Content Services 0.9

Public Review Draft 18 March 2002

This version:
www.content-services.org/specification_v0.9.html
Editor:
Ron Daniel, Jr. , Interwoven, Inc. <rdaniel@interwoven.com>

Abstract

Content Services (CS) is a specification for providing core content infrastructure functionality in the form of web services. Exposing those capabilities as web services offers the advantage of language and platform independence.

Status of this Document

This is the first public release of the Content Services specification. We will continue to pursue broader adoption of this standard, your comments are an important part of that. Please send your comments on the draft to spec-comments@content-services.org.

Acknowledgements

The Editor would like to thank the following people for the early and continuing development of the CS interfaces, and for all their help in editing the CS specification: Karen Auman, Allan Banaag, Sujata Bopardikar, Elham Chandler, Mark Hale, James Koh, Betty Koon, Navneet Joneja, Tom Lehmann, Kapil Mehrotra, Eric Melski and Eric Ye.

Table of Contents

1 Introduction 4
1.1 Audience 4
1.2 Notational Conventions 4
1.3 Namespace Definitions 4
2 Architecture 5
2.1 Key Terms 5
3 Services 7
3.1 Conventions 7
3.2 Session Management 7
3.2.1 startSession 7
3.2.2 startSessionResponse 8
3.2.3 endSession 8
3.2.4 endSessionResponse 8
3.2.5 ping 8
3.2.6 pingResponse 9
3.3 File Management 9
3.3.1 getDirectoryContents 9
3.3.2 getDirectoryContentsResponse 9
3.3.3 getFiles 9
3.3.4 getFilesResponse 10
3.3.5 getModifiedFiles 10
3.3.6 getModifiedFilesResponse 10
3.3.7 readFile 10
3.3.8 readFileResponse 11
3.3.9 writeFile 11
3.3.10 writeFileResponse 11
3.3.11 deleteFiles 12
3.3.12 deleteFilesResponse 12
3.3.13 getExtendedAttributes 12
3.3.14 getExtendedAttributesResponse 12
3.3.15 setExtendedAttributes 13
3.3.16 setExtendedAttributesResponse 13
3.4 Collaboration 13
3.4.1 editFile 13
3.4.2 editFileResponse 14
3.4.3 importFiles 14
3.4.4 importFilesResponse 15
3.4.5 uploadFiles 15
3.4.6 uploadFilesResponse 15
3.4.7 undoChanges 15
3.4.8 undoChangesResponse 16
3.4.9 lockFiles 16
3.4.10 lockFilesResponse 16
3.4.11 unlockFiles 16
3.4.12 unlockFilesResponse 17
3.4.13 update 17
3.4.14 updateResponse 17
3.4.15 submit 17
3.4.16 submitResponse 18
3.4.17 submitDirect 18
3.4.18 submitDirectResponse 18
3.5 Templating 19
3.5.1 getTemplateCategories 19
3.5.2 getTemplateCategoriesResponse 19
3.5.3 getWebDeskNewDcrHelper 19
3.5.4 getWebDeskNewDcrHelperResponse 19

3.6 Workflow 19
3.6.1 getTasks 20
3.6.2 getTasksResponse 20
3.6.3 getTasksWithoutFiles 20
3.6.4 getTasksWithoutFilesResponse 20
3.6.5 addTaskComment 20
3.6.6 addTaskCommentResponse 21
3.6.7 addTaskFileComment 21
3.6.8 addTaskFileCommentResponse 21
3.6.9 addTaskFiles 21
3.6.10 addTaskFilesResponse 22
3.6.11 removeTaskFiles 22
3.6.12 removeTaskFilesResponse 22
3.6.13 finishTask 22
3.6.14 finishTaskResponse 22
3.6.15 createWorkflow 23
3.6.16 createWorkflowResponse 23
3.7 Utility 23
3.7.1 getVersionString 23
3.7.2 getVersionStringResponse 23
4 Bibliography 23

Appendices

A Task Query Grammar 24
B WSDL Definition 25
C Workflow Grammar 47
D File Status Codes 50

1 Introduction

Content management encompasses the capabilities required by an organization to maintain integrity of all assets consumed by users and applications across the organization. Content management functional requirements are typically bound to asset type: HTML requires different maintenance capability, due to the test requirements of hyperlinks, than business documents, which are treated in their entirety. There are many asset categories including: Web pages, code, documents, video, images, audio, presentations, and XML. The definition varies by organization; however, typically the assets are created by various specialists with different skills.

As the number of contributors grows, so do the functional demands placed on the content management system. An enterprise content management system must handle a number of different tasks: It must provide services for basic management functions such as the creation and deletion of assets; it must allow for asset sharing with basic audit capability; and it should make it easy for people to create content. For content to be progressively developed, versions are tracked allowing changes to be undone and state to be restored. In addition, a content management system must provide workflow functionality so content can be developed in a repeatable production process with appropriate approvals for quality control, legal compliance, etc. Finally, it should provide methods for enhancing the content with rich metadata in order to allow for the classification of content and to simplify the automated handling of the content by other applications. A content management system must also provide functions to distribute the content to a variety of destinations, in any one of a number of formats, using any one of a number of protocols.

One can view the functions described above as services to be provided by the content infrastructure. Adopting this viewpoint makes it very natural to expose those capabilities as Web Services. Through a web service interface, the content management system responds to requests sent to it over the SOAP and HTTP protocols. The actual content of the requests and responses is carried either in the body of the SOAP message or as an attachment.

Exposing a content management system's functionality in this manner has a number of advantages. The greatest is that services can be provided to many applications independent of hardware, operating system, and programming language boundaries. Application debugging is simplified since requests and responses are sent as human-readable XML messages.

This document specifies the interfaces to an initial set of Content Services. The interfaces are defined by a WSDL (Web Services Description Language) document, provided in Appendix B. That is the normative portion of this specification. If any conflicts are detected between Appendix B and the rest of the document, Appendix B must be followed.

Section 2 briefly reviews the architecture which is assumed to underlie the services defined in this specification. Section 3, the bulk of the document, defines the various messages accepted by the different content services, and the responses they will return. A bibliography is provided in section 4.

1.1 Audience

This document is intended for the use of:

- Developers of applications which will be clients of the various services defined in this specification.
- Client-side API developers who wish to provide an alternative language binding to invoke these services.
- Content service developers who want their services to be accessible by a broad range of clients.

1.2 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119].

1.3 Namespace Definitions

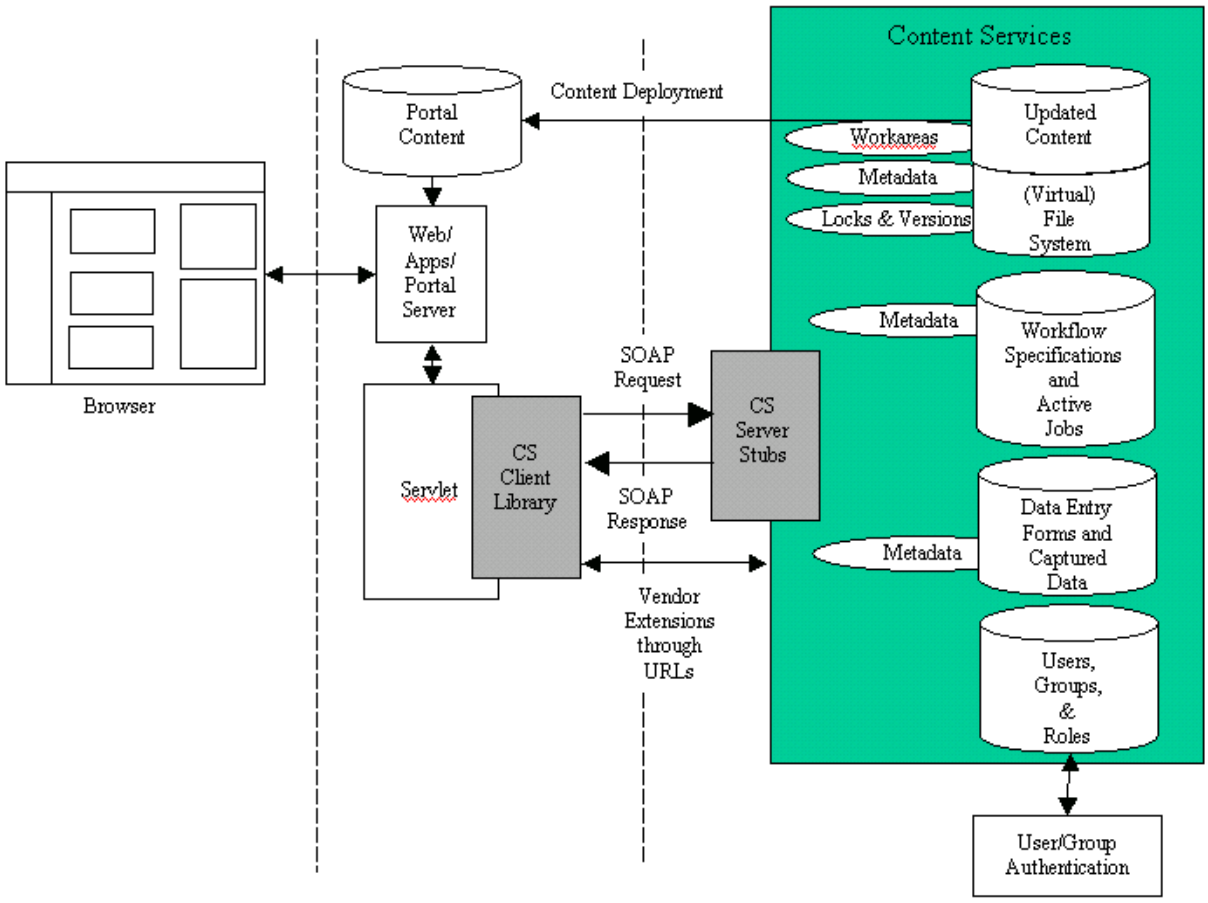
The following namespaces and namespace names are used in this specification:

Prefix	Namespace URI	Definition
wsdl	http://schemas.xmlsoap.org/wsdl/	Namespace for WSDL[WSDL] Framework.
soap	http://schemas.xmlsoap.org/wsdl/soap/	Namespace for WSDL SOAP[SOAP] binding.
xsd	http://www.w3.org/2000/10/XMLSchema	Schema namespace as defined by XSD[XSD].
ns	http://content-services.org/services1.0.xsd	Namespace for WS/CS as specified in this document.

2 Architecture

The Content Services defined in this document provide a set of basic functions needed across an organization's content infrastructure. As services, they wait for a client to make a request, which they will try to honor. They will then send a response message to the client.

The client could be a typical end-user application, however, that is not expected to be the most common situation. The client for the services will usually be a piece of middleware running under the control of a web applications server (see figure 1). That client for the service is, in turn, a server for requests coming in from end-users by way of entries in an HTML form, or from applications performing regularly scheduled tasks.



2.1 Key Terms

The content services interface defined in this document assumes an underlying implementation and content management architecture with the following logical characteristics:

Content Server Functionality

The underlying content server provides a persistent store for content files and extended file attributes, with both versioning capability and virtualization. Content can also consist of structured data records, whose data capture form definitions or templates are accessible from the content server. The content server also provides a workflow capability, providing both work assignment and an approval process, as well as the current status of active tasks pending for a particular user.

Workarea

Each workarea contains a virtual copy of a portion of the content server's file system, organized in a directory structure that can be navigated. Within a workarea, a file's content can be successively modified before being submitted. The update service replaces a workarea file with its latest version. Workareas can be provided for each user or they can be shared by groups of users.

Staging Area

Each group of related workareas has one staging area which contains the latest integrated version of each file within that portion of the content server's file system. An integrated file version reflects, successively, all past submissions of modified versions of that file from these workareas.

Modified

A modified file in a workarea is one whose content has changed but which has not yet been submitted to staging.

Overwrite

A version conflict occurs when a modified file in a workarea is out of date with respect to the integrated version of that file in staging, due to a submission of the equivalent file (with the same path and name) from a different but related workarea. A file's version conflict can be resolved either by overwriting the version in staging with the version in the workarea or by overwriting the version in the workarea with the version from staging.

Note: The content of two files in version conflict may or may not be in conflict in the sense of being able to trivially merge their content differences. Content merging, if performed, is a client side operation: A client can read the content of a modified file in a workarea, update that file by overwriting it with the latest version from staging, read the latest version of the file, perform the content merge, write the file's newly merged content, and then submit that file normally.

Lock

A lock is placed on a file in a workarea and establishes a privileged status for that particular file with respect to all the equivalent files located in different related workareas (i.e. with respect to those files with the same name and path relative to their workareas). Those files may now be viewed but can neither be edited nor submitted to staging until either the lock is released or the locked file itself is submitted to staging. A lock on a file is automatically released when that file is submitted (no client side operation is needed).

Note: Potential version conflicts can be avoided if all users always lock and then get the latest version of a file from staging, with overwrite, before opening that file for modification. Since this interface is intended to possibly support end users who do not have the knowledge or training to understand and resolve version conflicts, the editFile service allows this behavior to be specified if desired. While this approach avoids versioning conflicts and the need to possibly merge content, end users may run into lock conflicts instead, where a file is unavailable for editing due to an outstanding lock. The content management system should report to clients which user has the outstanding lock when a lock conflict is encountered. The undoChanges service allows an end user to relinquish a lock upon external request (such as an e-mail), giving up their content modifications in this process, as this service automatically gets the latest version, with overwrite, from staging before relinquishing the lock (to ensure no version conflicts will result). If a client exposes version conflicts to end users, the unlockFiles service can be used to relinquish locks without side effects.

Workflow

A workflow or job process is a specific instantiation of a general workflow job description, used, for example,

to assign work to a user or to implement a review procedure before submitting a file to staging. A workflow consists of a series of tasks. A workflow can have a number of files attached to it, allowing them to be part of an assignment or, after being changed, to be easily viewed by an approver. A workflow has a series of comments associated with it, which can be extended by each subsequent user involved with that workflow's execution, allowing them to specify, for example, what content changes were made or why a submission wasn't approved. Each file attached to a workflow has its own comment series as well, allowing detailed instructions or comments to be made on a per file basis if this is convenient. Workflow comments and workflow file comments are not stored persistently once that workflow's instantiation has finished executing.

Note: Work assignments, unless done using the createWorkflow service described below, are assumed to be instantiated outside of this interface by more advanced users using alternative interfaces (although such work assignments, once initiated, can be responded to by clients using this interface's services).

Workflow Job Description

A workflow job description describes all possible tasks, and the potential transitions between them, that could occur in workflows instantiated from it. The content server is assumed to have a number of workflow job descriptions stored in it, which a client program can request, typically to be displayed as a selection form so that an end user can select one, supply any needed parameters, and invoke that workflow. Alternatively, the createWorkflow service allows a client to supply and invoke an arbitrary workflow.

Submit

The submitDirect service takes modified workarea files and updates their integrated versions in staging with their modified content, releasing any outstanding locks on those files. Typically, however, a client will use the submit service instead, specifying a submission workflow job description to initiate an approval workflow, which in turn does the actual submit to staging as part of its final task (assuming the approvers actually approve the content modifications).

Hole

Once a file or directory has been submitted to staging, if it is deleted or otherwise hidden, it is represented by a hole.

Extended Attributes

A series of name-value pairs associated with a file and stored persistently.

3 Services

This section documents the Content Services defined by the WSDL file in Appendix B. The services are grouped into sets according to rough functionality. For example, workflow services or file management services. All services MUST provide a consistent user context according to the user's sessionToken.

3.1 Conventions

The service definitions follow a relatively small number of conventions. If a request/response pair have corresponding arrays, no guarantee is made that the order of the items in the response array will match that of the request array, unless stated otherwise. Nor is it safe to assume the array in the response will be the same length as the array in the request.

3.2 Session Management

Many operations on collections of content are most naturally handled as parts of a "session". During a session, successive operations use the state of the system established by earlier operations. Unfortunately, HTTP was designed to be a stateless protocol. Therefore we must provide operations to establish and maintain sessions.

3.2.1 startSession

A client which wishes to create a new session sends a startSession message to the service host.

Parts:

username (xsd:string)

The identifier used to authenticate the user.

password (xsd:string)

The password used to authenticate the user. The password is not encrypted. Therefore, the startSession message SHOULD be sent only over a secure connection. Clients MAY send this over an unencrypted connection, typically when debugging.

role (xsd:string)

An identifier for the role (author, editor, ...) the user wishes to assume in this session. Implementations must support at least the four roles: "Author", "Editor", "Administrator", and "Master". Implementations MAY support additional values, but interoperation cannot be guaranteed for additional roles. If a client supplies a role not supported by the server, it shall return a fault message.

locale (xsd:string)

The user's locale, provided for localization.

pathDelimiter (xsd:string)

The delimiter character to use in filesystem paths. If 'pathDelimiter' is '\', then all paths returned from the server will use '\' as the path delimiter. Otherwise, all paths returned from the server use '/' as the path delimiter.

3.2.2 startSessionResponse

Message from server to client in response to a startSession message. Contains the token identifying the session. All future messages from the client in this session requesting services must supply this token.

Parts:

sessionToken (xsd:string)

Unique token for this session on this server. The token is an arbitrary string. Server implementations are advised to limit the length of the returned token to no more than 1kB. However, client and server implementers are advised to take normal precautions about buffer overrun attacks. Note that the token is not encrypted. This means that a hostile intermediary could impersonate the user in future transactions. Therefore, the use of a secure communications channel, such as https, is recommended.

3.2.3 endSession

Message from client signaling the end of the session. The server shall free any resources used in maintaining the session. If the server receives any later messages with that sessionToken it SHOULD respond with a fault message.

Parts:

sessionToken (xsd:string)

The token used to persistently identify a session.

3.2.4 endSessionResponse

Message from server in response to client's endSession request. Server SHALL return a fault if the request failed.

Message contains no parts.

3.2.5 ping

Message from client to check that the server is running. This message, like the "ping" program, may also be used to measure network latency.

Parts:

packetIn (xsd:string)

Arbitrary data to send to server. It will be sent back to the client in the server's pingResponse message.

3.2.6 pingResponse

Message from server to client in response to a ping message. Contains the same data that was sent in the ping message from the client to the server.

Parts:

packetOut (xsd:string)

A copy of the data packetIn of the ping request.

3.3 File Management

A content management system must be able to deal with a wide variety of content. Much of the time that content is available in files, or can be made to look like files. This section documents a basic service for managing files.

3.3.1 getDirectoryContents

Message from client requesting the file server return a listing of the contents of the specified directory.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

Workarea-relative path to directory to list. The empty string ("") is used for the top-level directory.

3.3.2 getDirectoryContentsResponse

Message from file server in response to client's getDirectoryContents request. It contains an array listing the directory's contents.

Parts:

fileArray (schema:CSFileArray)

Simple files, directories, and symbolic links will be returned, but not 'holes'.

3.3.3 getFiles

Message from client requesting the file server return information about the specified files.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of filesystem paths to the files to be described. If one or more of the specified files do not exist on the server, it MUST return a fault.

3.3.4 getFilesResponse

Message from file server in response to client's getFiles request. It returns information about the specified files. If the server cannot locate one of the specified files, a fault is returned instead.

Parts:

fileArray (schema:CSFileArray)

Array of returned files. The order of the files is NOT guaranteed to match the order of the paths in the request.

3.3.5 getModifiedFiles

Message from client requesting the file server return an array of files in a directory that have been modified since the last version was submitted.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

Workarea relative path to the directory the server will examine for modified files. All subdirectories will be examined recursively.

onlyByCurrentUser (xsd:boolean)

If 'onlyByCurrentUser' is 'true', then server will list files modified only by the logged-in user. Otherwise, all modified files in the directory and its subdirectories will be listed.

3.3.6 getModifiedFilesResponse

Message from file server in response to client's getModifiedFiles request. It provides the array of modified files.

Parts:

fileArray (schema:CSFileArray)

The array of modified files.

3.3.7 readFile

Message from client requesting the file server return an array of bytes from the specified file.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

Filesystem path to the specified file, relative to the workarea.

offset (xsd:long)

The starting position in the file, in bytes, of the segment to be returned. 0 is the beginning of the file. The segment returned will, potentially, extend to the end of the file unless the numBytes part is > 0. Note that the array returned may be shorter, so the client is responsible for making sure it has actually gotten all the data it wants.

numBytes (xsd:int)

If 'numBytes' > 0, then the server will not return more than 'numBytes' bytes in the response. The server may choose to return fewer bytes, even if the file size is greater than 'offset' + 'numBytes'. If 'numBytes' is not a positive integer, then it will default to a value of 'infinity'.

3.3.8 readFileResponse

Message from file server in response to client's readFile request.

Parts:

fileSegment (schema:CSFileSegment)

The segment from the file being read.

3.3.9 writeFile

Message from client requesting the file server write the specified sequence of bytes into the specified file. The file is created if it does not already exist.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

Path to the file to update, relative to the workarea.

fileContents (xsd:base64Binary)

The bytes to be written into the file. The bytes are encoded as characters using the base64 encoding.

offset (xsd:long)

If 'offset' > 0, then the 'fileContents' will be written to the file beginning at that offset. Otherwise, the 'fileContents' will be written to the beginning of the file.

numBytes (xsd:int)

Number of bytes to be written.

isEndOfFile (xsd:boolean)

If 'isEndOfFile' is 'false', then the server will not truncate the file after the 'fileContents' has been written. Otherwise, the file will be truncated after the 'fileContents' has been written.

3.3.10 writeFileResponse

Message from server in response to client's writeFile request. Server SHALL return a fault if the request failed.

Message contains no parts.

3.3.11 deleteFiles

Message from client requesting the file server delete the specified files.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of filesystem paths to the files to be deleted.

wftSubmitCommand (xsd:string)

Basic clients should send the empty string. Advanced clients may use this to send a command to the server to determine the list of available workflow templates that can be used to submit the deleted files. The advanced client allows the user to select one of the templates (if more than one is available), fill it in, and instantiate that workflow job description as a job on the server.

3.3.12 deleteFilesResponse

Message from file server in response to client's deleteFiles request.

Parts:

webDeskSubmitHelper (schema:CSWebDeskHelper)

Helper for composing server requests.

3.3.13 getExtendedAttributes

Message from client requesting the file server return the specified 'extended attributes' (EAs) associated with the specified asset. EAs are akin to a resource fork for the asset. They are an array of textual name/value pairs associated with the asset but not part of the asset's content.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

extendedAttributeNameArray (schema:CSExtendedAttributeNameArray)

Array of EA names to search for. If the asset has an EA with a matching name, its value will be returned in the response.

3.3.14 getExtendedAttributesResponse

Message from file server in response to client's getExtendedAttributes request.

Parts:

extendedAttributeArray (schema:CSExtendedAttributeArray)

The array of results. Each element in the array is a complete Extended Attribute - both the name and the value. If the request asked for an EA that was not present on the asset, the server will silently ignore that request.

3.3.15 setExtendedAttributes

Message from client requesting the file server add (or update) the specified extended attributes on the specified asset. If the name and/or value exceed any server-specific length limits, a fault MUST be returned. Server implementations SHOULD allow at least 1kB for the name and 4kB for the value.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

extendedAttributeArray (schema:CSExtendedAttributeArray)

The array of Extended Attributes. Each EA is a name/value pair. If one of the given EAs already exists on the asset, its value will be overwritten with the specified value. Any new EAs are added to the asset, otherwise its current EAs are not modified.

3.3.16 setExtendedAttributesResponse

Message from server in response to client's setExtendedAttributes request. Server SHALL return a fault if the request failed.

Message contains no parts.

3.4 Collaboration

A content infrastructure is needed when the number of users, assets, and applications grows large. Large numbers of users will usually need to collaborate on the production, utilization, and/or distribution of content assets. This section documents a basic service for managing collaborations of multiple users and the versioning of their contributions to the assets.

3.4.1 editFile

Message from client to server requesting to edit the specified file. Various options control whether the server shall lock the file, get the working copy or the last submitted copy, etc. This message is for vendor extensions for the use of higher-capability clients which track the files obtained from the server. Simple clients are expected to use the readFile and writeFile messages to update the file in the workarea.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

Path to the file to edit, relative to the workarea.

withLock (xsd:boolean)

Locks the file on the server before returning it.

lockComment (xsd:string)

Optional comment to be associated with the lock on the file.

withGetLatest (xsd:boolean)

Obtain the most recently checked-in version of the file from the staging area, rather than the (possibly modified) copy currently in the workarea. If both 'withLock' and 'withGetLatest' are true, then the file will be locked first, then the Get Latest will be performed.

withOverwrite (xsd:boolean)

Only active if 'withGetLatest' is true. If 'withOverwrite' is false, the update shall only occur if the copy in the workarea has not been modified since its contents were last synchronized with the version in the staging area. If 'withOverwrite' is true, the workarea copy will always be updated, overwriting any changes that had been made.

wftSubmitCommand (xsd:string)

Basic clients should send the empty string. Advanced clients may use this to send a command to the server to determine the list of available workflow templates that can be used to submit the edited file. The advanced client allows the user to select one of the templates (if more than one is available), fill it in, and instantiate that workflow job description as a job on the server.

3.4.2 editFileResponse

Message from server in response to a client's editFile request.

Parts:

webDeskUtility (schema:CSWebDeskUtility)

List of Helpers so that advanced clients may have persistent access to server functionality. The Helpers are available for vendor extensions to server functionality.

3.4.3 importFiles

This message is intended for advanced clients. Simple clients are expected to send the writeFile message multiple times. This message allows advanced clients to upload multiple files from a user's local directory into a workarea.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePath (xsd:string)

This is workarea-relative path to the destination directory of the imported files. The directory MUST already exist. If it does not, the server MUST respond with a fault. The files will assume the same file name that they already have. For example, if you are importing the file "c:\a\b\c\d.doc" to the directory "x/y/z", then the file would be written to "x/y/z/d.doc". If 'localFilePathArray' contains only one file, then 'areaRelativePath' may be the name of a file, instead of a directory, in which case the file will assume the name indicated by 'areaRelativePath'.

localFilePathArray (schema:CSLocalFilePathArray)

This is an array of absolute paths to local files (e.g., "c:\documents\my document.doc") to be imported.

donePage (xsd:string)

This is a URL (absolute, or relative to the server) to be loaded when the import is complete. If 'donePage' is the empty string (""), then the default ("/iw/blank.html") will be used.

doneTarget (xsd:string)

This is the JavaScript name of the target to which 'donePage' will be targeted. If 'doneTarget' is the empty string (""), then the default ("_self") will be used.

3.4.4 importFilesResponse

Message from server in response to client's importFiles request. Server SHALL return a fault if the request failed.

Parts:

webDeskImportHelper (schema:CSWebDeskHelper)

Contains an array of Helpers allowing advanced clients to access server functionality. The Helpers are available for vendor extensions to server functionality.

3.4.5 uploadFiles

Optional request from client to server to upload files. Provided for the use of advanced clients that track files which have been downloaded from the server.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of paths to the files to be uploaded. Path is relative to the workarea in the virtual file system.

donePage (xsd:string)

This is a URL (absolute, or relative to the web server) to be loaded when the import is complete. If 'donePage' is the empty string (""), then a default MAY be used.

doneTarget (xsd:string)

This is the JavaScript name of the target to which 'donePage' will be targeted. If 'doneTarget' is the empty string (""), then the default ("_self") will be used.

3.4.6 uploadFilesResponse

Message from server in response to client's uploadFiles request. Server SHALL return a fault if the request failed.

Parts:

webDeskUploadHelper (schema:CSWebDeskHelper)

List of Helpers allowing persistent access to server functionality. The Helpers are available for vendor extensions to server functionality.

3.4.7 undoChanges

Message from client requesting the server unlock the specified file and replace it with the latest version currently in the staging area.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of paths to the files to revert, relative to the workarea.

3.4.8 undoChangesResponse

Message from server in response to a client's undoChanges request.

Parts:

pathStatusPairArray (schema:CSPathStatusPairArray)

Array of file name, status pairs indicating, for each file, whether the revert request succeeded.

3.4.9 lockFiles

Message from client requesting the server lock the specified files.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

pathCommentPairArray (schema:CSPathCommentPairArray)

Array of file name, comment pairs listing the file to lock and a comment to be associated with the lock.

3.4.10 lockFilesResponse

Message from server in response to a client's lockFiles request.

Parts:

pathStatusPairArray (schema:CSPathStatusPairArray)

Array of file name, status pairs indicating, for each file, whether the lock request succeeded.

3.4.11 unlockFiles

Message from client requesting the server unlock the specified files.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of paths to the files to unlock, relative to the workarea.

3.4.12 unlockFilesResponse

Message from server in response to a client's unlockFiles request.

Parts:

pathStatusPairArray (schema:CSPathStatusPairArray)

Array of file name, status pairs giving the status of the unlock request for each file.

3.4.13 update

Message from client to server requesting it to update a set of files in a workarea with the most recent version checked into the staging area.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

sourceAreaVPath (xsd:string)

The staging area to use. If 'sourceAreaVPath' is the empty string (""), then it will default to the staging area corresponding to the 'workareaVPath' workarea.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea to update.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of paths to the files to update, relative to the workarea.

withOverwrite (xsd:boolean)

If 'true', any files in the workarea modified more recently than the copy in the staging area will be overwritten, destroying those changes.

includeSuccessFiles (xsd:boolean)

If 'true', the results of the updates for all files are returned. If 'false', results will only be returned for files which were not updated successfully.

3.4.14 updateResponse

Message from server in response to a client's update request.

Parts:

pathStatusPairArray (schema:CSPathStatusPairArray)

Array of file name, status pairs. Depending on the 'includeSuccessFiles' part in the request, the status codes will be for all files, or only for the files that could not be updated.

3.4.15 submit

Message from client to server requesting that the specified file(s) be checked into the staging area.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of paths to the files to submit, relative to the workarea.

wftSubmitCommand (xsd:string)

Basic clients should send the empty string. Advanced clients may use this to send a command to the server to determine the list of available workflow templates that can be used to submit the files. The advanced client allows the user to select one of the templates (if more than one is available), fill it in, and instantiate that workflow job description as a job on the server.

3.4.16 submitResponse

Message from server in response to a client's submit request. Provides a Helper which the client can use for persistent access to server functionality. The Helpers are available for vendor extension.

Parts:

webDeskSubmitHelper (schema:CSWebDeskHelper)

Helper for composing server requests.

3.4.17 submitDirect

Message from client to server requesting that a file be submitted into the staging area without triggering a workflow.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

withOverwrite (xsd:boolean)

If 'true', any existing copy of the file in the staging area will be overwritten.

comment (xsd:string)

Comment to associate with the file submission.

subject (xsd:string)

Summary of the comment for the file submission.

pathCommentPairArray (schema:CSPathCommentPairArray)

Array of files and their comments for submission.

includeSuccessFiles (xsd:boolean)

If 'true', all the results of the update for each file shall be returned.

3.4.18 submitDirectResponse

Message from server in response to a client's submitDirect request. Provides the status for each requested file submission.

Parts:

pathStatusPairArray (schema:CSPathStatusPairArray)

Array of file path/status code pairs.

3.5 Templating

Much content that is created follows stereotyped formats, also known as templates. This section documents a template services that allow for the rapid creation and easy reuse of content assets.

3.5.1 getTemplateCategories

Message from client to server requesting the list of available template categories and template types.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

3.5.2 getTemplateCategoriesResponse

Message from server in response to a client's getTemplateCategories request.

Parts:

templateCategoryArray (schema:CSTemplateCategoryArray)

Array of the available Template categories.

3.5.3 getWebDeskNewDcrHelper

Utility message to help those implementing user interfaces for Content Services. After sending a getTemplateCategories message, this message can be sent for each of the TemplateTypes in the getTemplateCategoriesResponse.

Parts:

sessionToken (xsd:string)

Unique token for this session on this server.

workareaVPath (xsd:string)

Virtual filesystem path to the user's workarea in which the operation is to take place.

templateTypeId (xsd:string)

The id of the template type to get a helper for.

3.5.4 getWebDeskNewDcrHelperResponse

Message from server in response to a client's getWebDeskNewDcrHelper request.

Parts:

webDeskNewDcrHelper (schema:CSWebDeskHelper)

Helper for composing server requests.

3.6 Workflow

In large organizations, none of us work alone. We need to interact with other parts of the organization. If we are

creating pages for a corporation's external web site, there will be an approval process that must be followed. This section documents a set of workflow services that can be used to orchestrate such interactions and processes.

3.6.1 getTasks

Request from client that workflow server respond with the list of tasks assigned to the user identified by the supplied sessionToken.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

taskQuery (xsd:string)

If 'taskQuery' is empty, then 'taskArray' will contain all displayable tasks for this user. If a non-empty query is provided, it MUST validate according to the taskQuery DTD in Appendix A of this specification.

3.6.2 getTasksResponse

Message from server to client in response to a getTasks message. Contains the (possibly empty) list of tasks.

Parts:

taskArray (schema:CSTaskArray)

The array of tasks for the user and role corresponding to the sessionToken supplied in the getTasks request.

3.6.3 getTasksWithoutFiles

Request from client that workflow server respond with the list of tasks assigned to the user identified by the supplied sessionToken. All the CSTasks in the response will have an empty taskFileArray. This request is an alternative to the getTasks request. It offers higher performance by sacrificing some data not always needed.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

taskQuery (xsd:string)

If 'taskQuery' is empty, then 'taskArray' will contain all displayable tasks for this user. Otherwise, the taskQuery element must validate according to the taskQuery DTD in Appendix A of this specification.

3.6.4 getTasksWithoutFilesResponse

Message from workflow server in response to a client's getTasksWithoutFiles request.

Parts:

taskArray (schema:CSTaskArray)

The array of returned tasks.

3.6.5 addTaskComment

Request from client that workflow server add a comment to the array of comments it maintains for a particular task.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

taskId (xsd:long)

Identifier for the task to update.

comment (xsd:string)

The comment to add to the taskCommentArray.

3.6.6 addTaskCommentResponse

Message from server in response to client's addTaskComment request. Server SHALL return a fault if the request failed.

Message contains no parts.

3.6.7 addTaskFileComment

Request from client that the workflow server add a comment to a file, as associated with a particular task. A file may be associated with multiple tasks. Within a task, each file has a distinct array of comments separate from and in addition to the array of comments associated with the task. This message adds a new comment to that array.

Parts:

sessionToken (xsd:string)

The token identifying the current session.

taskId (xsd:long)

ID for the task to update.

areaRelativePath (xsd:string)

comment (xsd:string)

The comment to add to the task.

3.6.8 addTaskFileCommentResponse

Message from server in response to client's addTaskFileComment request. Server SHALL return a fault if the request failed.

Message contains no parts.

3.6.9 addTaskFiles

Associates a list of files with a task, typically for the purpose of requesting their review or editing.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

taskId (xsd:long)

ID of the task to update.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of paths to the files to be associated with the task. The paths are relative to the current workarea.

3.6.10 addTaskFilesResponse

Message from workflow server in response to client's addTaskFiles request.

Parts:

pathStatusPairArray (schema:CSPathStatusPairArray)

Array of file path/status code pairs. The order of the indications is not specified.

3.6.11 removeTaskFiles

Message from client requesting workflow server remove the specified files from the task.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

taskId (xsd:long)

ID of the task to update.

areaRelativePathArray (schema:CSAreaRelativePathArray)

Array of filesystem paths to the files to be removed. The paths must correspond to files known to the workflow server.

3.6.12 removeTaskFilesResponse

Message from workflow server in response to client's removeTaskFiles request.

Parts:

pathStatusPairArray (schema:CSPathStatusPairArray)

Array of filename, status pairs indicating the status of the remove request for each file. The order of the files in the array is not specified.

3.6.13 finishTask

Message from client requesting the workflow server mark a task as completed and remove it from the user's list of active tasks.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

taskId (xsd:long)

ID for the task to transition.

taskTransition (xsd:string)

String defining which transition out of task was taken. Possible values are "Done", "Approve", and "Reject". The string MUST match one of the values in the taskTransitionArray of the task.

comment (xsd:string)

Human-readable comment about the task's completion.

3.6.14 finishTaskResponse

Message from server in response to client's finishTask request. Server SHALL return a fault if the request failed.

Message contains no parts.

3.6.15 createWorkflow

Message from client requesting the workflow server register the given workflow specification. The workflow can then be used by messages which initiate workflows. such as submit.

Parts:

sessionToken (xsd:string)

The token used to identify the current session.

workflowSpecification (xsd:string)

If 'workflowSpecification' is a valid workflow specification XML document, then the workflow will be created.

3.6.16 createWorkflowResponse

Message from workflow server in response to client's createWorkflow request.

Parts:

workflowId (xsd:long)

If a job is successfully instantiated, a workflow ID will be returned. The client shall use that ID to refer to the job on the server. Once a job completes, it is removed from the server. Any further use of the workflowID is an error.

3.7 Utility

This section defines messages provided as utilities.

3.7.1 getVersionString

Message from client to retrieve a string identifying the version of the server software. Note that this is NOT a string identifying what version of the Content Services interface is being used. That information is available in the namespace declaration.

Message contains no parts.

3.7.2 getVersionStringResponse

Message from server to client in response to a getVersionString message. Contains the version string of the server.

Parts:

versionString (xsd:string)

A character string identifying the version of the server software. It is expected to typically contain information such as the name of the organization which developed the server software, and any version information they wish to provide. The precise content of this string is unspecified.

4 Bibliography

RFC-2119

S. Bradner, "Key Words for use in RFCs to Indicate Requirements Levels", RFC 2119, Internet Engineering Task Force, March 1997.

SOAP

Don Box, et. al., "Simple Object Access Protocol (SOAP) 1.1", W3C Note 08 May 2000, May 2000.

WSDL

Erik Christensen, et. al., "Web Services Description Language (WSDL) 1.1", W3C Note 15 March 2001, March 2001.

XSD

Paul V. Biron and Ashok Malhotra, "XML Schema Part 2: Datatypes", W3C Recommendation 02 May 2001, May 2001.

A Task Query Grammar

A basic grammar for querying workflow tasks is provided for use in messages like <getTasks>. The normative definition of the query grammar is:

```
<!ELEMENT taskquery (and|or|not|ownedby|active|wfactive|wfownedby|
                     workflow|needsattention|type|sharedby|undoableby)>
    <!ELEMENT and ((and|or|not|ownedby|active|wfactive|wfownedby|
                    workflow|needsattention|type|sharedby|undoableby)+)>
    <!ELEMENT or ((and|or|not|ownedby|active|wfactive|wfownedby|
                   workflow|needsattention|type|sharedby|undoableby)+)>
    <!ELEMENT not (and|or|not|ownedby|active|wfactive|wfownedby|
                    workflow|needsattention|type|sharedby|undoableby)>
    <!-- A taskquery is a simple query, or a Boolean combination of
         queries. All simple queries evaluate to 'true' or 'false'.
    -->

    <!ELEMENT ownedby EMPTY>
        <!ATTLIST ownedby v CDATA #REQUIRED>
        <!-- Evaluates to true if the task is owned by the specified user -->

    <!ELEMENT active EMPTY>
        <!-- Evaluates to true if the task is currently 'active'. That
        is, if it has been assigned and not yet completed. -->

    <!ELEMENT wfactive EMPTY>
        <!-- Evaluates to true if the [...] -->

    <!ELEMENT wfownedby EMPTY>
        <!ATTLIST wfownedby v CDATA #REQUIRED>
        <!-- Evaluates to true if the workflow is owned by the
             specified user. -->

    <!ELEMENT workflow EMPTY>
        <!ATTLIST workflow v CDATA #REQUIRED>
        <!-- Evaluates to true if the [...] -->

    <!ELEMENT needsattention EMPTY>
        <!-- Evaluates to true if the [...] -->

    <!ELEMENT type EMPTY>
        <!ATTLIST type v (usertask, grouptask, externaltask,
                          cgitask, submittask, updatetask, endtask,
                          dummytask, locktask, wftask) #REQUIRED>
        <!-- Evaluates to true if the task is of the specified type. -->

    <!ELEMENT sharedby EMPTY>
        <!ATTLIST sharedby v CDATA #REQUIRED>
        <!-- Evaluates to true if the [...] -->

    <!ELEMENT undoableby EMPTY>
```

```
        <!ATTLIST undoableby v CDATA #REQUIRED>
        <!-- Evaluates to true if the [...] -->
```

B WSDL Definition

The following WSDL document is the normative definition of the content services specified by this document.

```
<definitions         xmlns="http://schemas.xmlsoap.org/wsdl/"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns:schema="http://content-services.org/services1.0.xsd"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:ns="http://content-services.org/services1.0.wsdl"
 targetNamespace="http://content-services.org/services1.0.wsdl">

  <!-- ============================================================
                              types
       ============================================================ -->


   <types>

      <schema         xmlns="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://content-services.org/services1.0.xsd">

  <!-- ============================================================
                       complex type definitions
       ============================================================ -->


         <complexType name="CSPathCommentPair">
            <sequence>
               <element name="areaRelativePath" type="xsd:string"/>
               <element name="comment" type="xsd:string"/>
            </sequence>
         </complexType>


         <complexType name="CSPathStatusPair">
            <sequence>
               <element name="areaRelativePath" type="xsd:string"/>
               <element name="statusCode" type="xsd:int"/>
               <element name="statusMessage" type="xsd:string"/>
            </sequence>
         </complexType>


         <complexType name="CSFile">
            <sequence>
               <element name="areaRelativePath" type="xsd:string"/>
               <element name="lastModDate" type="xsd:dateTime"/>
               <element name="lastModUser" type="xsd:string"/>
               <element name="kind" type="xsd:string"/>
               <element name="size" type="xsd:long"/>
               <element name="owner" type="xsd:string"/>
               <element name="isLocked" type="xsd:boolean"/>
               <element name="isModified" type="xsd:boolean"/>
               <element name="isReadable" type="xsd:boolean"/>
               <element name="isWritable" type="xsd:boolean"/>
               <element name="relatedTaskArray" type="schema:CSTaskIdArray"/>
               <element name="lockAreaVPath" type="xsd:string"/>
               <element name="lockOwner" type="xsd:string"/>
```

```
      <element name="lockCreationTime" type="xsd:dateTime"/>
      <element name="lockComment" type="xsd:string"/>
      <element name="webDeskContentType" type="xsd:string"/>
      <element name="webDeskViewHelper" type="schema:CSWebDeskHelper"/>
      <element name="webDeskSceHelper" type="schema:CSWebDeskHelper"/>
      <element name="webDeskFileHelper" type="schema:CSWebDeskHelper"/>
      <element name="webDeskPreviewHelper" type="schema:CSWebDeskHelper"/>
   </sequence>
</complexType>


<complexType name="CSTask">
   <sequence>
      <element name="taskId" type="xsd:long"/>
      <element name="description" type="xsd:string"/>
      <element name="owner" type="xsd:string"/>
      <element name="isActive" type="xsd:boolean"/>
      <element name="activationTime" type="xsd:dateTime"/>
      <element name="expirationTime" type="xsd:dateTime"/>
      <element name="isUndoable" type="xsd:boolean"/>
      <element name="isReadOnly" type="xsd:boolean"/>
      <element name="activeSubWorkflowId" type="xsd:long"/>
      <element name="templateFilePath" type="xsd:string"/>
      <element name="areaVPath" type="xsd:string"/>
      <element name="kind" type="xsd:string"/>
      <element name="name" type="xsd:string"/>
      <element name="cgiData" type="xsd:string"/>
      <element name="workflowOwner" type="xsd:string"/>
      <element name="dueDate" type="xsd:dateTime"/>
      <element name="priority" type="xsd:string"/>
      <element name="taskTransitionArray" type="schema:CSTaskTransitionArray"/>
      <element name="taskFileArray" type="schema:CSTaskFileArray"/>
      <element name="taskCommentArray" type="schema:CSCommentArray"/>
   </sequence>
</complexType>


<complexType name="CSTemplateCategory">
   <sequence>
      <element name="name" type="xsd:string"/>
      <element name="templateTypeArray" type="schema:CSTemplateTypeArray"/>
   </sequence>
</complexType>


<complexType name="CSTemplateType">
   <sequence>
      <element name="name" type="xsd:string"/>
      <element name="id" type="xsd:string"/>
   </sequence>
</complexType>


<complexType name="CSTaskFile">
   <sequence>
      <element name="areaRelativePath" type="xsd:string"/>
      <element name="commentArray" type="schema:CSCommentArray"/>
   </sequence>
</complexType>


<complexType name="CSNameValuePair">
   <sequence>
      <element name="name" type="xsd:string"/>
      <element name="value" type="xsd:string"/>
```

```
      </sequence>
    </complexType>


    <complexType name="CSWebDeskHelper">
      <sequence>
        <element name="action" type="xsd:anyURI"/>
        <element name="data" type="schema:CSNameValuePairArray"/>
      </sequence>
    </complexType>


    <complexType name="CSWebDeskUtility">
      <sequence>
        <element name="webDeskContentType" type="xsd:string"/>
        <element name="webDeskEditHelper" type="schema:CSWebDeskHelper"/>
        <element name="webDeskSubmitHelper" type="schema:CSWebDeskHelper"/>
        <element name="webDeskViewHelper" type="schema:CSWebDeskHelper"/>
        <element name="webDeskSceHelper" type="schema:CSWebDeskHelper"/>
        <element name="webDeskFileHelper" type="schema:CSWebDeskHelper"/>
        <element name="webDeskPreviewHelper" type="schema:CSWebDeskHelper"/>
      </sequence>
    </complexType>


    <complexType name="CSFileSegment">
      <sequence>
        <element name="fileContents" type="xsd:base64Binary"/>
        <element name="isEndOfFile" type="xsd:boolean"/>
        <element name="fileSize" type="xsd:long"/>
      </sequence>
    </complexType>


    <complexType name="CSExtendedAttribute">
      <sequence>
        <element name="name" type="xsd:string"/>
        <element name="value" type="xsd:string"/>
      </sequence>
    </complexType>

<!-- ========================================================
                      array definitions
     ======================================================== -->


    <complexType name="CSPathCommentPairArray" base="soap:Array">
      <sequence>
        <element name="pathCommentPair" type="schema:CSPathCommentPair"
                 maxOccurs="unbounded"/>
      </sequence>
    </complexType>


    <complexType name="CSPathStatusPairArray" base="soap:Array">
      <sequence>
        <element name="pathStatusPair" type="schema:CSPathStatusPair"
                 maxOccurs="unbounded"/>
      </sequence>
    </complexType>


    <complexType name="CSAreaRelativePathArray" base="soap:Array">
      <sequence>
        <element name="areaRelativePath" type="xsd:string"
```

```
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSLocalFilePathArray" base="soap:Array">
   <sequence>
      <element name="localFilePath" type="xsd:string"
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSFileArray" base="soap:Array">
   <sequence>
      <element name="file" type="schema:CSFile"
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSTaskIdArray" base="soap:Array">
   <sequence>
      <element name="taskId" type="xsd:long"
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSTaskArray" base="soap:Array">
   <sequence>
      <element name="task" type="schema:CSTask"
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSTemplateCategoryArray" base="soap:Array">
   <sequence>
      <element name="templateCategory" type="schema:CSTemplateCategory"
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSTemplateTypeArray" base="soap:Array">
   <sequence>
      <element name="templateType" type="schema:CSTemplateType"
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSTaskFileArray" base="soap:Array">
   <sequence>
      <element name="taskFile" type="schema:CSTaskFile"
                maxOccurs="unbounded"/>
   </sequence>
</complexType>


<complexType name="CSNameValuePairArray" base="soap:Array">
   <sequence>
      <element name="nameValuePair" type="schema:CSNameValuePair"
                maxOccurs="unbounded"/>
```

```
         </sequence>
      </complexType>


      <complexType name="CSCommentArray" base="soap:Array">
         <sequence>
            <element name="comment" type="xsd:string"
                     maxOccurs="unbounded"/>
         </sequence>
      </complexType>


      <complexType name="CSTaskTransitionArray" base="soap:Array">
         <sequence>
            <element name="taskTransition" type="xsd:string"
                     maxOccurs="unbounded"/>
         </sequence>
      </complexType>


      <complexType name="CSExtendedAttributeNameArray" base="soap:Array">
         <sequence>
            <element name="extendedAttributeName" type="xsd:string"
                     maxOccurs="unbounded"/>
         </sequence>
      </complexType>


      <complexType name="CSExtendedAttributeArray" base="soap:Array">
         <sequence>
            <element name="extendedAttribute" type="schema:CSExtendedAttribute"
                     maxOccurs="unbounded"/>
         </sequence>
      </complexType>

   </schema>

 </types>

<!-- ============================================================
                           faults
     ============================================================ -->


 <message name="fault">
    <part/>
 </message>


<!-- ============================================================
                  session management messages
     ============================================================ -->


 <message name="startSession">
    <part name="username" type="xsd:string"/>
    <part name="password" type="xsd:string"/>
    <part name="role" type="xsd:string"/>
    <part name="locale" type="xsd:string"/>
    <part name="pathDelimiter" type="xsd:string"/>
 </message>

 <message name="startSessionResponse">
    <part name="sessionToken" type="xsd:string"/>
 </message>
```

```
 <message name="endSession">
    <part name="sessionToken" type="xsd:string"/>
 </message>

 <message name="endSessionResponse"/>


 <message name="ping">
    <part name="packetIn" type="xsd:string"/>
 </message>

 <message name="pingResponse">
    <part name="packetOut" type="xsd:string"/>
 </message>


 <message name="getVersionString"/>

 <message name="getVersionStringResponse">
    <part name="versionString" type="xsd:string"/>
 </message>


<!-- ============================================================
                          workflow messages
   ============================================================ -->


 <message name="getTasks">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskQuery" type="xsd:string"/>
 </message>

 <message name="getTasksResponse">
    <part name="taskArray" type="schema:CSTaskArray"/>
 </message>


 <message name="getTasksWithoutFiles">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskQuery" type="xsd:string"/>
 </message>

 <message name="getTasksWithoutFilesResponse">
    <part name="taskArray" type="schema:CSTaskArray"/>
 </message>


 <message name="getTasksById">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskIdArray" type="schema:CSTaskIdArray"/>
 </message>

 <message name="getTasksByIdResponse">
    <part name="taskArray" type="schema:CSTaskArray"/>
 </message>


 <message name="addTaskComment">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskId" type="xsd:long"/>
    <part name="comment" type="xsd:string"/>
 </message>
```

```
 <message name="addTaskCommentResponse"/>


 <message name="addTaskFileComment">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskId" type="xsd:long"/>
    <part name="areaRelativePath" type="xsd:string">
    Path to the file to comment on, relative to the user's workarea.
  </part>
    <part name="comment" type="xsd:string"/>
 </message>

 <message name="addTaskFileCommentResponse"/>


 <message name="addTaskFiles">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskId" type="xsd:long"/>
    <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
 </message>

 <message name="addTaskFilesResponse">
    <part name="pathStatusPairArray" type="schema:CSPathStatusPairArray"/>
 </message>


 <message name="removeTaskFiles">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskId" type="xsd:long"/>
    <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
 </message>

 <message name="removeTaskFilesResponse">
    <part name="pathStatusPairArray" type="schema:CSPathStatusPairArray"/>
 </message>


 <message name="finishTask">
    <part name="sessionToken" type="xsd:string"/>
    <part name="taskId" type="xsd:long"/>
    <part name="taskTransition" type="xsd:string"/>
    <part name="comment" type="xsd:string"/>
 </message>

 <message name="finishTaskResponse"/>


 <message name="createWorkflow">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workflowSpecification" type="xsd:string"/>
 </message>

 <message name="createWorkflowResponse">
    <part name="workflowId" type="xsd:long"/>
 </message>

<!-- ============================================================
                     file management messages
     ============================================================ -->


 <message name="getDirectoryContents">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
```

```
   <part name="areaRelativePath" type="xsd:string"/>
</message>

<message name="getDirectoryContentsResponse">
   <part name="fileArray" type="schema:CSFileArray"/>
</message>


<message name="getFiles">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
</message>

<message name="getFilesResponse">
   <part name="fileArray" type="schema:CSFileArray"/>
</message>


<message name="getModifiedFiles">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePath" type="xsd:string"/>
   <part name="onlyByCurrentUser" type="xsd:boolean"/>
</message>

<message name="getModifiedFilesResponse">
   <part name="fileArray" type="schema:CSFileArray"/>
</message>


<message name="readFile">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePath" type="xsd:string"/>
   <part name="offset" type="xsd:long"/>
   <part name="numBytes" type="xsd:int"/>
</message>

<message name="readFileResponse">
   <part name="fileSegment" type="schema:CSFileSegment"/>
</message>


<message name="writeFile">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePath" type="xsd:string"/>
   <part name="fileContents" type="xsd:base64Binary"/>
   <part name="offset" type="xsd:long"/>
   <part name="numBytes" type="xsd:int"/>
   <part name="isEndOfFile" type="xsd:boolean"/>
</message>

<message name="writeFileResponse"/>


<message name="deleteFiles">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
   <part name="wftSubmitCommand" type="xsd:string"/>
</message>

<message name="deleteFilesResponse">
```

```
    <part name="webDeskSubmitHelper" type="schema:CSWebDeskHelper"/>
 </message>


 <message name="getExtendedAttributes">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
    <part name="areaRelativePath" type="xsd:string">
    Path to the asset whose extended attributes are to be fetched.
    Path is relative to the workarea.
  </part>
    <part name="extendedAttributeNameArray" type="schema:CSExtendedAttributeNameArray"/>
 </message>

 <message name="getExtendedAttributesResponse">
    <part name="extendedAttributeArray" type="schema:CSExtendedAttributeArray"/>
 </message>


 <message name="setExtendedAttributes">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
    <part name="areaRelativePath" type="xsd:string">
    Path to the asset to update. Path is relative to the workarea.
  </part>
    <part name="extendedAttributeArray" type="schema:CSExtendedAttributeArray"/>
 </message>

 <message name="setExtendedAttributesResponse"/>


<!-- ============================================================
                   collaboration messages
     ============================================================ -->


 <message name="editFile">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
    <part name="areaRelativePath" type="xsd:string"/>
    <part name="withLock" type="xsd:boolean"/>
    <part name="lockComment" type="xsd:string"/>
    <part name="withGetLatest" type="xsd:boolean"/>
    <part name="withOverwrite" type="xsd:boolean"/>
    <part name="wftSubmitCommand" type="xsd:string"/>
 </message>

 <message name="editFileResponse">
    <part name="webDeskUtility" type="schema:CSWebDeskUtility"/>
 </message>


 <message name="importFiles">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
    <part name="areaRelativePath" type="xsd:string"/>
    <part name="localFilePathArray" type="schema:CSLocalFilePathArray"/>
    <part name="donePage" type="xsd:string"/>
    <part name="doneTarget" type="xsd:string"/>
 </message>

 <message name="importFilesResponse">
    <part name="webDeskImportHelper" type="schema:CSWebDeskHelper"/>
 </message>
```

```
<message name="uploadFiles">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
   <part name="donePage" type="xsd:string"/>
   <part name="doneTarget" type="xsd:string"/>
</message>

<message name="uploadFilesResponse">
   <part name="webDeskUploadHelper" type="schema:CSWebDeskHelper"/>
</message>


<message name="undoChanges">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
</message>

<message name="undoChangesResponse">
   <part name="pathStatusPairArray" type="schema:CSPathStatusPairArray"/>
</message>


<message name="lockFiles">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="pathCommentPairArray" type="schema:CSPathCommentPairArray"/>
</message>

<message name="lockFilesResponse">
   <part name="pathStatusPairArray" type="schema:CSPathStatusPairArray"/>
</message>


<message name="unlockFiles">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
</message>

<message name="unlockFilesResponse">
   <part name="pathStatusPairArray" type="schema:CSPathStatusPairArray"/>
</message>


<message name="update">
   <part name="sessionToken" type="xsd:string"/>
   <part name="sourceAreaVPath" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
   <part name="withOverwrite" type="xsd:boolean"/>
   <part name="includeSuccessFiles" type="xsd:boolean"/>
</message>

<message name="updateResponse">
   <part name="pathStatusPairArray" type="schema:CSPathStatusPairArray"/>
</message>


<message name="submit">
   <part name="sessionToken" type="xsd:string"/>
   <part name="workareaVPath" type="xsd:string"/>
   <part name="areaRelativePathArray" type="schema:CSAreaRelativePathArray"/>
```

```
    <part name="wftSubmitCommand" type="xsd:string"/>
 </message>

 <message name="submitResponse">
    <part name="webDeskSubmitHelper" type="schema:CSWebDeskHelper"/>
 </message>


 <message name="submitDirect">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
    <part name="withOverwrite" type="xsd:boolean"/>
    <part name="comment" type="xsd:string"/>
    <part name="subject" type="xsd:string"/>
    <part name="pathCommentPairArray" type="schema:CSPathCommentPairArray"/>
    <part name="includeSuccessFiles" type="xsd:boolean"/>
 </message>

 <message name="submitDirectResponse">
    <part name="pathStatusPairArray" type="schema:CSPathStatusPairArray"/>
 </message>


<!-- ============================================================
                       templating messages
     ============================================================ -->


 <message name="getTemplateCategories">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
 </message>

 <message name="getTemplateCategoriesResponse">
    <part name="templateCategoryArray" type="schema:CSTemplateCategoryArray"/>
 </message>


 <message name="getWebDeskNewDcrHelper">
    <part name="sessionToken" type="xsd:string"/>
    <part name="workareaVPath" type="xsd:string"/>
    <part name="templateTypeId" type="xsd:string"/>
 </message>

 <message name="getWebDeskNewDcrHelperResponse">
    <part name="webDeskNewDcrHelper" type="schema:CSWebDeskHelper"/>
 </message>


<!-- ============================================================
                          port types
     ============================================================ -->


 <portType name="ContentServicesPortType">

    <operation name="startSession">
       <input message="ns:startSession"/>
       <output message="ns:startSessionResponse"/>
       <fault message="ns:fault"/>
    </operation>

    <operation name="endSession">
       <input message="ns:endSession"/>
       <output message="ns:endSessionResponse"/>
```

```
   <fault message="ns:fault"/>
</operation>

<operation name="ping">
   <input message="ns:ping"/>
   <output message="ns:pingResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getVersionString">
   <input message="ns:getVersionString"/>
   <output message="ns:getVersionStringResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getTasks">
   <input message="ns:getTasks"/>
   <output message="ns:getTasksResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getTasksWithoutFiles">
   <input message="ns:getTasksWithoutFiles"/>
   <output message="ns:getTasksWithoutFilesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getTasksById">
   <input message="ns:getTasksById"/>
   <output message="ns:getTasksByIdResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="addTaskComment">
   <input message="ns:addTaskComment"/>
   <output message="ns:addTaskCommentResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="addTaskFileComment">
   <input message="ns:addTaskFileComment"/>
   <output message="ns:addTaskFileCommentResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="addTaskFiles">
   <input message="ns:addTaskFiles"/>
   <output message="ns:addTaskFilesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="removeTaskFiles">
   <input message="ns:removeTaskFiles"/>
   <output message="ns:removeTaskFilesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="finishTask">
   <input message="ns:finishTask"/>
   <output message="ns:finishTaskResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="createWorkflow">
   <input message="ns:createWorkflow"/>
```

```
   <output message="ns:createWorkflowResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getDirectoryContents">
   <input message="ns:getDirectoryContents"/>
   <output message="ns:getDirectoryContentsResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getFiles">
   <input message="ns:getFiles"/>
   <output message="ns:getFilesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getModifiedFiles">
   <input message="ns:getModifiedFiles"/>
   <output message="ns:getModifiedFilesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="readFile">
   <input message="ns:readFile"/>
   <output message="ns:readFileResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="writeFile">
   <input message="ns:writeFile"/>
   <output message="ns:writeFileResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="deleteFiles">
   <input message="ns:deleteFiles"/>
   <output message="ns:deleteFilesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="getExtendedAttributes">
   <input message="ns:getExtendedAttributes"/>
   <output message="ns:getExtendedAttributesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="setExtendedAttributes">
   <input message="ns:setExtendedAttributes"/>
   <output message="ns:setExtendedAttributesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="editFile">
   <input message="ns:editFile"/>
   <output message="ns:editFileResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="importFiles">
   <input message="ns:importFiles"/>
   <output message="ns:importFilesResponse"/>
   <fault message="ns:fault"/>
</operation>

<operation name="uploadFiles">
```

```
      <input message="ns:uploadFiles"/>
      <output message="ns:uploadFilesResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="undoChanges">
      <input message="ns:undoChanges"/>
      <output message="ns:undoChangesResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="lockFiles">
      <input message="ns:lockFiles"/>
      <output message="ns:lockFilesResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="unlockFiles">
      <input message="ns:unlockFiles"/>
      <output message="ns:unlockFilesResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="update">
      <input message="ns:update"/>
      <output message="ns:updateResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="submit">
      <input message="ns:submit"/>
      <output message="ns:submitResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="submitDirect">
      <input message="ns:submitDirect"/>
      <output message="ns:submitDirectResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="getTemplateCategories">
      <input message="ns:getTemplateCategories"/>
      <output message="ns:getTemplateCategoriesResponse"/>
      <fault message="ns:fault"/>
   </operation>

   <operation name="getWebDeskNewDcrHelper">
      <input message="ns:getWebDeskNewDcrHelper"/>
      <output message="ns:getWebDeskNewDcrHelperResponse"/>
      <fault message="ns:fault"/>
   </operation>

 </portType>


<!-- ========================================================
                          bindings
     ======================================================== -->


 <binding name="ContentServicesBinding" type="ns:ContentServicesPortType">

    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
<operation name="startSession">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="endSession">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="ping">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="getVersionString">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>
```

```
<operation name="getTasks">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="getTasksWithoutFiles">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="getTasksById">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="addTaskComment">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="addTaskFileComment">
```

```
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="addTaskFiles">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="removeTaskFiles">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="finishTask">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="createWorkflow">
      <soap:operation soapAction="" style="rpc"/>
```

```
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="getDirectoryContents">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="getFiles">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="getModifiedFiles">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="readFile">
   <soap:operation soapAction="" style="rpc"/>
   <input>
```

```
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="writeFile">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="deleteFiles">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="getExtendedAttributes">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="setExtendedAttributes">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="editFile">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="importFiles">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="uploadFiles">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="undoChanges">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
```

```
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="lockFiles">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="unlockFiles">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="update">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
   <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </output>
   <fault>
      <soap:fault/>
   </fault>
</operation>


<operation name="submit">
   <soap:operation soapAction="" style="rpc"/>
   <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
   </input>
```

```
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="submitDirect">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="getTemplateCategories">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>


   <operation name="getWebDeskNewDcrHelper">
      <soap:operation soapAction="" style="rpc"/>
      <input>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </input>
      <output>
         <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             use="encoded" namespace="http://content-services.org/services1.0.xsd"/>
      </output>
      <fault>
         <soap:fault/>
      </fault>
   </operation>

 </binding>


<!-- ============================================================
                           services
     ============================================================ -->
```

```
   <service name="ContentService">
      <port name="ContentServicesPort" binding="ns:ContentServicesBinding">
         <soap:address location="http://localhost:80/iw/services/1.0"/>
      </port>
   </service>


</definitions>
```

C Workflow Grammar

The createWorkflow message sends a workflow to the server for initiation. The workflow must follow the grammar in the following DTD.

```
<!ELEMENT workflow (description?, variables?, (usertask|submittask
                    |updatetask|externaltask|cgitask|endtask|grouptask
                    |dummytask|locktask|wftask)+)>
    <!ATTLIST workflow name ID #REQUIRED
                       owner CDATA #REQUIRED
                       creator CDATA #REQUIRED
                       description CDATA #IMPLIED>

<!ELEMENT description (#PCDATA)>

<!ELEMENT usertask (description?, areavpath, successors, timeout?,
                    files?, activation?, inactivate?, resets?, eastartop*,
                    eafinishop*, variables?)>
    <!ATTLIST usertask owner CDATA #REQUIRED
                       name ID #REQUIRED
                       start (t|f) "f"
                       description CDATA #IMPLIED
                       lock (t|f) "f"
                       readonly (t|f) "f">

<!ELEMENT grouptask (description?, areavpath, successors, sharedby, timeout?,
                     files?, activation?, inactivate?, resets?, eastartop*,
                     eafinishop*, variables?)>
    <!ATTLIST grouptask name ID #REQUIRED
                        start (t|f) "f"
                        description CDATA #IMPLIED
                        lock (t|f) "f"
                        retainowner (t|f) "f"
                        readonly (t|f) "f">
    <!ELEMENT sharedby (user|group)+>
        <!ELEMENT user EMPTY>
            <!ATTLIST user v CDATA #REQUIRED>
        <!ELEMENT group EMPTY>
            <!ATTLIST group v CDATA #REQUIRED>


<!ELEMENT submittask (description?, areavpath, successorset, timeout?,
                  files?, activation?, inactivate?, resets?,
                  eastartop*, eafinishop*, variables?)>
    <!ATTLIST submittask owner CDATA #REQUIRED
                         name ID #REQUIRED
                         start (t|f) "f"
                         skipconflicts (t|f) "f"
                         skiplocked (t|f) "f"
                         override (t|f) "f"
                         unlock (t|f) "f"
                         savecomments (t|f) "f"
                         description CDATA #IMPLIED>
```

```
<!ELEMENT updatetask (description?, areavpath, successorset, srcareavpath,
                      timeout?, files?, activation?, inactivate?, resets?,
                      eastartop*, eafinishop*, variables?)>
    <!ATTLIST updatetask owner CDATA #REQUIRED
                         name ID #REQUIRED
                         start (t|f) "f"
                         delete (t|f) "t"
                         overwritemod (t|f) "f"
                         description CDATA #IMPLIED
                         lock (t|f) "f">

<!ELEMENT externaltask (description?, areavpath, successors, command,
                        timeout?, files?, activation?, inactivate?, resets?,
                        eastartop*, eafinishop*, variables?)>
    <!ATTLIST externaltask owner CDATA #REQUIRED
                           name ID #REQUIRED
                           start (t|f) "f"
                           description CDATA #IMPLIED
                           lock (t|f) "f"
                           readonly (t|f) "f"
                           retry (t|f) "t">

<!ELEMENT cgitask (description?, areavpath, successors, command,
                   timeout?, files?, activation?, inactivate?, resets?,
                   eastartop*, eafinishop*, variables?)>
    <!ATTLIST cgitask owner CDATA #REQUIRED
                      name ID #REQUIRED
                      start (t|f) "f"
                      description CDATA #IMPLIED
                      lock (t|f) "f"
                      immediate (t|f) "f"
                      readonly (t|f) "f">

<!ELEMENT wftask (description?, areavpath, successors, jobfile|wftfile,
                  timeout?, files?, activation?, inactivate?, resets?,
                  eastartop*, eafinishop*, variables?)>
    <!ATTLIST wftask owner CDATA #REQUIRED
                     name ID #REQUIRED
                     start (t|f) "f"
                     description CDATA #IMPLIED>

<!ELEMENT endtask (activation?, eastartop*, eafinishop*)>
    <!ATTLIST endtask name ID #REQUIRED
                      description CDATA #IMPLIED>

<!ELEMENT dummytask (description?, timeout, files?,
                     activation?, inactivate?, resets?,
                     eastartop*, eafinishop*, variables?)>
    <!ATTLIST dummytask name ID #REQUIRED
                        start (t|f) "f"
                        description CDATA #IMPLIED>

<!ELEMENT locktask (description?, areavpath, success, failure,
                    files?, activation?, inactivate?,
                    resets?, eastartop*, eafinishop*, variables?)>
    <!ATTLIST locktask owner CDATA #REQUIRED
                       name ID #REQUIRED
                       start (t|f) "f"
                       description CDATA #IMPLIED>
    <!-- Locking is implied -->
    <!ELEMENT success (succ+)>
    <!ELEMENT failure (succ+)>

<!ELEMENT variables (variable+)>
    <!ELEMENT variable EMPTY>
```

```
        <!ATTLIST variable key NMTOKEN #REQUIRED
                           value CDATA #REQUIRED>


<!ELEMENT command EMPTY>
    <!ATTLIST command v CDATA #REQUIRED>

<!ELEMENT jobfile EMPTY>
    <!ATTLIST jobfile v CDATA #REQUIRED>

<!ELEMENT wftfile EMPTY>
    <!ATTLIST wftfile v CDATA #REQUIRED>

<!ELEMENT areavpath EMPTY>
    <!ATTLIST areavpath v CDATA #REQUIRED>

<!ELEMENT successors (successorset+)>

<!ELEMENT successorset (description?, succ+)>
    <!ATTLIST successorset description CDATA #IMPLIED>

<!ELEMENT succ EMPTY>
    <!ATTLIST succ v IDREF #REQUIRED>

<!ELEMENT files (file+)>

<!ELEMENT file EMPTY>
    <!ATTLIST file path CDATA #REQUIRED
                   comment CDATA #REQUIRED>

<!ELEMENT activation (and|or|not|pred)>

<!ELEMENT and (and|or|not|pred)*>

<!ELEMENT or (and|or|not|pred)*>

<!ELEMENT not (and|or|not|pred)>

<!ELEMENT pred EMPTY>
    <!ATTLIST pred v IDREF #REQUIRED>

<!ELEMENT inactivate (pred+)>

<!ELEMENT resets (reset+)>
    <!ELEMENT reset EMPTY>
        <!ATTLIST reset v IDREF #REQUIRED>

<!ELEMENT srcareavpath EMPTY>
    <!ATTLIST srcareavpath v CDATA #REQUIRED>

<!ELEMENT eastartop EMPTY>
    <!ATTLIST eastartop op (set|append|delete) #REQUIRED
                        name CDATA #REQUIRED
                        value CDATA #REQUIRED>

<!ELEMENT eafinishop EMPTY>
    <!ATTLIST eafinishop op (set|append|delete) #REQUIRED
                         name CDATA #REQUIRED
                         value CDATA #REQUIRED>

<!ELEMENT timeout (succ+)>
    <!ATTLIST timeout v CDATA #REQUIRED>
```

D File Status Codes

A number of operations result in messages containing file status codes. Those codes are:

```
ERROR_START         = 100
SERVER_ERROR        = -1
SUCCESS             = 0
ERROR               = ERROR_START + 1
THROWABLE           = ERROR_START + 2
INTERNAL_ERROR      = ERROR_START + 3
CONFIGFILE_ERROR    = ERROR_START + 4
UNKNOWN_NAME        = ERROR_START + 5
ROLE_DENIED         = ERROR_START + 6
AUTHENTICATE        = ERROR_START + 7
ALREADY_EXISTS      = ERROR_START + 8
ACCESS_INIT         = ERROR_START + 9
FILESYS_INIT        = ERROR_START + 10
WORKFLOW_INIT       = ERROR_START + 11
UI_INIT             = ERROR_START + 12
SESSION_STRING      = ERROR_START + 13
CREATEWF_FAILED     = ERROR_START + 14
NULL_INPUT          = ERROR_START + 15
PNO_ERROR           = ERROR_START + 16
LOCK_ERROR          = ERROR_START + 17
WAPNO_ERROR         = ERROR_START + 18
UNLOCK_ERROR        = ERROR_START + 19
TASKID_ERROR        = ERROR_START + 20
ADDTASKFILE_ERROR   = ERROR_START + 21
RMTASKFILE_ERROR    = ERROR_START + 22
SUBMIT_ERROR        = ERROR_START + 23
UPDATE_ERROR        = ERROR_START + 24
FILEKIND_ERROR      = ERROR_START + 25
OFFSET_ERROR        = ERROR_START + 26
CRFILE_ERROR        = ERROR_START + 27
LKFILE_ERROR        = ERROR_START + 28
OFFSETFILE_ERROR    = ERROR_START + 29
LOCKED_ALREADY      = ERROR_START + 30
UPDATE_FAILED       = ERROR_START + 31
IS_NOT_DIRECTORY    = ERROR_START + 32
NOT_ABSOLUTE_PATH   = ERROR_START + 33
WRITE_MODE_ERROR    = ERROR_START + 34
INACTIVE_TASK_ERROR = ERROR_START + 35
INVALID_INPUT       = ERROR_START + 36
TEMPLATING_CFG      = ERROR_START + 37
```