletes an existing user (page 4-28).

IdcCommand	Description
CHECKIN_ UNIVERSAL	This service performs a content server controlled check in (page 4-30).
CHECKOUT_BY_ NAME	This service marks the latest revision of the specified content item as locked (page 4-39).
UNDO_CHECKOUT_ BY_NAME	This service reverses a content item checkout using the Content ID (page 4-43).
DOC_INFO	This service retrieves content item revision information (page 4-47).
GET_FILE	This service retrieves a copy of a content item without performing a check out (page 4-51).
GET_SEARCH_ RESULTS	This service retrieves the search results for the passed query text (page 4-57).
GET_TABLE	This service exports the specified table in the content server database (page 4-63).
GET_CRITERIA_ WORKFLOWS_FOR_ GROUP	This service returns criteria workflow information (page 4-67).

Ping the Server

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

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

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

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.stellent.com/IdcService/
" IdcService="PING_SERVER">
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope
```

Response

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

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

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

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

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

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

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 authoization 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 have 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,accoun
t,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.stellent.com/IdcService/
" IdcService="ADD_USER">
<idc:user dName="Jennifer" dFullName="Jennifer Aniston"
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.stellent.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 Aniston" 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 not having 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 authoization 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.
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 Information	Description
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 have 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,accoun
t,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.stellent.com/IdcService/
" IdcService="EDIT_USER">
<idc:user dName="Jennifer" dFullName="Jennifer Aniston"
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.stellent.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 Aniston" 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.stellent.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.stellent.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 displayed to the user.

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.stellent.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.stellent.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>
```

Checkin Content Item

The CHECKIN_UNIVERSAL service performs a content server controlled check in.

❖ 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 it 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.
dDocType	The content item type.
doFileCopy	Set this flag to TRUE (1) or the file will be removed from your hard drive.
dSecurityGroup	The security group such as PUBLIC or SECURE.
IdcService	Must be set to CHECKIN_UNIVERSAL.

Parameter	Description
primaryFile	The absolute path to the location of the file as seen from the server. Use the forward slash as the file separator. 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.

Parameter	Description
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.stellent.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.stellent.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>
```

Checkout 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.stellent.com/IdcService/
" IdcService="CHECKOUT_BY_NAME">
<idc:document dDocName="soapy_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.stellent.com/IdcService/
" IdcService="CHECKOUT_BY_NAME">
<idc:document dDocTitle="soapy_sample" dID="10"
dRevLabel="1" dDocAccount="" dRevClassID="10"
```

```
dDocName="soapy_sample" dOriginalName="soapy_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="soapy_sample" dDocType="ADACCT"
dDocTitle="soapy_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="soapy_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 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.stellent.com/IdcService/
" IdcService="UNDO_CHECKOUT_BY_NAME">

<idc:document dDocName="soapy_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.stellent.com/IdcService/
" IdcService="UNDO_CHECKOUT_BY_NAME">
<idc:document dCheckoutUser="sysadmin" dPublishState=""
dDocTitle="soapy_sample" dID="10" dRevLabel="1"
dDocAccount="" dDocName="soapy_sample" dRevClassID="10"
dOriginalName="soapy_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.stellent.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.stellent.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**
IdcService	Must be set to GET_FILE.

Optional Parameters

These optional parameters may be specified:

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*.

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.
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.

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.stellent.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.stellent.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.stellent.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="soapy_sample" dDocType="ADACCT"
dDocTitle="soapy_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="soapy_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: <soapy_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 &lt;Substring&gt; &quot;ADSALES&quot;
</idc:field>
```

Note: See *Special Characters* (page page 4-8) 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).

Parameter	Description
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.stellent.com/IdcService/
" IdcService="GET_SEARCH_RESULTS">
<idc:document>
<idc:field name="QueryText">
dDocType &lt;Substring&gt; &quot;ADSALES&quot;
</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.stellent.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">
&amp;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">
&lt;$dDocName$&gt;
</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">
/intradoc-cgi/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.stellent.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.stellent.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.stellent.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.stellent.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>
```

Chapter

5

ASP INTEGRATION

INTRODUCTION

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

CALLING A STELLENT SERVICE FROM AN ACTIVE SERVER PAGE

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.

The embedded SOAP request forms the basis of the Active Server Page and executes the GET_SEARCH_RESULTS service and returns a list of content items checked in under the ADSALES content item type (dDocType).

Note: Refer to the *IdcCommand—Java Command Utility Reference Guide* for a list of available services and the required parameters.

Sample Stellent Service

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 in both URL-encoded and XML-encoded format:

  - Parameter with URL-encoded string.

    ```
    QueryText=dDocType <Substring> “ADSALES”
    ```

  - Parameter with XML-encoded string

    ```
    <idc:field name=”QueryText”>
    ```

```
dDocType &lt;Substring&gt; &quot;ADSALES&quot;
</idc:field>
```

Note: See *Special Characters* (page 4-8) 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).

Parameter	Description
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.

Embedded SOAP Request

```
<?xml version=’1.0’ ?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<idc:service
xmlns:idc="http://www.stellent.com/IdcService/"
IdcService="GET_SEARCH_RESULTS">
<idc:document>
<idc:field name="QueryText">
dDocType &lt;Substring&gt; &quot;ADSALES&quot;
</idc:field>
</idc:document>
</idc:service>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample Active Server Page

```
<%@ Language=VBScript %>
<%
Option explicit

’ SOAP GET_SEARCH_RESULTS
’ sample search ASP page using SOAP and DOM

With Response
   .Buffer=true
   .ContentType = "text/html"
End With

Dim oXml, oNodes, oNode, N
Dim strUser, strPwd, strSoapServer,strStatusMsg, strReq,
strStellentBaseURL, dDocType
Dim xmldom, xmlhttp

’ User
strUser = "sysadmin"
’ Password
strPwd = "idc"

’ change host here
strSoapServer = "http://testserver/intradoc-
cgi/idc_cgi_isapi.dll"
’ change host here
```

```
strStellentBaseURL = "http://testserver"

strStatusMsg = ""

’ Build SOAP Request
strReq = "<?xml version=’1.0’ ?>" + vbCrLf
strReq = strReq + "<SOAP-ENV:Envelope xmlns:SOAP-ENV=" +
Chr(34) + "http://schemas.xmlsoap.org/soap/envelope/" +
Chr(34) + "> " + vbCrLf
strReq = strReq + "<SOAP-ENV:Body>" + vbCrLf
strReq = strReq + "<idc:service xmlns:idc=" + Chr(34) +
"Http://www.stellent.com/IdcService/" + Chr(34) +
" IdcService=" + Chr(34) + "GET_SEARCH_RESULTS" + Chr(34) +
">" + vbCrLf
strReq = strReq + "<idc:document>" + vbCrLf
strReq = strReq + "<idc:field name=" + Chr(34) +
"QueryText" + Chr(34) + ">dDocType &lt;Substring&gt;
&quot;ADSALES&quot;</idc:field>" + vbCrLf
strReq = strReq + "</idc:document>" + vbCrLf
strReq = strReq + "</idc:service>" + vbCrLf
strReq = strReq + "</SOAP-ENV:Body>" + vbCrLf
strReq = strReq + "</SOAP-ENV:Envelope>" + vbCrLf

’ Send SOAP Request
set xmldom = server.CreateObject("Microsoft.XMLDOM")
set xmlhttp = server.CreateObject("Microsoft.XMLHTTP")

With xmlhttp
```

```
   .open "POST", strSoapServer, false, strUser, strPwd
   .setRequestHeader "Content-Type", "text/xml;
   charset=utf-8"
   .setRequestHeader "SOAPAction",
   "http://www.stellent.com/IdcService"
   .send(strReq)
End With

’ Error Check response
strStatusMsg = xmlhttp.Status
if xmlhttp.Status = 200 then ’405 means error, 200 means
ok.

’ Load and parse SOAP Response
Set oXml = server.CreateObject ("Microsoft.XMLDOM")
With oXml
   .Async = False
   .validateOnParse = False
   .Load xmlhttp.responseXML
End With

’ Use DOM to get SearchResults node,
’ pull Attrs from Row, and display results
   dim x, y, rsName
   dim dDocTitle, URL

for each y in oXml.documentElement.selectNodes("SOAP-
ENV:Body/idc:service/idc:document/idc:resultset")
```

```
   rsName = y.getAttribute("name")
   if rsName = "SearchResults" then
   for each x in y.selectNodes("idc:row")
      dDocTitle = x.getAttribute("dDocTitle")
      URL = x.getAttribute("URL")
      Response.Write("<a href="+ chr(34) +
      strStellentBaseURL + URL + chr(34)+">")
      response.write(dDocTitle)
      Response.Write("</a><br>")
   next
   end if
next

Else
   Response.Write("Didn’t Work<BR>")
   Response.Write("status="&xmlhttp.status)
   Response.write("<BR>"&xmlhttp.statusText)
   Response.Write("<BR>"&Request.ServerVariables("ALL_HTTP"
   ))
End if

set xmlhttp = nothing
set xmldom = nothing
%>
```

Index