IBM Content Manager CommonStore for Lotus Domino

IBM Content Manager CommonStore for Lotus Domino Administrator's and Programmer's Guide

Version 7 Release 1

SC33-7053-00

IBM Content Manager CommonStore for Lotus Domino

IBM Content Manager CommonStore for Lotus Domino Administrator's and Programmer's Guide

Version 7 Release 1

SC33-7053-00

Note
Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

(First Edition December 2000)

This edition applies to IBM Content Manager CommonStore Version 7 Release 1 (product number: 5724–A30) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct documentation level for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for your comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3591
Schönaicher Strasse 220
D-71032 Böblingen
Germany

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user. IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Any references in this publication to Web sites are provided for convenience only and do not in any manner serve as an endorsement of these sites. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site mentioned in this publication.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ADSTAR
AIX
EDMSuite
IBM
OS/390
OS/400
S/390
System/390
VisualInfo

The following terms are trademarks of other companies:

Trademark	Company Name
Domino	Lotus Development Corporation
Lotus	Lotus Development Corporation
Lotus Notes	Lotus Development Corporation
Lotus Script	Lotus Development Corporation
Notes	Lotus Development Corporation
R/3	SAP AG
SAP	SAP AG

Tivoli is a trademark of Tivoli Systems, Inc., in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Incorporated, in the United States and/or other countries.

Pentium is a trademark of Intel Corporation.

Windows and Windows NT are registered trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.

About this book

This document is targeted to the administrators and application programmers of IBM Content Manager CommonStore for Lotus Domino (hereinafter: CSLD).

The reader should be familiar with the basic concepts of Lotus Notes administration and development and Content Manager administration.

The manner in which documents are archived using CSLD depends entirely upon the given Notes application. For this reason, this document is not to be considered as a user's guide for a CSLD-enabled Notes application.

Product names

Throughout this book, the brand name ″VisualInfo″ refers to the products ″IBM EDMSuite VisualInfo″ and ″IBM Content Manager″. The name ″OnDemand″ refers to the products ″IBM EDMSuite OnDemand″ and ″IBM Content Manager OnDemand″,[1] and the names ″ADSM″ or ″TSM″ refer to ″ADSTAR Distributed Storage Manager″ and ″Tivoli Storage Manager″.

Highlighting conventions

Throughout this book, *italics* are used for
- book titles
- emphasis
- options / variables / parameters / keywords

Boldface is used for
- check box labels
- choices in menus
- column headings
- commands and subcommands
- entry fields
- field names in windows
- forms and subforms
- index classes
- items
- menu-bar choices
- menu names
- radio button names
- spin button names
- statements
- tables.

`Monospace` is used for
- coding examples
- directory / folder names
- entered data
- file names
- group and user IDs
- message text
- path names
- transaction codes (T-codes)

Underlined bold indicates
- default values.

1. For convenience, Content Manager OnDemand is also referred to in this document as CMOD.

Further information and suggestions

Additional information about CSLD is available on the ESD home page:
`http://www.ibm.com/software/data/commonstore/`

You can send comments and questions about the software by e-mail to
`cstore@de.ibm.com`

Be sure to state the Version and Release of CSLD you are using.

To comment on this book, fill out the form at the back and return it by mail, by fax, or by giving it to an IBM representative.

Part 1. CSLD Overview

Chapter 1.1 What is CSLD?

This chapter gives a rough overview of the components in the CommonStore for Lotus Domino system. The components shown in the figure below are briefly described in the following sections.

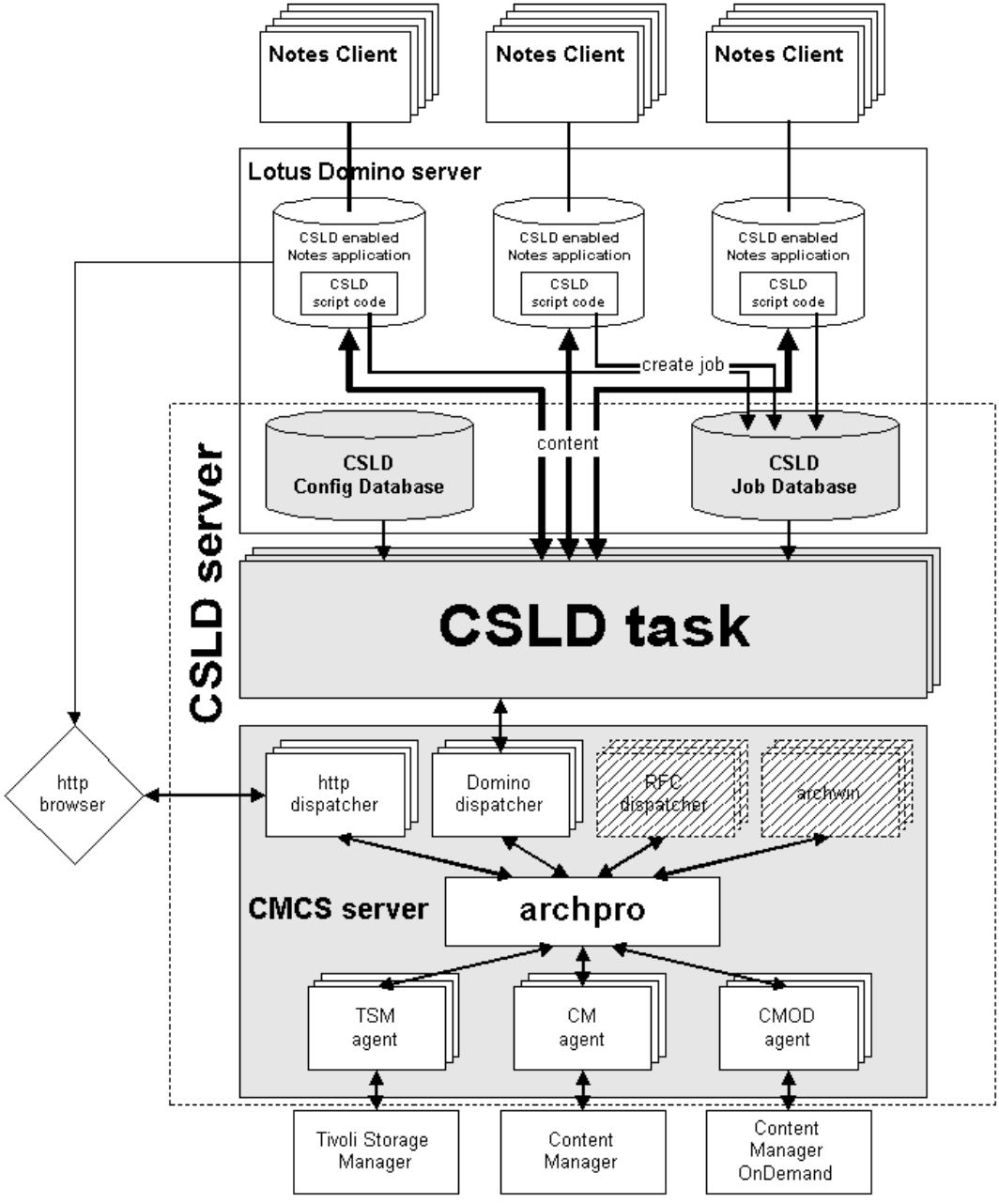

Figure 1. Components in a CommonStore for Lotus Domino system

Figure 1 shows the various components in a CommonStore for Lotus Domino system. The hatched boxes RFC dispatcher and archwin are needed only when using CommonStore for SAP.

Archpro

The Archpro is the heart of CSLD. It maintains a list of archives it is connected to, and controls the flow of information to and from these archives. All input and output is routed through the Archpro. For example, whenever content is to be archived, an archiving request is sent to the Archpro, along with the content and the content's descriptive information.

The Archpro is an application independent archiving engine. It does not really know where the content comes from, what format the content has, or any other semantic information about the content. Although the Archpro is responsible for handling Domino-related content, it has nothing to do with Notes/Domino, i.e. it does not call any Notes/Domino related code.

The Archpro does not handle the communication with the archives itself, i.e. the archpro is not an archive client. It maintains a queue for archive-dependent worker agents, and load balances all requests among these agents.

Agents

The agents are the interface to the archive. Every agent is an independent archive client process. For every archive supported by CSLD, there is a special agent. An agent calls the archive's native client code, that is, the TSM client API, OD client API and Content Manager client API. Of course, the client code must be installed on the CommonStore server.

Multiple agents, even for the same archive, can run in parallel. An agent gets its orders from the Archpro. When all agents are busy, the request is held in the Archpro's queue until an agent becomes available. Every agent keeps one connection to the archive. The more agents, the more requests can be processed concurrently. However, you should not run too many agents because every agent requires system resources.

Agents always run on the same machine as the Archpro and are automatically started by the Archpro.

Archives

Tivoli Storage Manager, Content Manager, or Content Manager OnDemand server. Archives are accessed by the agents.

Domino Dispatcher

The Domino Dispatcher is the interface between the application independent Archpro, and the CSLD Task, the Domino dependent part of the system. It translates Domino requests into the language of the Archpro. For every data producing/consuming system that CommonStore supports (currently Domino, SAP, and a Web browser) there is a special dispatcher. This ″pluggable″ scheme allows to attach new data producing/consuming systems to CommonStore by just writing special dispatchers for them. It also ensures that the Archpro is application independent.

The Domino dispatcher is a multithreaded application, allowing multiple CSLD Tasks to connect to it and send requests. It runs on the same machine as the Archpro, and is automatically started by the Archpro.

Http Dispatcher

The Http Dispatcher implements a set of http commands which map the main functions of the Archpro to a browser accessible interface. This allows CSLD to replace an archived attachment by a special URL link in the document and make it viewable by an ordinary Web browser without restoring it to the document. Hitlists and result documents resulting from a query also contain such a URL link.

The Http Dispatcher runs on the same machine as the Archpro and is automatically started by the Archpro.

CSLD Task

The CSLD Task is the only Notes-related part of the system. It is a server process that periodically polls a so-called job database for jobs. Jobs are requests to CSLD. The CSLD Task converts Notes documents to files that are archived and vice versa. According to the request stated in the job, it sends the request along with the file to the Domino Dispatcher, which in turn forwards it to the Archpro. In CSLD, there are archiving tasks and retrieval tasks. Archiving tasks process requests that add or modify content in the archive (e.g. archiving, updating, deleting). Retrieval task process requests that retrieve content from the archive (e.g. searches or single document retrieval). CSLD tasks are not Domino server addin-tasks, but standalone applications.

Multiple tasks can run on one machine. CSLD tasks do not necessarily have to run on the same machine as the Archpro, but we recommend it for performance reasons. For performance reasons, we do not recommend to run CSLD on the same machine as the Domino server.

Since the task is a Notes application, it requires a Notes runtime. Therefore, a Notes R5 client (or Domino R5) must be installed on the same machine. CSLD tasks read their configuration information from the configuration database.

The CSLD Tasks are not automatically started by the Archpro. They are separate processes and must be started manually.

Job Database

The job database is the only interface between a CSLD-enabled Notes application and CSLD. That is, for every request made to CSLD, a job document must be created in the job database. Usually, this is done via LotusScript from within the Notes production application. In large CSLD environments, there can be multiple job databases. A job database can be located on any Domino server.

Configuration Database

A regular Notes database that contains configuration information for CSLD tasks. Multiple CSLD tasks usually share a single configuration database. Tasks read in their configuration during startup. A configuration database can be located on any Domino server.

CSLD-enabled Notes application

A regular Notes production database containing script code to create CSLD job documents. It is up to the Notes application to determine when and how documents are archived.

Chapter 1.2 Features and Scenarios

This chapter gives a short overview of the CSLD features and the scenarios that can be handled by CSLD. The list is divided into two parts. The first part lists the features existing since the last release and the second part lists the new features of CSLD release 7.1.

Features since CSLD Release 2.1

- Archive or offload any kind of Notes documents.
- Archive/offload attachments, optionally detach attachments after archiving.
- Archive documents in binary "Notes native" format. Allows to restore the original document completely.
- Archive "images" of Notes documents in Microsoft RTF or ASCII format for long-term archiving.
- Optionally delete archived documents.
- Store descriptive Notes fields in archive index fields for searching purposes.
- Archive complete Notes views and folder structures (folders with all their subfolders).
- Perform searches over the archive.
- Archive interactively or scheduled.
- Retrieve search results as one hitlist document or multiple result documents, each representing one hit.
- User interface is 100% Notes. No client code needs to be installed on user desktops.
- Interface from user databases to CSLD is fully Notes based. Send all requests to CSLD as regular Notes documents.
- Configuration managed from within Notes via configuration database.
- Supported archives: IBM Content Manager and Tivoli Storage Manager.
- Define target archives for Notes documents based on document types (form).
- Define target archives for Notes documents based on rules.
- "Private data": Protect your archived documents from other users. Very helpful for mail archiving.
- Protect retrieved data in your Notes database from being seen by other users via Readers fields.
- User security exit: Map Notes user IDs to archive user IDs.

- Ready-to-use LotusScript libraries to make use of CSLD functionality within your application.
- Mail archiving sample application containing ready-to-use code, actions and other design elements for standard archiving tasks.
- Heavy multithreading: Archive or retrieve to multiple databases concurrently for maximum throughput.
- Archive every possible Notes server platform with one CSLD server.
- Intelligent scheduling of CSLD request processing.
- Use CSLD as Notes frontend to documents archived by other applications (e.g. SAP, scanning applications).

New features in CSLD Release 7.1

- Content Manager folder browsing: The contents of Content Manager folders can be displayed in a hitlist.
- Browser viewing: Document archived with CSLD can be displayed in a browser without retrieving them in Notes.
- Content Manager OnDemand support.
- Better scalability: Tasks can now be configured to process jobs of entire Domino servers and database directories.
- Archiving and restoring complete Notes folder structures: entire Notes folders including subfolders can be archived and completely restored preserving folder structure.
- Restoring of document UNID: When documents archived in Notes native format are retrieved, the original document UNID is preserved.
- Agents for automatic processing of archived or retrieved documents.
- Content Manager workbasket support and OnDemand ″virtual workbasket″ and ″virtual folder″ support.
- Document Rearchiving: You can now archive the attachments of a document, attach new ones, and archive the new attachments.
- CSLD Tasks can now be started without typing in passwords.
- Raster User Exit: Raster engines implementing the new ″raster user exit″ can be installed to convert documents (including their attachments) to various formats like TIFF.
- Default content types: Attachments with unknown file extensions can be mapped to a default content type.
- Retrieved documents can be made response documents to the hitlists from which they were retrieved.
- Easier license management.
- Simplified configuration for Tivoli Storage Manager archives.
- CSLD can now be configured to send users an email when their jobs fail.

Part 2. CSLD Concepts

Chapter 2.1 User IDs and roles

CSLD defines the following IDs and roles:

CSLD Administrator

The Notes ID that manages the Notes-related parts of CSLD. The tasks of the CSLD Administrator include:

1. CSLD installation
2. creating configuration database and job database(s). The CSLD Administrator has Manager access rights to these databases
3. creating the documents in the configuration database
4. managing access rights to the job database
5. starting and shutting down CSLD tasks and the Archpro (CommonStore server)

The CSLD Administrator should not be a single user ID. We recommend it to be a group of administrators.

Archive Administrator

The user that manages the archives behind CSLD. The tasks of the archive administrator include:

1. Archive installation
2. managing user IDs within the archive
3. setting up access rights.
4. creating index classes, workbaskets and content types (CM), application groups (OnDemand) and management classes (TSM) for CSLD
5. managing the archint.ini file which contains archive-related definitions

Typically, the archive administrator and the CSLD administrator are not the same person. However, they must closely work together to define which Notes fields are mapped to which index fields, and which Notes document types are mapped to which archive document types (CM index classes or OD application groups).

The CSLD User ID

The Notes specific part of the CommonStore server is called CSLD task. It is a regular Notes client application named csld.exe, written on the C++ and C

API. Every Notes application runs under a certain ID, which is stored in the *notes.ini* file. For example, when you use your Notes client to switch to the ID of user A, the Notes client runs under id of user A. The *notes.ini* file will then contain an entry pointing to user A's ID file. The CSLD user ID is the ID the CSLD task is running under. It is **not** the ID of an user working on a CSLD-enabled production database. CSLD accesses production databases on behalf of the CSLD user ID. The CSLD user ID must have the right to read documents (to archive them), create new (retrieved documents), and modify documents in the production database. Thus, the CSLD user must have at least Editor rights on every CSLD-enabled database. Make sure the CSLD user ID is not a regular Notes ID nor the CSLD administrator ID, as it can read and modify documents. In case of email archiving, this means that the CSLD user ID can read the email of probably thousands of users.

You can choose the CSLD user ID to be the server ID, which is usually already part of your database ACLs. Most companies define a group consisting of server IDs. This group typically has Manager access to every database. The simplest way to grant the CSLD user ID access to all CSLD enabled databases would therefore be to add the CSLD user to the server group. However, you can also modify the ACL of the database templates to contain the CSLD user ID. After the next design refresh (manually or via server add-in task) the CSLD user ID will be able to access the databases.

Domino Administrator

Manages the Domino servers running the production databases as well as the CSLD configuration and job database(s). In smaller companies, the CSLD administrator and the Notes administrator are usually the same person. Major companies usually have a sophisticated, centralized administration scheme with many Domino administrators, having different access rights.

The CommonStore Archive User ID.

The user ID under which CSLD logs on to the archive server. I.e., it is an archive user ID, not a Notes ID. When CSLD archives a document, it reads the document from the database via the CSLD user ID, but stores the document in the archive under the archive user ID. For every defined archive, the CommonStore server ID is defined in the *archpro.ini* file.

The [CSLDUsers] Role

In a large company it makes sense to define more than one CSLD user ID. In Notes application programming, it is a good practice to use roles instead of managing hardwired user IDs in databases. The CSLD job database defines the role [CSLDUsers] which stands for all CSLD user IDs (not regular Notes users!) that access the job database. Therefore, besides adding CSLD user IDs

to a job database ACL, the CSLD administrator must assign every CSLD user ID to the [CSLDUsers] role. When a user creates a job using the CSLD script classes, the job is protected from other users via a Readers field. However, the CSLD user ID, under which the task is running, is just another ID having access to the job database. In order to process jobs, the CSLD task must be able to see them. Thus, the Readers field in the jobs must contain the [*CSLDUsers*] role. Otherwise, the jobs will never be processed.

We highly recommend to use the CSLD script classes (Script libraries *createCSLDJobs* and *CSLDJobSamples*). This will ensure that jobs will be created correctly.

CSLD-enabled Notes User

A regular Notes user allowed to make use of CSLD functionality in a CSLD-enabled production database. It is up to the CSLD application developer to decide which users can make use of CSLD functionality. For example, a production database could define the groups Archivers and Retrievers. Only users in the Archivers group would be allowed to archive documents, and only users in the Retrievers group would be allowed to retrieve documents. All other users would simply not see any CSLD design elements. One way of doing this is to hide design elements from certain users via hide-when formulas.

Generally, every CSLD-enabled Notes user must be able to create jobs in the job database. Therefore, CSLD-enabled Notes users must have Author access rights to the job database. It is the task of the CSLD Administrator to manage the job database ACL.

We recommend to maintain a group called, say, *CSLDEnabledUsers*. Instead of adding every single user to the job database, add the user to the group and add the group once to the job database ACL.

CSLD Application Developer

Enables an existing Notes production database for CSLD use. A CSLD application developer typically adds design elements like views (e.g. search results), actions (e.g. archiving buttons) or agents (for scheduled archiving) to the database, so he must have Designer rights. The modified application template is then handed to the Domino administrator who deploys the application within the company.

Chapter 2.2 How to use CSLD services

In CSLD, all archive, update, delete, or retrieve services are used by creating *CSLD jobs*. A CSLD job is a regular Notes document with a defined set of parameter fields. CSLD jobs reside in a so-called *job database* from which they are processed by CSLD. CSLD does not specify how jobs are created. There are two methods of creating CSLD jobs:

Creating CSLD jobs from Lotus Script code running in the Notes application database

This script can:

- run in an agent that can be scheduled (e.g. archiving a certain view every day at midnight) or manual (e.g. by selecting a number of documents in a view and starting an agent, e.g. in the Plug 'n' Go Mail Demo via the path **Actions -> CSLD --> archive selected documents**);
- be triggered interactively via actions (e.g. in the Plug 'n' Go Mail Demo by pressing the **Archive** button in a document's action bar);
- be triggered automatically via Notes events (e.g. **Querysave**) or any other Notes method for invoking Lotus Script code.

The CSLD **CreateCSNJobs** Lotus Script Library supplies standard classes and functions to create jobs for CSLD services.

Creating CSLD jobs from an application based on the Notes C, C++, or Java API

This application creates documents in the job database and fills document items with request parameters. The application can run as an independent, stand-alone application or as a server add-in task.

Chapter 2.3 Configuration Database

2.3.1 Mapping Notes forms to archives

Document mappings

In Lotus Notes, the type of a document is given by its *form*. Basically, a form is a collection of fields, also called *items*, that have a name and a type. In Lotus Notes, the user sees only items of the data types

- ″Text″
- ″Number″
- ″Date/time″ or
- ″Rich Text″

For example: All documents describing CDs usually have fields named **Artist**, **Album Name**, **Catalog Number**, etc.

At the same time, all documents in a Content Manager or CM OnDemand archive have a common set of attribute names. Like in Notes, these attributes have different types.

Obviously, in order to archive a Notes document, Notes forms must be mapped to archive containers. Also, Notes fields within a form must be mapped to attributes within an archive container. This mapping is done by defining a *document mapping* in the CSLD configuration database. A document mapping specifies the Content Manager index class or OnDemand application group to which Notes documents of a given form are archived. However, a document mapping does not specify the archiving method (attachment archiving, native archiving, or rasterized). A single index class or OnDemand application group can contain Notes documents archived using different archiving methods.

When documents have been archived (regardless of the archiving method used), the Notes fields listed in the document mapping are extracted from the document and taken as attributes. These attributes are used to find the document later by means of queries.

Notes forms sometimes define alias names. For example: The mail form **Memo** has the two alias form names **New Memo** and **Document**. CSLD allows alias form names to be specified in a document mapping, so that CSLD will automatically find the right mapping for a document whose form name is one of the alias form names.

Important!

With CSLD, it is possible to map different forms to the same Content Manager index class or application group, provided all the forms have a number of fields in common with the same name and type. To do this, simply define document mappings for each form to the same index class or application group. However, documents in a particular index class or application group represent documents of the same type. Once a document is in the archive, all information about the original Notes form is lost. Thus, when a document is retrieved from an index class or application group, CSLD will take the Notes form defined in the first document mapping for the index class or application group.

Example:

Suppose that you have mapped forms **A**, **B**, and **C** to the same index class or application group, and that, in the document mappings view of the CSLD configuration database, **A** appears before **B** and **C**. Documents retrieved from the index class or application group will then always be created using form **A**.

However, if you want retrieved documents to be created using form **B**, you must take a different approach: Instead of defining three document mappings for **A**, **B**, and **C**, you can also define one document mapping for form **B**, and add forms **A** and **C** to the aliases field of the mapping. This is possible even when **A** and **C** are not ″real″ form aliases for **B** (that is, when they are not listed in the **Aliases** field of the form's properties [**Edit ->Properties...** or the SmartIcon properties]). They are aliases in the sense that they have a number of fields in common that are mapped to the same index class or application group.

When a document is archived, CSLD determines the document mapping using the following algorithm:

- If the document has a form item containing **<formname>**, the document mapping for **<formname>** is taken.
- If the document contains the form embedded, the document mapping for the embedded form name is taken.
- If the document has a form item **<formname>** but there is no document mapping for **<formname>** defined, CSLD checks whether or not **<formname>** is an alias defined in a document mapping for form **<otherform>**. In this case, the document mapping for **<otherform>** is taken.
- If the document has no form item, the database default form is taken.

- If the database does not define a default form either, CSLD cannot determine the document type and returns an error message.

Example:
For e-mail archiving, the form names **Memo**, **New Memo**, and **Document** are used, where **Document** and **New Memo** are aliases for **Memo**. Suppose a document mapping for the form **Memo** is defined. Then, if a document of form **New Memo** or **Document** is to be archived, the document mapping for **Memo** is taken. The same mapping is taken if a document has no form set at all, because **Memo** is the default form of the mail template.

In order to be able to search for or to display retrieved archive documents, query and result forms for every mapped Notes form must be created. You can either define these forms yourself or use the setupDB tool which is shipped with CSLD to create these forms for yourself. Regardless of the manner in which these forms are created, note that only the mapped fields will be considered in search requests or retrieve results, respectively.

Example:
Suppose that you have an index class named **CD** defining attributes for **Title**, **Artist**, **Duration**, **Release Date**, and **Label**, but that only **Title**, **Artist**, and **Release Date** were mapped to Notes document fields. As a consequence, the attributes **Duration** and **Label** can neither be searched for nor will they be returned in a search or retrieve result.

Note:
Besides the archive attributes defined to contain document data, CSLD does not require or create any special archive attributes to maintain status information. That is, you can easily integrate CSLD into an existing archive without reconfiguring the archive.

Special mappings

When a Notes document is archived, the target is determined by the document mapping for the document's form. That is, all documents of that particular form would be stored in the same index class or application group. However, sometimes you need to store special documents in special containers according to a particular rule. For example: In a bank, some customers could be considered 'more important' than others, and information about these users could be stored in containers with access restricted to

'Managers', only. Or, in a record label, the administrator might want to store classic, rock and jazz music CDs in separate containers.

One possible strategy would be to leave it up to the user to decide which archive to put the document into (e.g. by selecting a container name from a list box). However, in most cases, an automatic mechanism is more effective and less error-prone. Special mappings allow a container based on the value of a Notes document field to be automatically selected.

Example:
Suppose that the **AudioCD** Notes form contains a **MusicStyle** field indicating the style of music. You can then define special mappings evaluating this field. One special mapping would store all jazz CDs in a Content Manager index class named **Jazz**, a second special mapping would store all Rock CDs in a second index class, etc.

You can basically define as many special mappings as you want. However, the more containers that are defined, the less efficient the mapping will be, and the less efficient queries will be executed.

You can define either an exact value or a value range within a special mapping. For example, you can define a special mapping that stores documents in the Classic Music container if the **Music Style** field contains the exact text "Classic". Or, to give an example for a value range, you could define a special mapping that maps all bank accounts of "important people" to a particular "VIP container" when the **AccountNumber** field is between, say, 1900000 and 2000000.

2.3.2 Mapping Notes document fields to archive attributes

Data types

Based on the document mappings defined in the configuration database, index information for archived documents is extracted automatically from Notes documents. This index information is used to identify archived documents and find them in queries. Fields in Notes documents may have different types but not all of them can be used to map them to archive attributes. The supported field data types are "text," "number," and "date/time." The use of the supported data types is described below.

Text fields

A text field is represented by a character string. They contain alphanumeric characters. The multibyte content of a text field is passed on to the archive as is. Text fields can be mapped to Content Manager attributes of the type "character" and "variable character."

Number fields

Number fields contain any kind of numerical value. Number fields can be mapped to Content Manager attributes of the type ″integer,″ ″long integer,″ or ″decimal.″

Date/time fields

In Lotus Notes, date/time fields can be created in three different formats: date-only, time-only, or with a date-and-time value (timestamp). The Content Manager attribute type to which a Notes Date/Time field is mapped depends upon the field's format:

- Date-only fields are mapped to Content Manager attributes of type ″date.″
- Time-only fields strings are mapped to Content Manager attributes of the type ″time.″
- Date-and-time (timestamp) fields are mapped to Content Manager attributes of the type ″timestamp.″

A special characteristic of date-and-time (timestamp) values is their dependency upon a particular locale. If a time field is set in Germany to the 25th of August, 20:45, the same field displays as 2:45 in the afternoon when viewed from a Notes client in New York and as 2:45 on the morning of August 26th in Bangkok. Since the way a date/time value is stored also affects the way it will be used in searches, *CSLD always stores date/time values in the locale the date/time value was archived in*, rather than transforming them into a generic time format. This way, users searching for these documents from Notes as well as from native archive clients can search for their local times without having to transform them as long as they are in the same locale. Besides that, there are always two things that can be done to prevent confusion with date/time values:

- Configure time fields in Notes to always display the time zone. This way, any user can see what value will be stored to the archive.
- Do not distribute archive servers across time zones.

Field length

In Content Manager or OnDemand, attributes can be defined with variable length or fixed length. In both cases, a maximum length must be defined.

When a document is retrieved from an archive, archive attributes are mapped back to Notes fields. In case of fixed-length attributes, the resulting Notes field will have the length the attribute has been defined with. That is, the Notes field is padded with blanks to reach the fixed length.

When a Notes field is longer than the maximum length the corresponding attribute has been defined with, Content Manager or OnDemand will not archive the document and will return an error message. For example: Suppose

that you have retrieved a document back to Notes and that the document contains fields mapped to fixed-length attributes. Then, when you modify and update a fixed-length field in the Notes document, you should take care that a modified field is not longer than it was before modification.

2.3.3 Content types and filenames

When documents are archived, information about the data format of the documents must be stored. This allows client applications to interpret a document correctly. Therefore, with every document, CSLD stores a *content type*. The content type is the only information about the document's format, and is somehow similar to the MIME type that Web servers (HTTP servers) provide when content is downloaded. It depends on the archive how CSLD stores the content type. In order to determine the content type of a document, CSLD provides the content type mapping table in the configuration database. This table maps file extensions to content types, and must be filled in by the CSLD administrator. The archives always store the content type in this table.

Besides the content type, CSLD stores the original filename of a document. This allows CSLD to restore the filename of archived documents when they are retrieved back to Notes. It depends on the archive where this information is stored.

Content Manager

IBM Content Manager generally does not store filenames by itself. CSLD stores the filename of a document in an internal attribute, which is not visible to the ″outside″ (that is, this attribute need not be created in an index class).

- CM stores the content type provided by CSLD natively with the document. That is, no attributes must be defined to store the content type.

OnDemand

Like Content Manager, OnDemand does not store filenames by itself. For this reason, CSLD stores the document's original filename in an attribute named *ORIGFILENAME*, which must exist in every application group CSLD archives documents to. This allows CSLD to reconstruct the filename of archived attachments, when they are retrieved back to Notes. While content types are a native construct built-in into Content Manager, in OnDemand content types are ″emulated″ by CSLD: The document's content type is stored in an attribute named *CONTENT_TYPE*. This attribute must exist in every application group CSLD stores documents to.

Tivoli Storage Manager/ADSM:

CSLD stores the document's content type internally (not visible to the user).

How CSLD determines content types

Independent of the archive used, CSLD determines the content type, under which to archive a document, according to the following algorithm:

1. CSLD looks at the file extension of the document. The file extension is .csn for archiving in native format, .txt for archiving in ASCII format, .tiff for archiving in TIFF format (via Compart DocBridge), or any other file extension for archiving attachments ″as is″.
2. CSLD checks whether the content type mapping table in the configuration database contains an entry for the given file extension.
3. If yes, the document is stored under the configured content type.
4. If no, CSLD checks whether the content type mapping table contains an entry for the ″default file extension″ .unk (unknown).
5. If yes, the content type given here is used.
6. If no, CSLD will not archive the document and will return an error indicating that it cannot determine a content type. In addition, a mail is sent to the CSLD administrator (if configured) to inform him that a mapping for the file extension must be created.

That is, in theory, you could simply add one default content type for the ″default file extension″ .unk. All documents would then be stored under this content type. However, every information about the document's data format would get lost. Archived documents could also be accessed via the Content Manager or OnDemand client, or any external archive application. No archive application except CSLD would then be able to interpret the archived content.

Note

Another reason we highly recommend to maintain different content types for different file formats is the CSLD browser viewing feature, which uses the content type to determine the MIME type of a document. **With only one content type, the browser would not be able to display the data correctly**.

When documents are retrieved, CSLD checks whether a filename has been stored with the document. If yes, the filename is reconstructed completely. If not, a temporary name is created. The file extension is determined by the content type mapping table. If the content type of the retrieved document is not listed in the mapping table, it is given file extension .unk (unknown). This can happen only for documents not archived via CSLD, or if a content type mapping has been removed from CSLD after the document has been archived.

Chapter 2.4 Archiving and retrieval tasks

CSLD tasks are the heart of CSLD. These server processes implement the entire functionality for all CSLD services. There are two types of tasks:

archiving tasks
These tasks are responsible for all services that add or modify content in an archive.

retrieval tasks
These tasks are responsible for all services that retrieve content to a user database.

Both archiving and retrieval tasks periodically poll the job database for jobs.

When a CSLD tasks finds a job, the job state is set from ″Waiting to be processed″ to ″Pending″. The job is then processed. Finally, if no errors occurred during job processing, the job state is set to ″Finished successfully,″ otherwise it is set to ″Error″.

CSLD tasks run under the ID of the CSLD user (by passing the CSLD user's `notes.ini` file to the task executable).

Polling the user database

The CSLD administrator can specify the weekdays and the start/stop time of polling. Also, a polling frequency must be set. The polling frequency specifies how often a task polls the job database for jobs. For example, a task can be configured to poll every 10 seconds for jobs between 6 am and 8 pm every Monday and Wednesday.

Usually, as users desire short response times for queries/retrieval, the polling frequency for retrieval tasks is set to a short time (e.g. 5 seconds), while the polling frequency for an archiving task can be set much lower. Sometimes it is even sufficient to start archiving once a day or week.

CSLD can be configured to delete jobs that have been processed successfully. But in spite of deleting finished jobs from the Notes database, this database will still grow in size, thus slowing down access to it. Therefore, the job database must be compacted once in a while. Compacting databases is possible only if they are not being accessed at the moment. That is, the CSLD administrator should compact the job database only if no CSLD task is currently polling on it. Otherwise, the CSLD task stops and must be restarted

after compacting. An empty time slot for compacting can be found by checking the polling times in the profiles stored in the configuration databases.

Assigning tasks to databases, servers, or data directories

Every task is assigned a profile in the CSLD configuration database. A profile is a named set of parameters for a task. When a task is started, it reads in the parameters defined in the profile. In a profile, every CSLD task is assigned to a particular job database. Also, every CSLD task is assigned to a certain "source". A source can be a single database, an entire Domino server, or a data directory containing a number of databases.

A CSLD task searches for jobs and processes them only for the source they are assigned to.

Note

It is not possible to assign two or more tasks to the same source and the same job database. This is because Notes does not supply a document locking mechanism. That is, it is not possible to start two tasks with the same profile. See "Defining profile documents" on page 125 for more details.

Chapter 2.5 Performance

The overall performance of CSLD depends heavily upon the following factors:

Performance of the Domino Server

Network performance
Even the fastest servers will not help you when your network is overloaded.

The number of CSLD tasks per machine

The number of databases to which a CSLD task is assigned
For every source a CSLD task is assigned to, it processes jobs and therefore uses additional system resources. Therefore, do not assign too many sources to a single task. Instead, assign some databases to a new task. Especially databases producing many jobs should be assigned to their own task.

Processor speed
On a Pentium II 300 MHz machine with 158 MB RAM, the processor load during native archiving for a task assigned to one database is about 5%. However, since CSLD is a Notes and Content Manager client, most of the processor time is spent for I/O rather than ″real computation″.

Physical memory
Avoiding system swapping by adding additional memory will increase performance drastically.

Archiving strategy
Archiving a thousand documents by creating a thousand jobs with one document each will be much slower than archiving all with one job.

Scheduling of polling
Polling the job database unnecessarily will increase the load of your Domino server and therefore lower overall Notes performance.

The average number of jobs per polling period
Idle CSLD tasks fall asleep between two polls and release system resources. The lower the polling frequency, the more CSLD tasks can be started. However, a low polling frequency can decrease the response time for a job because the CSLD task will find and process the job later.

The distribution of CSLD tasks, Notes Server and CSLD databases

- Other processes/applications running concurrently on the same machine will slow down CSLD tasks.
- We do not recommend running CSLD tasks on Domino servers.
- When CSLD tasks run on the Domino server machine, they can access the job database and user database more efficiently. However, a short ″distance″ between CSLD tasks and user databases has a greater impact upon archiving performance than a short distance between CSLD tasks and job databases.

The number of CMCS Agents

See also “Chapter 2.6 Scalability” on page 31.

Performance of the archive server

See also “Chapter 2.6 Scalability” on page 31.

Chapter 2.6 Scalability

CSLD supports scaling on three different levels:

1. scaling on the job database level,
2. scaling on the CMCS agent level, and
3. scaling on the CMCS server level.

These three different levels are explained in the following sections.

Scaling on the job database level (using more than one job database)

As stated before, it is not possible to start two or more CSLD tasks with the same profile (that is, same source and job database). However, two or more tasks can be assigned to the same source if every task has its own job database. To make use of additional job databases, the archiving application must distribute jobs among the job databases. This can be achieved, for example, by using random numbers or a counter combined with a MOD operator (next job database number = current counter MOD number of job databases). Using several tasks and job databases for archiving may increase archiving performance.

Scaling on the CMCS level

Using more than one CMCS agent/dispatcher

Within CMCS, there are two components that can be scaled in order to meet the customer's service requirements: the *Domino dispatcher* and the *CMCS agent*.

The Domino dispatcher

The Domino dispatcher is a component which takes requests and routes them to the CMCS server. The number of dispatchers started is determined by the parameter *DOMINODPS* in the CMCS configuration file `archint.ini`. As the number of sources in one profile increases, the number of dispatchers should likewise be increased using this parameter. Since CSLD starts a thread for each source in a profile, it is possible that a large number of jobs will be found concurrently, resulting in a lot of requests being sent to the CMCS at the same time. To ensure that all of these concurrent requests can be processed, the number of Domino dispatchers waiting for them must be increased. When the number of Domino dispatchers started is too low, communication problems with the CMCS server might occur, resulting in erroneous jobs.

The CMCS agent

The CMCS agent is the entity responsible for communicating with the archive. For every supported archive (i.e. CM, CMOD and TSM), there is a special type of agent. CMCS can start a configurable number of agents. The number of agents is configured via the parameter *VIAGENTS* (or ADSMAGENTS or ODAGENTS, as the case may be) in the `archint.ini` file. All work is distributed among these agents. With CSLD, the distribution is on a per source and per task basis. That is, if there are as many tasks as agents, every task will be served by its own agent. Also, if a task is assigned to *n* sources via its profile, and there are *n* agents, every source processed by this task is served by its own agent. If there are fewer agents than sources or tasks, requests to agents will be put in a queue until an idle agent becomes available. Starting more agents than tasks/sources makes no sense and will result in unused agents wasting system resources.

Note:

Every agent is a process. Therefore, starting too many agents will not improve performance, but rather will cause system swapping. In any case, do not start more than ten agents.

Starting more than one agent makes sense only if the archive server performance is high enough to handle all requests by agents at the same time. For example: Having a large number of agents archiving a lot of data into a slow server will not increase performance.

In order to increase the performance of CSLD, it is necessary to increase both the number of tasks/job databases *and* the number of CMCS agents/dispatchers (as the case may be).

Note that every CSLD task must have its own export directory defined in the task's profile.

Using more than one CMCS server

Every task is assigned to a CMCS server via an entry in the task's profile. Usually, the CMCS server runs on the same machine as CSLD tasks, and multiple CSLD tasks are assigned to the same CMCS server. However, for maximum performance, every CSLD task can be assigned to its own CMCS server running on a separate machine. Every CMCS server can even have its own archive. This scaling concept allows to fulfill even the highest requirements of performance.

Chapter 2.7 Security

Security on the Notes database level

Job database

Every user who is listed in the job database's ACL can create job documents, that is, use the full functionality of CSLD.

If CSLD is set up correctly, all users have 'Author' rights to the job database, so they can only delete or modify their own job documents. If the CSLD administrator wants to deny certain users the use of CSLD services, he merely needs to remove these users from the job database's ACL. Job documents must be signed in order to be processed by CSLD. Unsigned jobs are rejected with an error message. The signer of the job is considered the job's requester. This mechanism ensures that no user can execute CSLD services under the ID of another user. When CSLD processes a job, the requester's signature is replaced with the signature of the CSLD user. This happens because CSLD modifies job documents.

The job database defines the role [CSLDUsers], to which only the CSLD user and the CSLD administrator are added. When jobs are created correctly, they have a readers field set, containing only the job creator and the role [CSLDUsers].

This ensures that every user can see only his own job documents, while CSLD tasks can see all job documents. See "Chapter 2.2 How to use CSLD services" on page 17 for details on how to create jobs.

Configuration database

If set up correctly, only the CSLD administrator and the CSLD user have access to the configuration database. CSLD tasks run under the CSLD user ID (given via a `notes.ini` file). Users with access to the task executable can start CSLD tasks under their own ID, but then the task has no access to the CSLD configuration database. Further, the task has access to only those databases to which the user has access.

User databases

When documents archived in the Notes native format are retrieved from an archive, CSLD restores all document-level security properties. In this way, a document retrieved from the archive behaves as it did prior to archiving. In the case of attachment archiving or rasterizing, CSLD cannot archive encrypted parts of a document without the encryption key. To enable archiving of encrypted documents, the CSLD user must have the decryption

key. To send the decryption key to the CSLD user, use the **File-->Tools-->User ID-->Encryption-->Mail** feature. The CSLD administrator must then manually detach the key from the CSLD user's incoming mail. Databases are CSLD-enabled only if the CSLD user is listed in the user database's ACL.

Identifying Lotus Notes users to the archive

In CSLD, all operations on an archive are performed via CMCS agents. Such agents are archive clients, and must therefore be logged in with a certain user ID. At the same time, CSLD users log into CSLD user databases using their Notes user IDs.

However, there is basically no relation between the Notes user ID and the Content Manager user ID.

CSLD supports two methods of defining the user ID used to log into the archive:

- using a "fixed" default archive user ID and
- mapping Lotus Notes user IDs to archive user IDs.

These two methods are described in the following sections.

The default archive user ID

The CMCS configuration file archint.ini defines archive IDs, which are logical names for a particular archive container (in CM terms an index class on a particular library server, in OD terms an application group). Every archive ID definition block has the parameters to define an archive server, a container on that server and a userID. This userID is used to log onto the archive server. When archpro.exe is started the first time, you must use the parameter *-serverpasswd* to set the passwords for all users defined in archint.ini. Each such user is logged in during the first request to CMCS (this can take a couple of seconds) and stays logged in for all following requests.

The advantages of this approach are

Usability
Simple setup.

Performance
No additional log-in time is consumed.

Security
If the only user who has access to the archive container used by CSLD is the default CSLD user ID, other users cannot use the archive's native client to access the documents archived by CSLD.

The disadvantage of the default archive user ID approach is that security can be implemented only in a simple manner: All data is accessed within the archive under the same user ID. This means that, in the case of attachment archiving and rasterized archiving, all Notes security features on the document level are lost once a Notes document is archived.

Example:
Suppose that user A archives the attachments of a Notes document to which user B has no access. Once the document is in the archive, user B can retrieve it via CSLD, because B has access to CSLD, and CSLD can access all documents archived under the CSLD user ID.

The CSLD administrator must make sure that users do not have access to the archive containers used by CSLD via the archive's native client.

Note:
Remember that, with CSLD, a user can use CSLD services if he has access to the CSLD job database, and if a database is CSLD-enabled (that is, has code to create CSLD jobs). Thus, when using the default archive user ID approach, the responsibility to ensure that all documents in the archive are protected from illegal access lies completely on the Notes database, i.e. the application side.

However, this principle does not apply to documents archived in the Notes native format: When a Notes document protected by document-level security mechanisms (Readers fields, ECLs or encryption) is retrieved from an archive, the document is completely restored. Suppose a Notes user A has no access to a certain document. When this document is archived by user B under the ID of the default user, all other users (including A) with access to CSLD can retrieve it from the archive. However, after retrieval, the document has the same Notes security settings as before archiving, and can therefore still not be accessed by user A.

Note:
Where the security features supplied by an archive are not sufficient to map a given Notes security to archive security, native archiving should be used. However, remember that documents archived in the Notes native format cannot be viewed with the archive's native viewer.

Mapping Lotus Notes user IDs to archive user IDs

In some environments, the security features of an archiving system are not sufficient to implement a certain security model. When users access the archive not only via CSLD (that is, through Lotus Notes), additional security features might have been implemented on the archive side. For example: In Content Manager , a user security add-on (sometimes called user exit) might be implemented that further restricts archive access on the document level. Such security add-ons are based on the principle that each user logging into the archive must have a different archive user ID. Consequently, security add-ons would not work if CSLD was set up with a default archive user ID. Therefore, CSLD offers a user exit to enable users to implement their desired access restrictions. CSLD provides the user with a C function interface that takes the Notes user ID as an input parameter and returns an archive user ID. The user must provide the logic needed to map a Notes user ID to an archive user ID. The function is implemented in C and compiled into C DLL `CSExit.DLL` that is called by the CMCS agent whenever. data is accessed in the archive. You can switch on the security user exit in the CMCS configuration file `archint.ini` for each archive separately. Simply add the line

```
ACCESS_CTRL ON
```

to any archive ID definition block for which you want to enable the user exit. When the user exit is enabled, any request to update an archived document's index information, delete archived documents, or retrieve archived documents (by single retrieval or by query) is performed on behalf of the user ID provided by the user exit.

Important:
Using the user exit instead of a default archive user ID will slow down the overall performance of the above functions significantly, since CMCS must log off and log in again when user IDs are switched. However, CMCS always tries to find an agent that is currently logged in under the user ID making the request.

Restricting access to archived documents on a per-user/database level

In most cases, it will make sense to have different users archive documents to the same archive container. For example, in Content Manager, access rights are granted on the index class level. Thus, if a user can read one document in an index class, he can read all documents from that index class. However, in some cases, security on the index class level is not fine-grained enough to ensure document security.

Considering the cases of e-mail archiving where users want to be sure that their archived e-mails are not read by other users. In this scenario, there must

be one index class per user - which is hard to handle in larger companies. Therefore, it would be advantageous to have the ability to store documents of different users in a single index class or application group and still ensure that users can access only their own documents. For this scenario, CSLD provides two security modes:

- the *retrieve documents by original user only* mode and
- the *retrieve documents to original database only* mode.

These two modes are described in the following sections.

The ″retrieve documents by original user only″ mode

When the ″retrieve documents by original user only″ mode is turned on, documents can be retrieved only by the user who archived them. When an attempt is made to retrieve another user's document, an error message is returned.

Important:
This mode requires that all archive containers that are used to archive documents in a ″secure way″ define an attribute named *CSLDOrigUser*. When a document is archived in this mode, the user's Notes ID name (in canonical format) is stored in this attribute by CSLD. *CSLDOrigUser* must be of the type 'variable character'. The minimum length must be longer than the longest user name in canonical Notes format. We recommend configuring at least 100 characters. Since this feature is optional, we recommend defining this attribute as **optional**. To enforce retrieval by the original user only, define *CSLDOrigUser* as a required attribute. However, you should take care in designing your archiving strategy: Once you have defined this attribute as required and the index class contains data, you will not be able to turn this mode **OFF**.

The ″retrieve documents to original database only″ mode

When the ″retrieve documents to original database only″ mode is turned on, documents can be retrieved only to the database from which they were archived. If an attempt is made to retrieve a document to a different target database, an error message is returned.

Important:
This mode requires that all archive containers that are used to archive documents in a "secure way" define an attribute named *CSLDOrigDB*. In this mode, when a document is archived, the database in which the document resides (server and path) is stored in this attribute by CSLD, the replicaID of the database in which the document resides. *CSLDOrigDB* must be of the type 'variable character'. Configure this attribute to have a length of 17 characters. Since this feature is optional, we recommend defining this attribute as **optional**. To enforce retrieval to the original database only, define *CSLDOrigDB* as a required attribute. However, you should take care in designing your archiving strategy: Once you have defined this attribute as required and the index class contains data, you will not be able to switch this mode **OFF**.

Things you need to know about both security modes

- *For both modes, CSLD must log into the archive using the default user ID.* Do not use the user exit. Further, it is essential that only the default user ID and the Content Manager administrator have access to the "secure" index classes or application groups, because otherwise users could use the Content Manager native client to find and view other users' documents.
- You need not add a document mapping or special mapping entry for the attribute *CSLDOrigUser* or *CSLDOrigDB* in the configuration database. The mapping is done internally by CSLD.
- When a document is archived with the security modes turned **OFF**, neither *CSLDOrigUser* nor *CSLDOrigDB* will be filled in, even if these attributes exist in the index class or application group definition.
- When a document is retrieved with the security modes turned **OFF**, *CSLDOrigUser* and *CSLDOrigDB* are ignored.
- When a document is retrieved with either mode turned **ON**, no Notes items will be added to the document with the content of the attribute *CSLDOrigDB* or *CSLDOrigUser*.
- The attribute names *CSLDOrigDB* and *CSLDOrigUser* are reserved.
- The modes are turned **ON** and **OFF** via the profile configuration dialog in the configuration database. See "Defining profile documents" on page 125 for details.
- There is no direct connection between the two modes. Either one can be turned **ON** or **OFF** independently from the other.
- When the "retrieve documents to original database only" mode is turned **OFF**, no checking of the document's ownership is performed. That is, users can basically retrieve other users' documents, but only to the original database. This is not a security loophole: if a user did not have access to the

original database at archiving time, he will most probably still have no access to it at retrieval time. Therefore, he can retrieve it, but not read it. The only harm which very sophisticated users with a profound knowledge of LotusScript, the CSLD job format, and other users' environments can do is by filling up another user's database with retrieved data owned by the other user, but they cannot read it. For maximum security, turn both modes **ON**.

- The two security modes cannot be applied to existing index classes or application groups that have not been filled by CSLD because they do not define the attributes *CSLDOrigUser* or *CSLDOrigDB*.

The separation of document ownership and original database allows CSLD to apply to scenarios like the "executive secretary problem": A manager has given his secretary access to his mail database (by adding her to the database ACL), from which he has archived documents. While he is in the office, he has both modes turned on to ensure that no one else can retrieve his documents. However, while he is on vacation, the secretary might need to retrieve important documents from the archive. Therefore, the "restore documents by original user only" mode is temporarily turned **OFF**. Then, while the manager is on vacation, the secretary can retrieve documents archived by her manager, but only to the manager's mail database.

Part 3. CSLD Features

Chapter 3.1 Archiving methods

CSLD supports the following three different archiving methods:

- *attachment archiving*,
- *archiving in the Notes native format*, and
- long-term archiving by *rasterizing* documents.

These three archiving methods are described in the following sections.

Attachment archiving

Attachments can take up a considerable amount of your database space. While typical Notes documents have sizes of only about 50 KB, in the case of such attachments as bitmaps or executable files, the size of a single document can easily expand to several megabytes. CSLD allows you to free space by offloading attachments to an archive and retrieving them only when needed. You can archive attachments from a single document, from a number of selected documents (even from the entire database), or from an entire view or folder.

With CSLD, you can choose between having archived attachments actually removed from the container documents or having them archived for backup purposes, only (in which case the attachments remain in the document). When archived attachments are removed from the container document, the attachment is replaced with a URL link stating when and by whom the attachment has been archived. When the user clicks on that link a browser will open and CommonStore will display the content directly from the archive. For details on browsing archived content please refer to “Chapter 3.10 Browser viewing of archived content” on page 69. This message is removed when attachments are restored back to a container document.

CSLD can archive encrypted attachments only if the CSLD user has the decryption key. However, the complete document (including encrypted attachments) can be archived in the Notes native format. When restored from the archive, users still need the decryption key in order to open an encrypted attachment.

Attachments of Notes release 3 and higher reside in an RTF field. CSLD also supports attachments created with a Notes version prior to release 3 that reside in a Notes item and not in an RTF field. In the Notes Client, such attachments are usually displayed at the bottom of a document.

Typical examples for attachment archiving are

- user mail databases that often have a disk quota (size limitation) on them.
- a production database where incoming documents have scanned order forms as attachments. A Notes Agent automatically finds new documents and creates backups of the order forms by archiving them.

An attachment archived by CSLD appears in the archive like any other document. That is, you can use the archive's native client application to find or display the attachment. For every attachment in a Notes document, one document (or *item* in Content Manager terms) is created in the archive. All of the attachments from the same document are archived with the same descriptive information.

CSLD always archives *all* of the attachments in a document, not just a subset of those attachments.

Archiving in the Notes native format

Up to Notes/Domino 4.6, the maximum Notes database size is still 4 GB. When databases grow quickly, a mechanism is needed to temporarily offload Notes documents to an archive and restore them only when needed. CSLD makes it possible to send a complete Notes document as a *native bytestream* to an archive, and to restore it back to a database to its original state so that it can be processed in the Notes client like any other document. *Encryption as well as the document ACL (Access Control List) are preserved.*

Note:
The bytestream representation of archived documents depends upon the platform and the given Notes version. There is no guarantee that Lotus will not change the internal native document format in the future. In other words: It might be *impossible* to restore documents archived with an older Notes version to a later Notes version. For this reason, native archiving should *not* be used for long-term archiving.

A possible drawback of documents archived in the native format is that they cannot be viewed using the archive's native client.

Long-term archiving by rasterizing documents

When thinking of long-term archiving, i.e. archiving times of 10 or more years the format of the archived document becomes essential. To ensure that archived documents are still readable after residing in the archive for years an application-independent format is needed. This way users do not have to care about preserving the applications to view archived content but can focus on the content itself.

In other cases, e.g. for legal archives a requirement might be given that a document cannot be edited once it has been archived.

Both requirements can be achieved by means of rasterizing, i.e. converting Notes documents to an application-independent, unmodifiable image format.

CSLD makes it possible to rasterize Notes documents to Microsoft Rich Text Format (RTF) and plain ASCII format. Other formats (e.g. TIFF) can be implemented through a user exit, that CSLD provides, the so-called Raster Exit. Using this exit, external products that allow converting Notes documents to other formats can be plugged in.

Note

CSLD uses the export filters provided by Notes when converting documents to ASCII or Rich Text Format. Since these export filters may or may not be available and working correctly, CSLD cannot guarantee this functionality.

Microsoft Rich Text Format

The Notes document as seen on the screen is converted into an RTF image, including all formatting as well as text colors and highlighting information. Embedded images are displayed in black and white. Attachments are not converted. The RTF file can be imported into any standard word processing tool.

Plain ASCII format

The text parts of a document are written to a plain ASCII file. All formatting or images are lost. Only blanks are preserved. However, the size in bytes of the resulting document will be much smaller compared to that of the document in any other archiving format.

This format is used when users are interested only in the actual written content of a document, or when the size of archived documents is an issue.

Other raster formats using the Raster Exit

When another format is desired, third-party providers can plug-in

their functionality through this user exit. CSLD passes the Notes document to be converted, the options for the converted document and a file name under which the result of the conversion is expected.

CSLD supports the following raster options:

Rasterize Notes document and embed the attachments
The Notes document is rasterized including its attachments. The attachments are rasterized at the position they were attached to.

Rasterize Notes document and append attachments
The Notes document is rasterized including its attachments. The attachments are rasterized and appended following the note content.

Rasterize attachments only
The Notes document itself is not rasterized, only its attachments. All attachments are rasterized into a single file.

In addition, CSLD passes some format options, telling the user exit to rotate images that are wider than high and to trim leading and trailing white spaces.

Note
These options are passed as request types to the Raster Exit. However, the responsibility to fulfill these requests lies with the third-party provider who implements the exit.

The rasterized document (image) is archived with index information taken from the Notes documents; thus, it can be found in the archive by means of queries. Rasterized documents appear in the archive like any other document. That is, you can use the archive's native client application to find or display them. When rasterized documents are retrieved from an archive, they are attached to a target container document as a regular Notes document attachment, where they can be viewed with the Notes internal viewer. The target document is selected by the CSLD application programmer.

See also "Chapter 7.5 The Raster Exit DLL" on page 145.

Chapter 3.2 Archiving options

Selecting documents for archiving

Notes documents are selected by passing their Universal Notes Identifiers (UNIDs) to CSLD. You can pass either a single UNID or a number of documents from the same database to CSLD.

It is also possible to select a view or folder for archiving. See “Archiving folders and views” for details.

Deleting documents after successful archiving

When documents have been archived (regardless of the archiving method used), you have the option of deleting the original document. This feature is used when the original document is no longer integrated into a workflow and can therefore be deleted after archiving.

Note:
In the case of attachment archiving, however, CSLD application developers must take care: When a document without any attachments is selected for attachment archiving, there is nothing to archive. CSLD considers this document as ″having been successfully archived.″ Consequently, when you choose to delete the document after successful archiving, the document is *lost*. This option makes sense in a scenario in which you want to archive only the attachments of a selected view or folder, and you want to get rid of the container documents.

Archiving folders and views

With CSLD, you can archive all documents of a given view or folder. Folders and views are dynamic objects, i.e. users can add new documents to the view or folder even during the document archiving process. When CSLD begins archiving, it takes a ’snapshot’ of the current content of the view/folder, and documents added to the view/folder after archiving has begun are not archived. That is, you cannot continuously create new documents and have them archived by adding them to a folder/view which is being archived.

When a folder is archived, all documents included in its subfolders are archived, too. The folder or view is selected for archiving by passing its UNID to CSLD. Selection is not done by view/folder name.

Optionally, CSLD preserves the folder structure in the archive. This way, starting from a given root folder, the complete folder/subfolder structure can be moved to the archive. When a complete Notes folder has been archived preserving its structure, it can be restored as that very same structure back to Notes.

Document rearchiving

CSLD does not restrict documents to be archived only once. It is up to the logic of a CSLD enabled database whether a user can archive a document twice or more. For example, the mail archiving demo database does not allow to archive a document twice. Whether a document has already been archived can be determined by checking the *CSNDArchiveID* item. If it does not contain the word "Error", the document has been archived. If it does not exist, the document has not been archived, or has been deleted from the archive.

For long term archiving, e.g. for legal requirements (attachment archiving or converting a document to TIFF, RTF or ASCII format), a document is usually archived only once. The CSLD administrator may also want to prevent users from archiving duplicates of a document. There are many ways how an application can prevent rearchiving. For example, it could hide an archiving button for archived documents. Or in a view, an error message could be displayed if a user tries to select an archived document and click the "archive" button. For agent-based archiving, the agent could simply ignore all archived documents.

However, you may want to archive different versions of a document you are working on. In this case, you should carefully design your application logic, and you should have clearly understood how CSLD writes state and document IDs to archived Notes documents. For archiving state, see "Defining profile documents" on page 125.

Suppose you archive the attachments of a document, and you let CSLD remove them (via the delete flag in the job). Further suppose you want to continue working on this "container document", e.g. add some text and attach further documents. If the *write state to special field* mode is enabled in the archiving profile, it is now possible to archive the new attachments in the container document. CSLD will then attach the new document IDs to the existing IDs in the *CSNDArchiveID* item. When the attachments are finally retrieved back to the container document, CSLD will retrieve all attachments that have ever been archived from that document. Note, however, that you can only rearchive documents when the request type is attachment archiving. You cannot mix different formats, i.e., CSLD does not permit to archive the attachments of a document, and rearchive the document in native format.

Important

The intention for CSLD was to provide an archiving tool, not a document management tool. The attachment rearchiving feature is not a true version management tool. CSLD will not restore a certain version of a Notes document, nor does CSLD keep any information about the different versions of a Notes document. Via the *CSNDArchiveID* item, the Notes document always links to all attachments in the archive that have ever been archived from that document.

Caution

If the *Write state to CSNDArchiveID field (CSLD 2.1 behavior)* mode is selected in the profile document, and you rearchive a document in whatever format, all existing entries in the *CSNDArchiveID* item will be overwritten with the new document IDs.

Chapter 3.3 Retrieval methods

When a Notes document is archived, it is assigned an archive ID to uniquely identify it within the archive. When the original document is not deleted after successful archiving, CSLD stores all archive IDs in a special Notes item named **CSNDArchiveID**. In the case of archiving in native and rasterized format, only one document is created in the archive; thus, this field will contain only one entry. In the case of attachment archiving, the number of entries in the **CSNDArchiveID** field is equal to the number of archived attachments. Optionally, in case of attachment archiving all archive document IDs can be accumulated. For details see "Document rearchiving" on page 48

In CSLD, there are five ways to retrieve documents from the archive:

- **Retrieving a single document by id:** If the ID of the archived document is known (i.e. taken from the CSNDArchiveID field of an archived Notes document or from a hitlist), the document can be retrieved.
- **Archive search:** A search in the archive returns all documents that match a certain search criteria.
- **Listing the documents in a workbasket:** All documents in a workbasket are returned as a hitlist or multiple result documents.
- **Listing the content of a Content Manager folder:** Besides regular documents, a search can also return folders. Clicking the "Open" button in a hitlist will return the folder content in a second hitlist (or as multiple result documents).
- **Restoring a Notes folder or folder structure:** Archived Notes folder structures are restored to their original position, together with the documents in the folder structure.

These methods will be explained in the following sections.

Retrieval by archive ID

A number of archive IDs of archived documents is passed to CSLD, which then finds the documents in the archive and writes them to a target location chosen by the user.

- In the case of *attachment archiving*, a retrieved attachment is written to an RTF field in a target document chosen by the user.

  When the user does not explicitly specify an RTF field in a target document, CSLD creates the attachment in a separate item at the bottom of the document. If no target document at all is specified, CSLD will create a new result document as a container for the attachment. For example: In a

mail database, users would click on a **Retrieve attachments** button to retrieve all archived attachments of the selected documents and write them back to their original container document.

- In the case of *archiving in the Notes native format*, retrieved documents are written to a target database chosen by the user. When the original document, i.e. the document the native content was created from, still resides in the database, that document will exist twice after restoring the natively archived document. If the user specifies a target document, the retrieved native document will replace this document, i.e. upon successful restoration, it will have that document's UNID. Usually, native documents are not retrieved on the basis of their archive ID, but rather by means of queries.
- *Rasterized documents* retrieved from an archive are written as an attachment to an RTF field in a target document that can be chosen by the user. When the user does not explicitly specify an RTF field in a target document, CSLD creates the attachment in a separate item at the bottom of the document. If no target document at all is specified, CSLD will create a new result document as a container for the attachment. See "Result documents" on page 166 for a description of result documents.

In addition, for all three archiving methods, retrieved documents can also be written to a target folder chosen by the user.

The archive IDs can be taken from the **CSNDArchiveID** field.

Retrieval by query

Regardless of the archiving method, every Notes document is archived with descriptive information taken from document fields. This descriptive information is used to search the archive for certain documents.

When the original document is deleted after successful archiving, the **CSNDArchiveID** field and consequently all links to documents in the archive are lost. Thus, the only way to find an archived document whose original has been deleted is by means of *queries*.

Queries are performed by filling in search parameters into a query form. For every Notes form for which you defined a document mapping in the configuration database, you should create a corresponding query form. Simple query forms for a given Notes form can be created using the CSLD setupDB tool. The query form contains one search parameter field for every field listed in the form's document mapping (see "Document mappings" on page 19 for details).

Since a filled-out query form is a regular Notes document, you can save frequently-used queries for future use, for example in a special queries folder.

CSLD supports two methods for displaying a query result:

- by means of a *single hitlist documents* or
- by means of *multiple result documents*.

These methods for displaying a query result are described in the following two sections.

Displaying a query result by means of a single hitlist document

In the case of hitlist documents, all hits of a query are displayed in a table with descriptive information. Only one hitlist document is created per query.

Every hit is represented as a single row in the table. Single documents in the hitlist document can be retrieved by simply clicking the **Fetch** button. You can also retrieve all documents in a hitlist by clicking the **Fetch All** button. Queries returning no hits will result in an empty hitlist.

It is up to the application logic to decide what to do with hitlist documents. Usually, once they are no longer needed, they are deleted or written to a special folder.

Displaying a query result by means of multiple result documents

The other method of displaying a query result is to create a simple result document for each hit. The result documents consist only of a number of fields displaying the document's index information in the archive. The document content itself remains in the archive. It can be retrieved in a second step, e.g. by defining an action in the view. Using multiple result documents allows a query result to be displayed in a customized view. The downside is that many documents might be created that must be deleted afterwards.

Each query is bound by two parameters:

The number of directly-built documents (maxNumOfDirectHits)
: If a query returns a number of documents less than or equal to this parameter, no hitlist is generated. Instead, all hit documents are directly retrieved from the archive and written to a target location. Otherwise, a hitlist is created.

The maximum number of hits (maxNumOfHits)
: This parameter specifies the maximum number of hits returned by the query.

Every hitlist document or retrieved document contains two fields:

- the name of the person who issued the query (i.e. the **requester**) and
- a timestamp stating when the query was started (**reqTS**).

In databases with many users, this information can be used to associate users with their query result documents. For example, a view can be generated and categorized by user name and/or query timestamp.

Every hitlist document has an item named **CSNDArchiveID** containing the archive IDs for all hits in the list.

Chapter 3.4 Agents for processing archived and retrieved documents

CSLD supports the automatic invocation of custom Notes agents for processing archived and retrieved documents. These agents can be LotusScript, formula language, Java or JavaScript agents, and allow to apply custom logic to your CSLD enabled application

- **Pre-archiving agent**s are invoked before a document is archived. Typically, they prepare a document for archiving
- **Post-archiving agents** are invoked after a document has been archived. Use these agents to perform cleanup, set state fields, trigger workflows, set access rights on a document, create a stub from a document (see below), etc.
- **Post retrieval agents** are invoked after a document has been retrieved. For example, these agents can be used to set access rights on retrieved documents.

All these agents are invoked on a single document level, even when CSLD processes jobs for multiple documents. That is, when ten documents are archived, the post-archiving agent will be invoked ten times. The document is passed via the document context of the Notes session. For example, in LotusScript, you would access the document via the *DocumentContext* property of class *NotesSession*

Pre and post archiving/retrieval agents are associated with a certain Notes form. This allows to invoke different agents depending on the document type that is archived or retrieved. Set the names of your agents in the document mapping dialog of the configuration database. Remember that archiving of documents can fail for several reasons. Therefore, a pre-archiving document should never run code that assumes that the document is archived successfully. For example, the code should not remove content from the document.

Important

Failure of pre or post archiving/retrieval agents will not result in a job with state error, because these agents are part of the application logic, not CSLD. However, in case of failing agents, the agent log is written to the task's trace file, sent to the CSLD admin as email (if an admin is configured in the profile), and written to the console. We recommend to test such agents manually for correctness before they are invoked with CSLD.

Using post-archiving agents to create document stubs

Instead of deleting a document after archiving, you can use a post-archiving agent to strip it down to small size. We recommend to leave a few descriptive fields, and delete all other space-consuming fields of the document. For example, for email archiving, you could leave items *Subject*, *From*, *PostedDate* and *CSNDArchiveID*, and remove the *Body* field and all *$FILE* fields. The remaining stripped-down document is called a *stub*. With help of the descriptive fields, the Notes fulltext index can then be used to find the document stub in a database. Since the stub still contains the *CSNDArchiveID* item, the full document can be retrieved

The *Create Stub from Native Document* agent in the email archiving demo application demonstrates how such an agent could be written in LotusScript.

Chapter 3.5 Updating archived documents

When Notes documents are archived via CSLD, each document in the archive is stored with a number of attributes (also called index fields) containing descriptive information extracted from the document. Each attribute value is the value of a field (item) in the Notes document (see "2.3.1 Mapping Notes forms to archives" on page 19 for details).

CSLD supports updating index fields. However, CSLD does not support updating the document content.

Example:
Suppose you have archived a document attachment and that the attachment is archived with two index fields containing a person's name and address. When the name and the attachment is modified in the original Notes document and the document is updated with CSLD, the attachment in the archive is not replaced with the modified version. Instead, only the name index field is updated.

Important!
CSLD does not support updating documents archived in the Notes native format. The reason is simple: Index fields values are extracted from Notes document fields. When index fields are updated but the archived document is not, the index and document content no longer match.

Example:
Consider the scenario in which a customer record document having a certain address field is archived in Notes native format. Now the address index field is updated because the customer changed his address. Suppose an address query returns the updated document. The document will then still contain the old address.

Updates are performed by passing the UNID of the document containing the index fields to be updated and the document archive ID to CSLD via an update job. In CSLD-based applications, there are two types of documents that can be updated:

The original Notes document

Every Notes document that has been archived maintains the **CSNDArchiveID** item, which contains a list of archive IDs pointing to the archived content of the document. In the case of *attachment archiving*, there is one archive ID for every attachment.

When the original Notes document is updated, CSLD first extracts new index values from it. CSLD then goes through the list of archive IDs and updates the index fields of every archived content with the new index values.

Result documents

When a rasterized document or attachment is retrieved from an archive, users have the option of creating a result document containing the retrieved content as an attachment. Besides the actual content, a result document contains the index fields of the retrieved document. After modifying the index fields, a result document can be updated. CSLD extracts the archive ID from the **CSNDArchiveID** item of the result document and updates the corresponding archive document with the modified index values.

Chapter 3.6 Deleting archived documents

CSLD supports deletion of archived documents. To delete a document in the archive, the document's archive ID must be passed to CSLD via a CSLD delete job. There are various methods of retrieving an archive ID:

- Every Notes document that has been archived maintains the **CSNDArchiveID** item containing the archive IDs of archived content.
- The **CSNDArchiveID** item in a result document points to the content in the archive that is attached to the result document.
- In a query hitlist document, the **CSNDArchiveID** item contains the archive IDs of all hits in the list.

3.6.1 Consistency of Notes documents and the archive

To keep the Notes user database consistent with the archive, users should have CSLD remove the **CSNDArchiveID** item from the archived Notes document. This can be done by passing the UNID of the document containing the reference to CSLD via a CSLD delete job. If the **CSNDArchiveID** is not removed, it will point to content that no longer exists in the archive. This will lead to error messages when users try to retrieve, delete, or update the content belonging to an archived Notes document.

Note:
CSLD assumes that documents archived with CSLD are deleted only from within CSLD applications. That is, if an entry in a document's **CSNDArchiveID** item points to archived content, and this content is deleted from the archive via the archive's native client or any other mechanism, CSLD will not automatically synchronize its Notes databases to the archive content. It is the Content Manager/CSLD application developer's responsibility to ensure consistency between the archive and CSLD.

Chapter 3.7 Workbasket support

In IBM Content Manager, workflow is implemented via workbaskets. A workbasket is basically a named container for documents. Workbaskets typically contain documents in a certain state of a workflow. A Content Manager document can not be member of two workbaskets at the same time.

If Content Manager is used as an archive for CSLD, the workbasket concept can be leveraged to be used from within Lotus Notes. With CSLD workbasket support, the following scenario would be possible: A customer sends an order via fax. Via any Notes-enabled standard fax software, the incoming fax is converted to a Lotus Notes document with a TIFF attachment. Every incoming fax is archived automatically into workbasket ″Incoming″. User A retrieves all documents from the ″Incoming″ workbasket, and processes them within Notes. Then, he moves the document to workbasket ″Done″, which in turn triggers an acknowledgment mail to the customer.

CSLD workbasket support includes the following functionality

Archiving documents into a workbasket:
When a document is archived, it can be added to a workbasket with a given name. Archiving and adding to the workbasket appears to the user as a single (atomic) operation. The mail archiving demo application (*function archiveSelectedDocuments()*) shows an example of how this feature can be applied In the archiving dialog, a user can enter the workbasket name. However, an application could also implicitly set the name of a workbasket based on the value of a document field.

Moving documents to a workbasket:
Documents that have already been archived can be moved to a workbasket later. You can not move a non-archived document to a workbasket. If the document is already in a workbasket, it is moved from the current workbasket to the target workbasket. The mail archiving demo application (function *moveSelectedDocumentsToWorkbasket()*) demonstrates how this feature can be applied. Here, a dialog pops up asking the user for the target workbasket.

Removing documents from a workbasket:
Removes documents from their current workbasket. If the documents have been archived, but are not member of a workbasket, the request

is ignored. In the mail archiving demo, function *removeSelectedDocumentsFromWorkbasket()* demonstrates how this feature can be applied.

Listing documents in a workbasket:
Returns all the documents in a given workbasket as a hitlist. As a Content Manager workbasket can contain documents from different index classes, the documents in the hitlist can be of different type. To actually retrieve the documents, simply click the ″Fetch″ or ″Fetch All″ button. In the mail archiving demo, function *listWorkbasket()* demonstrates how this feature can be applied. Here, a dialog pops up, asking the user for the workbasket name to list.

For OnDemand being used as the archive, workbasket support can only be ″emulated″ by CSLD: Whenever a document is added or moved to a workbasket, an index field with the name *workbasketName* is set to the name of the workbasket. That is, membership to a workbasket is identical to having the *workbasketName* attribute set to some value. While Content Manager will ensure that a document cannot be added to a workbasket with a wrong name, OnDemand does not know which ″virtual″ workbaskets exist. It is up to the application logic to set workbasket names correctly.

Chapter 3.8 Archiving of Notes folder structures

In Lotus Notes documents can be organized in folders, e.g. to collect all memos concerning one customer or project. An effective archiving solution should be capable of preserving that organizational structure in the archive so at a later time the entire structure can be restored back to Notes.

In contrast to the option to archive all documents within a certain folder or view, this archiving option will preserve the folder structure in the archive. Starting with a given root folder, CSLD will archive the folder itself, all documents within the folder and also all subfolders and the documents and subfolders within these. After successful archiving, the complete folder structure can be deleted from Notes. However, note that CSLD will only delete the documents after they have successfully been archived, but not the folders.

To enable CSLD to archive entire Notes folder structures, an additional archive container has to be created first. CSLD will use this archive container internally to store all information about the Notes folder and to handle the documents residing in it. For information on how this special archive container is set up and what precautions must be taken, please refer to “Preparing Content Manager and OnDemand for Notes folder archiving:” on page 119.

Restoring archived notes folder structures

Unlike folders in a Content Manager archive, Notes folders are only identified by a name, i.e. they do not have their own set of attributes. For a single Notes database, a folder name is sufficient to identify that folder. But for different Notes databases there might be other folders with the same name. To prevent those folders from being mixed up in the archive, CSLD stores along with the name and possible alias of a Notes folder also its originating database and the name of the user that archived it.

When a folder is to be restored to Notes it can be done based on the folder’s name or based on the archive ID CSLD assigned to the folder when it was archived. In any case, restoration of a Notes folder is limited to the database it originally came from.

Although Notes folders look like they are hierarchical, in fact their structure is flat. Creating a subfolder does not really create a folder within the root folder, but instead will create a folder with name ″Root Folder\Subfolder″. Thus,

when retrieving folders by name, subfolders within a complex (looking) folder structure can be retrieved by specifying the above naming scheme ″Folder\Subfolder\SubSubFolder...″.

Restoring a Notes folder will (re-)create the original folder including all columns, actions, etc., defined before archiving and then restore the archived documents back into the original hierarchy. For example, suppose folder ″Customers″ containing several notes was archived in TIFF format, preserving its structure. Restoring this folder will either recreate that folder in the Notes database or use any existing ″Customer″ folder still residing in the database. CSLD will then restore all documents back to that folder. Since the format in which the documents from this folder were archived was TIFF format, CSLD will create result documents containing the rastered originals as attachments. If the original note is to be preserved then only native archiving will work.

Rearchiving of Notes folders

Notes folders may only be archived once. When a folder is archived, CSLD will write back the archive ID to the original folder. When that same folder is archived again, the folder itself will not be archived a second time. Only the documents residing in that folder will be archived according to the following rules:

- Documents that have never been archived (i.e. documents containing no archive ID) will be archived in the requested archive format and added to the already archived folder.
- Documents that have been archived before (i.e. documents containing an archive ID) will be archived again, if the archive format (request type) of this archiving request differs from the previous one. In this case, a second archive document in the requested format will be created and added to the already archived folder.
- Documents that have been archived before in the same archive format will not be archived again. In this case the archived document will only be moved to the already archived folder.

If another subfolder is added to an already archived folder structure, only the new subfolder will be archived and added to the already archived folder. All subfolders that were already archived in an earlier step will not be archived again.

Important

Archiving entire Notes folder structures is complex. Since the only way to identify a Notes folder in the archive is by its name and originating database, extreme caution must be applied when using this feature. Suppose Notes folder ″Customers″ has been archived and the original has been deleted from Notes. Later on, a second folder ″Customers″ is created. When this folder is archived, due to restrictions of the underlying archive there is no way for CSLD to prevent this folder from being archived as well. The result is two folders with the exact same descriptive attributes stored in the archive. Thus, retrieving folder ″Customers″ by name will not work anymore.

Chapter 3.9 Browsing of archive folder structures (Content Manager only)

In existing paper archives there are usually a variety of different document types that need to be archived. For example in a Human Resources department there might be index classes for applications to the company, sick reports about all employees as well as their resumes, etc. However, since a set of documents always logically belongs to a certain employee, this data could be organized in folders. The folder itself could be described with the employee's personal data (like Name, Address, Birthday, etc.). All documents concerning that employee would then virtually be filed into that folder. Thus each employee folder would contain references e.g. to one application form, several sick reports for the past years and an annual resume about that employee. So when the Human Resources department seeks information about a certain employee they would only have to search the Employee folders, from there browsing everything filed for that employee.

Since CSLD can be used as an archive frontend to existing archives, these folder structures are also reflected back to Notes. Therefore, CSLD allows for searching and retrieving of archive folders as well as archived documents.

Note
Archive folder structures are only considered for browsing purposes, while it is not possible to archive certain documents from Notes to specific archive folders.

In Content Manager both folders and documents are treated completely alike. The only difference between a folder and a document is that a document describes the actual archived content whereas a folder has no content but instead contains references to documents or other folders. Thus, searching the archive using CSLD will just as transparently return both, folders and/or documents. When issuing a request to retrieve the content for an archived document CSLD will restore that content, while a request to retrieve the content of an archived folder will return all documents and/or (sub-)folders contained in the respective folder.

This way, users can navigate along the folder/subfolder structure by retrieving the subfolders level by level, thus moving downwards in the hierarchy.

Note
Even though it is possible to retrieve documents from the archive to a certain folder in Notes, CSLD will not create Notes folders and subfolders according to the structure in the archive.

Note
Archive folder structures can only be retrieved to Notes one level at a time.

The representation of archived documents and archived folders depends on which retrieve configuration was chosen in the retrieve profile.

Hitlist representation

A hitlist contains a single row for each hit returned by the archive search. In addition to the attributes that describe the returned documents or folders each row contains two additional columns. One column indicates whether the respective row is an archive folder or a document, and the other contains a button to create a retrieve request for the respective hit. If the hit refers to a document this button will be labeled ″Fetch″, if it refers to a folder it will be labeled ″Open″.

Creating a retrieve request for a document will return a result document that contains that hit's content as an attachment. Opening a folder creates a second hitlist, consisting of all documents and subfolders contained in that folder.

Single result document representation

CSLD result documents consist of all attributes describing a single search hit and optionally the content itself appended as an attachment. One result document will be created for each hit returned by an archive search. In addition to the attributes, CSLD result documents also contain a hidden field flagging the document as representing either a document or an archive folder. This field can be used to hold documents and archive folders apart in a database view.

Creating a retrieve request for a document will return the document content for that hit, appending it to the result document as an attachment. Retrieving back a folder will result in a single result document for every document and subfolder contained in that folder.

Chapter 3.10 Browser viewing of archived content

Instead of restoring archived content back to the Notes database by creating an attachment in a Notes document, CSLD allows to view archived content directly in the archive by displaying it in a browser.

> **Note**
> Viewing content in a browser requires an appropriate plug-in to be installed. Otherwise a dialog will pop up asking the user to download the file.

There are two options to view archived content in a browser:

Browsing of archived attachments

When an attachment has been archived and removed from its container document, CSLD will create a URL link hotspot in the document at the position from which this attachment has been removed. Clicking on this link will open up the browser and display the content without first restoring the attachment to the document.

Browsing of search results

When users perform archive searches, CSLD either returns a set of result documents, each representing a single hit, or a single hitlist document containing a row for each found document.

Both search results include a URL link to allow users to view the search hit in a browser without first restoring the content to Notes.

> **Note**
> CSLD will create view links only for search results describing a non-natively archived document, since the Notes native format cannot be viewed in a browser. However, this cannot be done when upgrading from a previous version of CSLD, because in version 2 of CSLD the additional information to identify a natively archived document without restoring it was not available.

Configuring browser viewing

To enable CMCS to invoke a browser the data type of the content needs to be associated with a so called MIME type. The browser uses this MIME type to find out which application will open the content.

Thus, when configuring browser viewing, two steps have to be completed:

1. Create MIME types for archived content types.

   A file named `csmimes.properties` can be found in the CommonStore subdirectory of the CSLD installation. Edit this file to associate CSLD content types to browser MIME types. A set of common mappings is already defined in this file. Follow this example to create new mappings. If a certain document type has no respective MIME type defined, CMCS defaults the content to **text/plain**.

2. Configure MIME types in browser configuration

   When opening the file types tag of the browser properties, the plugin or application that the browser should use to display a certain MIME type can be configured. Make sure that the MIME types entered in the CommonStore `csmimes.properties` file find a corresponding entry here. If no plugin or application is found to display a certain MIME type, the browser will prompt the user to store the content in a file instead.

Part 4. CSLD Installation

Chapter 4.1 Prerequisites

CSLD supports a wide variety of hardware platforms from which customers are free to choose based on the requirements of the given application scenario. While CSLD itself runs on AIX and Windows NT, the main archiving load can be distributed among every system supported by Tivoli Storage Manager, CMOD, or Content Manager, ranging from an NT-based PC to an S/390 mainframe computer. Through the flexible hardware support offered by Tivoli Storage Manager, CSLD also allows the storage of documents on quite arbitrary storage media, including hard disks, tape drives, optical media, or corresponding library systems.

Archive server

Depending on which archive you want to use with CSLD the following prerequisites must be fulfilled on the archive server.

Archive	Archive Server Platform(s)	Prerequisites on archive server
Tivoli Storage Manager	NT, AIX, HP-UX, Sun Solaris, OS/390	installed, tested, and operational ADSM 3.0 (or later) or Tivoli Storage Manager 3.7 (or later). ADSM/Tivoli Storage Manager is configured with defined management classes and nodes — which should be used for archiving (see “Chapter 5.1 Base setup” on page 95)
Content Manager	NT, AIX, OS/390	Installed, tested, and operational complete Content Manager System 2.4 (or later). Configured with defined index classes, workbaskets, and Content Manager users — which should be used for archiving (see “Chapter 5.1 Base setup” on page 95)
Content Manager OnDemand	NT, AIX, HP-UX, Sun Solaris, OS/390	installed, tested, and operational, complete Content Manager OnDemand 2.2.1.8 (or later). Configured for CSLD as described in “Chapter 5.1 Base setup” on page 95.

Note:
ADSM/Tivoli Storage Manager on HP-UX does not support optical devices.

CSLD server

You must prepare the environment before you can begin installing CSLD. You must therefore ensure that the appropriate software is already installed.

Operating system and additional software

Platform	Installation Prerequisites
NT	— installed Windows NT 4.0 SP5
AIX	— AIX 4.3 or higher
all	— The latest Java runtime environment or Java development kit 1.1 must be installed. For NT and AIX, it can be downloaded at `http:/www.ibm.com/java/jdk/download`

Lotus Domino software

Platform	Domino software
NT	— Notes R5 installed
AIX	— Notes 4.6 or later installed

Archive client software

Depending on the archive system to which you want CSLD to archive your data you can find the necessary prerequisites for your CSLD server machine in the table below.

Archive	CSLD Platform(s)	Prerequisites
Tivoli Storage Manager	NT, AIX	— installed ADSM V3.7 or Tivoli Storage Manager client (API).
	all	— The Tivoli Storage Manager clients can be downloaded via FTP from the following servers: IBM internal: shasta.sanjose.ibm.com IBM external: index.storsys.ibm.com

Content Manager	NT, AIX	— Installed Content Manager client 2.4 (or later)
Content Manager OnDemand	NT, AIX	— installed Content Manager OnDemand server 2.2.1.8

CSLD — Archive combinations

This section lists the possible platform combinations in which CSLD can work with the different archives. The combinations marked with an X are supported, those marked with a '—' are not supported, and those marked with a '?' may work but are not tested and not supported yet.

ADSM/Tivoli Storage Manger

CSLD on ADSM/TSM on	NT	AIX
NT	X	X
AIX	X	X
HP-UX	X	X
SOLARIS	X	X
OS/390	X	X

Content Manager

CSLD on CM on	NT	AIX
NT	X	X
AIX	X	X
OS/390	X	X

Content Manager OnDemand

CSLD on CMOD on	NT	AIX
NT	X	X
AIX	X	X
HP-UX	X	X

SOLARIS	X	X
OS/400	—	—
OS/390	— [1]	—
[1]Will be shipped as a fixpack for IBM Content Manager CommonStore release 7.1.		

Chapter 4.2 Installation of multiple instances

It is possible to run more than one instance of the CMCS server. In other words, several independent sets of CMCS processes can be activated at the same time on the same machine.

While the same set of executables can be employed for all instances of the CMCS server, it is necessary to maintain distinct CMCS server configuration profiles and (except the Windows platforms) distinct users, one for each instance of the CMCS server.

All profiles will employ identical values of BINPATH to make use of the same set of binaries. To distinguish instance specific files every profile must define different values of INSTANCEPATH pointing to the instance directory.

All CMCS command invocations should include the option -i to specify the profile to be used (refer to"Chapter 5.3 Working with the CMCS server" on page 103). For unassisted operation, it is recommended that you configure the /etc/inittab file on AIX, or install corresponding CMCS services on Windows NT (refer to "Multiple installations of the CMCS service" on page 83), as appropriate.

The following steps are necessary for installing multiple instances:

1. Create instance users and directories.
2. Separate the CMCS server configuration profiles.

Create instance users and directories

AIX

For every CMCS server instance a user should be created. INSTANCEPATH should point to the users home directory. All instance related files are maintained in the users home directory.

Windows

Under Windows there should be only one CMCS user which installs the CMCS package. To separate the instance specific files every instance should have its own instance directory which has to be referenced by INSTANCEPATH in the instances configuration profile.

Separate the CMCS server configuration profiles

It is necessary that each profile be assigned a unique name or, if they have the same name, that they be placed in distinct directories in the file system. Additionally, these configuration profiles must differ with respect to

- The INSTANCEPATH which should be the directory where all instance dependent files are stored, i.e.
  - *CMCS configuration profile* (archint.ini)
  - *TRACEFILE* (if *TRACE* is not switched to **OFF**);
  - *CONFIG_FILE*, (archint.cfg);
  - *LOGPATH* (if *LOG* is not switched to **OFF**);
  - *WAITQ_FILE*, *ACTIVEQ_FILE*, and *SENDQ_FILE*.
  - *nodelock* file (containing the license passwords for the instance)
- The TCP/IP ports used for communication to remote CMCS modules, i.e.
  - *WEBPORT* (if the *WEBDPS* parameter is present and set to a non-zero value).
  - *DOMINOPORT* (if the *DOMINODP*S parameter is present and set to a non-zero value).

Chapter 4.3 Installation on UNIX

This section describes the steps to be performed to install the CSLD package on UNIX. In order to perform a base setup for archiving with CSLD, please see “Chapter 5.1 Base setup” on page 95

Base installation steps

To install CSLD correctly on your UNIX server the following steps have to be performed:

1. Install the CSLD software package
2. For every instance of CSLD do the following three steps:
   a. Create an CSLD instance user
   b. Log on as CSLD instance user
   c. Modify *.profile* file of the CSLD instance user

If not stated differently, root permission is required. The steps must be performed on the machine on which the CSLD will run.

Install the CSLD software package

AIX

To install the CSLD package on AIX, call SMIT and perform the following steps:

1. Install and Update Software.
2. Install/Update Selectable Software (Custom Install).
3. Install Software Products at Latest Level.
4. Install New Software Products at Latest Level.
5. Input device / directory for software.

Select your CD-ROM drive and then choose the components you want to install. The CSLD software will be installed in the directory **/usr/lpp/csld**. This directory will be created if it does not already exist.

Create a CSLD instance user

To create and customize the instance user perform the following steps:

1. Create the CSLD instance user.
2. Log on as CSLD instance user.
3. Modify the .profile:
   on AIX add the line
   ```
   ./usr/lpp/csld/bin/csenv.sh
   ```

The home directory of the user should be the instance directory and therefore be specified under INSTANCEPATH in the instance profile.

Chapter 4.4 Installation on Windows

This chapter describes the steps to be performed to install the CSLD package on Windows. In order to perform a base setup for CSLD, please see “Chapter 5.1 Base setup” on page 95

Base installation steps

Install the CSLD server code by running the setup program on the CSLD CD. This will run an install shield that guides you through installation. The setup program is located in the root directory of your product CD-ROM

Additional installation steps for Tivoli Storage Management

On Windows, the following settings are required when Tivoli Storage Manager archives are used:

1. Create client option files.
2. Set environment variables for Tivoli Storage Manager API client

Create Client option files

For each Tivoli Storage Manager archive server specified in the CMCS server configuration profile by:

1. ARCHIVE xx
2. STORAGETYPE ADSM
3. SERVER <server_name>

there must exist a separate client option file <server_name>.opt containing all Tivoli Storage Manager parameters required for connecting to the specified Tivoli Storage Manager server. All of these client option files must reside in the same directory, and this directory must contain a client option file named dsm.opt. Although the contents of dsm.opt are ignored by CMCS (it could be empty), dsm.opt must exist, and the environment variable DSMI_CONFIG must point to it (see next section).

Set environment variables for Tivoli Storage Manager API client

The Tivoli Storage Manager API client needs the following three environment variables to locate certain files:

1. *DSMI_CONFIG* - fully qualified name (path + file name) of the client option file named dsm.opt
2. *DSMI_DIR* - the path to the directory where the Tivoli Storage Manager message file named dscameng.txt resides

3. *DSMI_LOG* - the path to the directory where the Tivoli Storage Manager error log named dsierror.log resides.

In addition, the location of the two Tivoli Storage Manager API DLLs named adsm32.dll and blkhook.dll must be included in the *PATH* environment variable, e.g.:

1. set DSMI_CONFIG to `c:\ibm\csld\adsm_opt\dsm.opt`
2. set DSMI_DIR to `c:\win32app\ibm\adsm\baclient`
3. set DSMI_LOG to `c:\ibm\csld\adsm_opt`
4. set PATH to `%path%;c:\win32app\ibm\adsm\baclient;c:\win32app\ibm\adsm\api\dll`

In this example, it is assumed that you have created a directory `c:\ibm\csld\adsm_opt` containing all client option files with the extension *.opt. You can define these environment variables at **Control panel->System->Environment**.

Installing the CMCS server as a service

On a Windows NT machine, the CMCS server component can be installed as a Windows NT service. This allows CMCS server to run continuously even if all users have logged off.

Note:
The installation of the CMCS server as a service should be done only in the last step of the installation. See also "Hints" on page 83. The service functionality is implemented as a separate program named archservice.exe. This program (exe) must reside in the same directory as the other CMCS server programs like archpro.exe . In addition to the service functionality, archservice.exe also implements the installation and deinstallation of the CMCS service. As soon as the CMCS service has been installed, it appears in the control panel's **Services** applet. The CMCS service can be started and stopped from the **Services** applet. When the CMCS service is removed, it disappears from the **Services** applet. The program archservice.exe contains none of the CMCS server functionality. Rather, the CMCS functionality is fully contained in archpro.exe and the other CMCS server executables. When the CMCS service () is started, it starts archpro.exe, which in turn starts the remaining programs. When the CMCS service is stopped, it stops archpro.exe, which in turn stops the other executables. The CMCS service checks every few seconds to see whether archpro.exe is still running. When the service detects that archpro.exe has died, the service will wait 1 minute and then restart archpro.exe .

Installation and deinstallation

Installation and deinstallation of CMCS service is performed as follows:

1. Open a command-line window.
2. Switch to the directory in which archservice.exe resides.
3. Enter **archservice install** to install the service.
4. Enter **archservice help** to get help.

Once the service is installed, the Windows NT Service Control Manager executes archservice.exe . It is not possible to run archservice.exe from the command line without specifying parameters. This way of executing the service is restricted to internal calls of the Windows NT Service Control Manager. During installation, some registry keys are created in the Windows NT registry. However, all modifications of the registry are displayed to the command-line window. Information about success or failure of installing, removing, starting, or stopping archservice.exe can be viewed in the Windows NT Application Event Log by using the Event Viewer tool.

Starting and stopping

The CMCS service can be started and stopped from the **Services** applet. From the command line, these actions will be performed when using the commands **archservice start** and **archservice stop**. The status can be displayed using the command **archservice status**.

Multiple installations of the CMCS service

Beginning with CMCS version 1.6.0.0, it is possible to install multiple instances of the CMCS service on a single machine. Each instance of the service must have a different name (set via command-line option -n while installing), and each instance must use a separate fixed port (set via parameter *ARCHWIN_PORT* in the CMCS server configuration profile). This means that you have to specify a separate configuration profile for each instance of the CMCS service.

Example

- `archservice install -i c:\ibm\csld\instance02\archint.ini -n 2`
- `archservice remove -n 2`
- The first command installs CMCS as a service under the name CommonStore_2 .
- The second command removes CommonStore_2 from the **Services** applet.

Hints

1. It is recommended that CMCS (i.e. archpro) first be run directly from the command line and that you go through the whole installation with this setup. Only if this works without problem may you install CMCS as a service. Once this is done, there is no longer any screen output. However,

the CMCS trace file still contains all warnings and error messages. During initial testing, the tracing should therefore be activated in the CMCS server configuration profile.

2. When using Content Manager as an archive, it is necessary to run the CMCS service under the account of a Content Manager user (**Services applet->Start-Up->Log On As->account**). This provides the necessary environment variables for the service, which might otherwise not be available if the service were to run under System Account.
3. In order to run CMCS service under a normal user account, that account requires the Log on as a service right. This right is assigned in **User Manager->User rights policy**. Please do not forget to activate the Show advanced user rights option.
4. Furthermore, the exchange paths defined by the keywords *BASEPATH* and *ARCHPATH* might not be available when running CMCS as a service. In this case, archpro will not start up, since archpro verifies that all paths are available. Again, the trace file contains all corresponding error messages.

Chapter 4.5 Installed files

Executable files

This section describes the most important executables files installed by CSLD. Not described are shared libraries used by CSLD. The CSLD for Lotus Domino package contains the following executables:

> **Note**
> The names of the executable files listed below are valid for installation on UNIX. When installed on Windows NT, the suffix .exe must be added to those files without a suffix.

archadmin
This program allows a (remote) connection to a CMCS server in order to view the messages issued by the CMCS server. It is possible to open the connection across machine and platform boundaries.

archagent
This is the CMCS Tivoli Storage Manager agent program which runs at the request of the CMCS server. It is responsible for archiving/retrieving the data and contains the Tivoli Storage Manager API client functionality. The CMCS server starts as many parallel Tivoli Storage Manager agents as are defined in the CMCS server profile under the keyword *ADSMAGENTS*.

> **Note:**
> This program is needed only if Tivoli Storage Manager is the archiving system.

archagentod
Same function as archagent, but for CMOD. The CMCS server starts as many CMOD agents as are defined in the CMCS server profile under the keyword *ODAGENTS*

> **Note:**
> This program is needed only if CMOD is the archiving system.

archagentvi
Same function as archagent , but for Content Manager. The CMCS

server starts as many Content Manager agents as are defined in the CMCS server profile under the keyword *VIAGENTS*.

> **Note:**
> This program is needed only if Content Manager is the archiving system.

archpro
This is a continuously-running CMCS server main program which controls all the other CMCS components.

archservice.exe
This program contains the whole service functionality of CMCS on Windows NT.

> **Note:**
> This program is available only on Windows NT.

archstop
This program stops the complete CMCS via a regular shutdown.

csld This is the CSLD task executable. The CSLD task is the only Notes-related part of the CSLD server. It converts Notes documents to files and vice-versa.

archcls.jar and csmimes.properties
Archcls.jar is a CMCS Java class library required for accessing CMCS server via HTTP. Among other things it contains the Web dispatcher and DOMINO dispatcher functionality. The CMCS server starts one Java Runtime process with as many DOMINO dispatcher threads as are defined in the CMCS server profile under the keyword *DOMINODPS*. The csmimes.properties is a resource file used by the Web dispatcher. This file specifies the mappings of archive content types to MIME types. The Web browser needs this information to display retrieved content correctly.

infobus.jar
This is a third party Java class library.

ivjsap20.jar
This is a third party Java class library.

mail.jar
This is a third party Java class library.

Sample configuration profiles for CMCS server

The CMCS server configuration profile is also referred to as the initialization file or ini file. If, when performing an operation (starting, stopping, etc.) with the CMCS server, the option -i is not used, the CMCS server looks for the default configuration profile (name: archint.ini) in the local directory. You must create this file by taking one of the three sample configuration profiles corresponding to the type of archive you wish to configure making the appropriate settings, and then renaming the file. It contains all necessary information for the CMCS server and its connections to the archiving systems and to the application to be archived. It has to be configured when the CMCS server is set up, see "Chapter 5.1 Base setup" on page 95

Note:
The CMCS server needs only this file for customization (no sideinfo, no environment variables).

Note:
The CSLD Task is configured by a profile document in a Lotus Notes database and by environment variables.

The following three sample configuration profiles are provided as part of the CSLD package. They are intended for your use as templates. In other words, after you have opened and edited them (i.e. customized them according to your setup) as described below, they will assume the role of the CMCS server configuration profile whenever you specify them using the option -i.

archint_sample_tsm.ini

This is the sample profile which you should take as a template when using Tivoli Storage Manager as an archiving system.

archint_sample_cm.ini

This is the sample profile which you should take as a template when using Content Manager as an archiving system.

archint_sample_cmod.ini

Sample profile which you should take as a template when using Content Manager as an archiving system.

Chapter 4.6 Files created at runtime

The CMCS server configuration file archint.cfg

Before you can use CMCS server you have to register the passwords needed to access the archives and the application to be archived. This is done by calling 'archpro –f serverpasswd' and 'archpro –f r3passwd'. This will create or modify the CMCS server configuration file archint.cfg automatically. The passwords in it are stored in encrypted form.

Trace files

CMCS server can produce different trace files to provide you and the support team with information about error cases. Tracing can be configured in the CMCS server profile. The following trace files are available.

archint_startup.trace

This file contains only error information on starting and stopping the CMCS server.

archservice.trace (only for Windows NT)

This file contains only error information on starting and stopping the CMCS service.

archint.trace

This file is written by the CMCS server if specified in the CMCS server configuration profile. The name of this file is also configurable in the CMCS server configuration profile. This file contains all CMCS server trace information (including information about starting, stopping, file names, and errors.

Note
All trace files are configurable in the CMCS server configuration profile.

Log files

Operation Log

The operation log file contains the names of all files which are archived and retrieved. Errors will be recorded as well. A new file is created for each day. Log file names are all of the form aiYYYYMMDD.log.

Examples

1. `ai19980101.log`
2. `ai19980102.log`
3. `ai19980103.log`

> **Note**
> Log files are written automatically if specified in the CMCS server configuration profile.

Error Log

The error log file records all errors occurring in a CMCS server operation. The error log file is a text file. The entries in the error log file consist of one section per failed operation. The error log file grows without size limitation. Name and location of the error log file can be configured in CMCS server profile by the ERRORLOG_FILE keyword. If the ERRORLOG_FILE keyword is not specified in your CMCS server configuration profile the error log file is named cserror.log and located in the directory specified by INSTANCEPATH.

Each error section in the error log file starts with a line containing the date and time when the error occurred. In the next four lines follows the detailed error information consisting of:

1. information about the code module where the error occurred.
2. internal return code. This is the return code returned by the CMCS server.
3. external return code. This contains additional return codes from an API if the error occurred in an API call.
4. error description. Additional textual information about the error.

Example

```
[2000/08/24  14:01:39]
1. error reported by CSS 2.2.0.6 module 'archagentvi (CM agent)',
2. internal return code is ' (6288)',
3. external return codes are RC = '6288', ExtRC = '9030', Reason = '0',
4. error description = "CSS6000E: API call 'SimLibLogon()' failed"
```

Part 5. CMCS Server Administration

Chapter 5.1 Base setup

This section provides an overview of the basic steps to be performed in order to get the connection running between the CSLD task process and the CMCS server.

1. Create a Content Manager user.
2. Create a Content Manager index class.

CMCS server configuration profile setup

The installation package includes three sample profiles: `archint_sample_tsm.ini` (for Tivoli Storage Manager), `archint_sample_cm.ini` (for Content Manager), and `archint_sample_cmod.ini` (for Content Manager OnDemand).They are intended for your use as templates. In other words, after you have opened and edited them (i.e. customized them according to your setup) as described below, they will assume the role of the CMCS server configuration profile whenever you specify them using the option *-i* (see also "The option -i" on page 103).

The CMCS server reads the profile before it executes.

Any change in this profile requires that the CMCS server be restarted to effect the changes.

The following rules apply to the profile's syntax:

- Every line is analyzed separately.
- Keywords can start in any column of the line.
- There must not be any strings — except for blanks — before keywords.
- If a keyword is encountered several times, the last one is used.
- Scanning of the file continues until the keyword *END* is encountered or the end of file is reached.
- It is strongly recommended that you not use keywords as the values of keywords. However, keywords can be used as values *when enclosed in single quotes*, e.g. `SERVER 'VI'`.

For the basic setup, the following steps will need to be performed:

1. Copy the CMCS server sample profile to a file named `archint.ini`.
2. Adjust the paths specified in the profile to correspond to your environment, e.g. *BINPATH*, *LOGPATH*, *TEMPPATH*.
3. Enter the settings for an archive created in your archiving system. At this point, it will be enough if a test archive is defined in the CMCS server

profile. For each logical archive in `archint.ini`, the following settings (depending upon your archiving environment) are necessary:

For Tivoli Storage Manager:

- `STORAGETYPE ADSM`
- `SERVER`
- `MGMT_CLASS`
- `ADSMNODE`

> **Note:**
> If your Tivoli Storage Manager server is named `ADSM`, please include that name as `'ADSM'` (i.e. enclosed in single quotes).

For Content Manager:

- `STORAGETYPE VI`
- `LIBSERVER`
- `INDEX_CLASS`
- `VIUSER`

For CMOD:

- `STORAGETYPE ONDEMAND`
- `ODHOST`
- `APPGROUP`
- `APPLICATION`
- `FOLDER`
- `ODUSER`

4. Set the system type from which your CMCS server should archive. There are different systems from which CMCS server can archive. Depending on the licenses you bought, you specify the system type in `archint.ini`. Using the keyword *SYSTEMTYPE*, you can specify if archiving will be done in an SAP R/3 system, a Lotus Domino system, or in both systems.

   To archive from an SAP R/3 system, specify:

   ```
   SYSTEMTYPE SAPSYSTEM
   ```

   To archive from a Lotus Domino system, specify:

   ```
   SYSTEMTYPE DOMINOSYSTEM
   ```

   To archive both SAP R/3 and Lotus Domino, specify:

   ```
   SYSTEMTYPE SAPSYSTEM DOMINOSYSTEM
   ```

   If the keyword *SYSTEMTYPE* is not specified, its value defaults to:

```
SYSTEMTYPE    SAPSYSTEM DOMINOSYSTEM
```

5. Set the number of agents to be used for each archiving system. The agent of the archiving system you are using should be set at least to 1.

> **Note:**
> CMCS supports a mixed archiving environment. In one CMCS profile you can specify agents for different archiving systems.
>
> **EXAMPLE:**
>
> ```
> ADSMAGENTS                1
> VIAGENTS                  2
> ODAGENTS                  1
> ```

Configuring the CMCS server for http

For communication via http the number of Web dispatchers must be at least 1 and a valid Web port must be specified in the CommonStore server profile

Passwords

When you start the CMCS server for the first time, you will be prompted for passwords.

Please start the CMCS server by entering from the command line:

```
archpro -f serverpasswd
```

You will be prompted for the passwords for each defined `ARCHIVE` which was found in the CMCS server configuration file `archint.ini`.

> **Note:**
> Each time you want to change an archive password, please enter this command.

Getting started

Start the CMCS server by issuing the command `archpro` or

```
archpro -i <ini file>
```

Please see “Starting the CMCS server” on page 103.

Chapter 5.2 License setup

This section describes the new license model of IBM Content Manager CommonStore 7.1 and how to setup licensing for both try&buy usage and productive usage.

Changes to Version 2.1.5

The licensing model of CommonStore since version 2.1.5 needed the installation and configuration of *License Use Runtime* which is a set of services providing sophisticated license management.

Beginning with release 7.1 of CommonStore, License Use Runtime is no longer needed. All licenses are of the type *non-runtime-based simple nodelocked* and are managed by CommonStore itself. This simplifies license setup dramatically.

New license types and certificate files

CommonStore distinguishes between two license types:

License type	Features	License file
Try&buy license	— 90 day validity (beginning with first start) — full functionality	— None required.
Production license	— unlimited validity — full functionality	— **CSSAP7.lic** for SAP — **CSLD7.lic** for Lotus Domino

CommonStore profile keywords

Since version 2.1.5 there have been two keywords and four subkeywords concerning license setup:

1. INSTANCENAME
2. SYSTEMTYPE
   - PRODUCTIONSYSTEM
   - TESTSYSTEM
   - SAPSYSTEM
   - DOMINOSYSTEM

Beginning with release 7.1 of CommonStore, the keyword INSTANCENAME is obsolete and the keyword SYSTEMTYPE can only have the values SAPSYSTEM, DOMINOSYSTEM, or both. This will decide for which product CommonStore should request a license.

The values PRODUCTIONSYSTEM and TESTSYSTEM are obsolete because there will no longer be a distinction between a productive license and a test license.

The new keyword INSTANCEPATH specifies the path in which instance specific files should be stored. Since there is exactly one nodelock file, containing the license password, for each instance of CommonStore server, this file cannot be stored in the binary path when running more than one instance of CommonStore server. Every instance will have its own nodelock file in its INSTANCEPATH.

Enrolling productive licenses

To run CommonStore server with a productive license, the license has to be enrolled previously. To enroll a productive license call archpro with the '-f license' parameter:

```
> archpro -f license
>
>
>       ************************************************************
>       * IBM Content Manager CommonStore - Server 7.1.0.0         *
>       * (c) Copyright IBM Corporation, 1997-2000 All Rights Reserved *
>       * Build  92, Compiled at Nov 16 2000.                      *
>       ************************************************************
>
> CSS0213I: Please enter the name of the certificate file:
```

Please enter the path to the certificate file containing the productive license. For SAP enter the path to the file **CSSAP7.lic**, and for Lotus Domino enter the path to **CSLD7.lic**. These files are placed in the directory named **license** on the respective product CD.

If the path is entered correctly the license password will be extracted from the file and registered in the **nodelock** file in INSTANCEPATH.

Now archpro can be started for productive use.

Note

The enrollment of the production license has to be done for each instance of CommonStore.

Note:
If you want to run the same instance of CommonStore server for both SAP and Lotus Domino you must specify in the profile:

SYSTEMTYPE SAPSYSTEM DOMINOSYSTEM

and you have to do the enrollment for both licenses.

Enrolling the try&buy license

When starting archpro without having enrolled any license previously, the following dialog appears:

```
> archpro
>
>           ****************************************************************
>           * IBM Content Manager CommonStore - Server 7.1.0.0             *
>           * (c) Copyright IBM Corporation, 1997-2000 All Rights Reserved *
>           * Build  92, Compiled at Nov 16 2000.                          *
>           ****************************************************************
>
> CSS0926E: Archpro could not find the certificate file
> CSS0910I: Trying to get a LUM Production License for IBM Content Manager
>           CommonStore for Lotus Domino
> CSS0916E: Could not get a LUM Production License for IBM Content Manager
>           CommonStore for Lotus Domino
> CSS0912I: Trying to get a LUM Try and Buy License for IBM Content Manager
>           CommonStore for Lotus Domino
> CSS0933I:
>           ****************************************************************
>           *                                                              *
>           * IBM Content Manager CommonStore for Lotus Domino             *
>           * does not have a license enrolled.                            *
>           *                                                              *
>           * If you have purchased a license for this product, then you   *
>           * have a production license, which you should now enroll.      *
>           * To do this, select 'EXIT', and call archpro -f license.      *
>           * Archpro will ask you for the full qualified path to the      *
>           * production license file. The product password will be        *
>           * installed in the 'nodelock' file in the directory            *
>           * specified by the INSTANCEPATH keyword in the CommonStore     *
>           * profile.                                                     *
>           * If you have not purchased a license for this product, you    *
>           * may select 'CONTINUE' to enroll an Evaluation license.       *
>           *                                                              *
>           *      to EXIT     enter: 1                                    *
>           *      to CONTINUE enter: 2                                    *
>           *                                                              *
>           ****************************************************************
```

Type '2' and press enter.

The try&buy license will then be saved in the nodelock file in INSTANCEPATH and will expire after 90 days.

Note:
The enrollment of the try&buy license can be done only once. If you try it again, archpro will show the following message:

```
CSS0926E: Archpro could not find the certificate file
```

Running more than one instance with try&buy license

The try&buy license can be enrolled exactly once. During this enrollment a nodelock file is created in the current INSTANCEPATH. The license in this nodelock file will expire after 90 days. If you want to run multiple instances using the try&buy license you must copy this nodelock file to the INSTANCEPATH of each CommonStore instance.

Chapter 5.3 Working with the CMCS server

Starting the CMCS server

The CMCS server consists of several components. They are all controlled by `archpro`, the continuously-running CMCS main program. In order to start the entire CMCS server, it is sufficient to simply start `archpro`. `archpro` then reads the CMCS ini file (either `archint.ini` in the current directory or the specified ini file) and automatically starts all configured components (so-called child processes).

The syntax for starting `archpro` directly from the command line is as follows:

```
archpro [-i <ini file>] [-n <name>]
```

The option -i

If archpro is started without any parameters the CMCS server profile has to be located in the current directory and must be named `archint.ini.` .

If you want archpro to use a profile located in another directory as the current one or a profile that is not named `archint.ini` you have to specify the fully qualified name (path + file name) of the profile.

The option -n

The option *-n* `<name>` is necessary only on Windows NT and when the CMCS server is already installed as a service (see "Installing the CMCS server as a service" on page 82). In this case, it specifies the name of the CMCS instance. Otherwise, this option is ignored.

Examples:

- `archpro`
- `archpro -i c:\ibm\csld\instance02\archint.ini2 -n 2 (NT)`

You can get on-line help for using the command syntax of **archpro** by entering the command **archpro -h**.

When starting the CMCS server, the connections to all configured archive servers are verified. When an archive server is not available, CMCS issues an error message describing the problem and then stops immediately. It is possible to disable this immediate shutdown by setting *CHECK_ARCHIVE_SERVER* to **OFF** in the CMCS ini file. In this case, if an archive server is not available when starting up CMCS, only a warning is

issued. It is clear that CMCS cannot perform a successful operation on an archive which is not available. For this reason, disabling the initial check of the archives makes sense only if you are sure that the archives will be available before it comes time to actually access them.

As soon as CMCS is running, the main component (`archpro`) checks the other child processes every few seconds. If `archpro` detects that some process is no longer active, this component will be automatically restarted. When restarting, the archives are not checked for availability.

In order to allow (on Windows NT) CMCS to run continuously even if all of the users have logged off, the CMCS service must be started. This service must be installed before it can be used (see "Installing the CMCS server as a service" on page 82). This service can be started from the **Services** applet or by using the command **archservice start**. When the service is started, it executes `archpro`, checks regularly whether or not `archpro` is still active, and restarting it if necessary.

The CMCS service can be started either from the **Services** applet of the Control Panel or using the command **archservice start**.

Stopping the CMCS server

The CMCS server is stopped using the `archstop` program. This program opens a connection to the main component (`archpro`) and sends it a shutdown command. Whenever `archpro` receives the shutdown command, it shuts down all child processes and stops itself. All stop messages issued by the CMCS server will now also appear in the **archstop** command window.

The syntax for this command is as follows:

```
archstop [-p <port>] [-i <ini file>] [now]
```

The parameter *<port>* is the fixed port number specified in the CMCS server configuration file `archint.ini` under the keyword *ARCHWIN_PORT*. Alternately, you could also specify the corresponding CMCS server ini file. In this case, the port is obtained directly from there. Without the keyword *NOW*, the CMCS server stops only after it has finished all currently active jobs. In order to stop the CMCS server immediately, the keyword *NOW* must be used.

Examples:

- `archstop`
- `archstop -p 5510`
- `archstop -p 5510 now`
- `archstop -i c:\ibm\csld\instance01\archint.ini`

Again, you can obtain on-line help by entering the command **archstop -h**.

The CMCS service can be stopped either from the **Services** applet of the Control Panel or by issuing the command **archservice stop**. The **archstop** command cannot be used to stop the service. **archstop** will stop only `archpro`; however, the service will soon detect that `archpro` is down and will restart `archpro` automatically.

Chapter 5.4 Troubleshooting

CMCS server

Problem:

Archiving takes too long. Where is the bottleneck?

Solution:

Possible bottlenecks are

- The Domino server is too slow. In most cases, slow archiving is caused by bad Notes response times.
- The archive server is overloaded. Content Manager and ADSM/TSM can simultaneously import only a limited number of documents.
- The number of archiving agents is too low. If you create archiving jobs faster than the current number of CMCS agents can import documents, the documents will be written to a queue. Increase the number of agents (keywords *VIAGENTS* and *ADSMAGENTS* in file `archint.ini`). Do not specify more than ten agents, because this will slow down the system due to swapping.
- The number of CSLD dispatchers is too low (keyword *DOMINODPS* in file `archint.ini`), so the CSLD task has to wait until a dispatcher becomes available.
- The CSLD server has too little memory.

Problem:

CMCS does not start.

Solution:

CMCS checks at startup to determine whether or not all settings are correct and whether or not it was possible to establish a connection to the archive servers. Further, all paths and file specified in the CMCS configuration file `archint.ini` must be accessible. Proceed as follows:

1. Make sure that the paths and files are accessible for the user. Do not use file names where directories are expected or vice versa. When there is no screen output of `archpro.exe` available, enable the tracing (*TRACE* **ON**) and look for error messages in the trace files.
2. Determine which child process has problems with the connection. There should be a corresponding error message displayed by `archpro.exe`.

The same error message is also found in the trace file. If you cannot find out which component failed, check them separately. Enable only one dispatcher, and disable all agents or vice versa.

Problem:
How can I tell that the CMCS child component is working (is ready)?

Solution:
The connection is working when `archpro.exe` displays the following typical messages:

- `archpro.exe` is informed that `xxx` has started (from dispatcher and agents).
- `archpro.exe` is informed that `xxx` is ready to obtain order (only from agents).
- Dispatcher is ready and waiting for RFCs ... (only from dispatcher).

The message about `xxx`'s start is sent by the child process immediately after the start. It means that a connection between `archpro.exe` and the child process has been established. The ready message is sent by the child after the corresponding check has been done. For the dispatchers, this means that the connection has been established. For the agents, it means that they have performed a log-on to the archive server and have verified all corresponding settings (user name, password, management class, index class, ...). When you see the ready message, you know that this component is working completely.

Problem:
The Windows NT command **net start/stop** does not start/stop the CMCS service.

Solution:
Use CMCS's **archservice start/stop** command instead.

CSLD task

Instead of error messages, jobs contain "No message found for..."
The error message file `csnmsg.msg` cannot be found because the environment variable *CSNBASE* is not set correctly to the CS binary directory, or because `csnmsg.msg` has been deleted.

I cannot shutdown a task for some reason. After I stopped a task using the NT Task Manager, Notes shows strange behavior.
The Notes internal thread and session handling is highly volatile. Log out and in again, or reboot.

After starting a task, it displays the CSLD user name, then ″hangs″ (does not display a log-in prompt).

You have two copies of `nnotes.dll`, which was installed with the Notes Client. Remove one of them.

When starting up a CSLD task, it stops with the error message ″The ID file is locked by another process. Try again later.″

You have started another CSLD task that is still waiting until you type in the password. When the input prompt appears, the ID file is locked. During job processing, starting a CSLD task will do no harm.

A job contains the error message ″The archive server returned error code `<errorcode>`.″

Content Manager returns error IDs rather than error messages. Look up the error description in your Content Manager or OnDemand documentation.

After processing, a job document contains the Content Manager error ″Archiving to CommonStore failed. The archive server returned error code 6056.″ Most probably a Content Manager setup problem. Error 6056 signals a generic Content Manager error. To find out the exact error code, perform the following steps:

1. Start `archpro.exe` in the debugging mode (if not already started in this mode) and repeat the action that caused the error. A log file entry will be created in `archint.trace`.
2. Search for `6056`, starting from the bottom of the file. Right next to `6056`, you will find the extension error code `ExtRC=<code>`. The extension error code contains the ″real″ error number.
3. Look up the extension error code in the Content Manager documentation for a description for this error code.

Extension Code = 7389

When an index class is created, you must wait a while until it can be used; this is because Content Manager must create a `.dll` file for the index class.

Extension Code = 7937

The size if a Notes field being archived is bigger than the length specified in the attribute definition. For example, if you reserve 100 bytes for a ″Subject″ character attribute, and you try to archive a mail document whose subject field is 110 bytes, you will receive this error message.

A CSLD task ignores the jobs that I created.

First, make sure that the database server field **sourceSrv** in the job is identical to the server given in the task's profile (case is ignored). Then, check if both are given in abbreviated Notes syntax name/Domain, for example, `BOEDOM1/IBM_IDE`. Also, check your job's

requestType field. See “Chapter 8.1 Creating job documents” on page 153 for a description of job parameters. Further, the CSLD user ID (the ID the CSLD task is running under) must be assigned the role [CSLDUsers]. See “7.1.2 Creating the job database” on page 123 for details. Another reason for the jobs being ignored by the CSLD task might be that the current time lies outside the polling interval defined in the CSLD configuration database. See “Defining profile documents” on page 125 for details.

The archive server returned error code 1.

The archive does *not* have an index class with the name you specified. Check the spelling of your index class name in the CMCS configuration file `archint.ini`. Index class names are case-sensitive.

″Export filter is unable to export document to <format> file.″

For document rasterizing (RTF, ASCII), CSLD uses the export filters shipped with Lotus Notes. These export filters provide only basic functionality. Thus, complex documents containing sections, tables, etc. might cause the export filter to fail.

I have set the Notes password using csld -f serverpasswd, but the task still prompts for the Notes ID.

Answer: Check whether your *notes.ini* file contains the line EXTMGR_ADDINS=CSLDExtPwd.dll. Also check whether the ID the task is running under is the same as the ID for which you set the password. Use the -i parameter to specify a notes.ini file containing a particular ID.

How do I find out under which ID my task is running?

Answer: When the task starts up, it displays the current ID. The ID is always stored in your default *notes.ini* file. If you use the -i parameter to specify a different .ini file, the task will use the ID in that file. When you use the Notes client to switch to a different ID, the ID will be stored in the default *notes.ini*file.

When starting a CSLD task (csld.exe), it aborts with an error message that notes.dll cannot be found.

Answer: CSLD is a Notes application, and therefore requires that the Notes dynamic link libraries are installed. The notes.dll library belongs to the Notes client, and is located under the Notes installation directory. Add this directory (e.g. `c:\notes`) to the PATH environment variable.

I have entered a user name in the ″CSLD administrators to send error notification mails to″ field of the profile document, but instead of an email I get the error message that the user ID does not exist.

Answer: The user name must be added by selecting it from the local address book. You probably entered the user name manually.

I started the archpro but it does not process my jobs.

Answer: The archpro has nothing to do with Notes! It is a application independent archiving engine archiving files that either SAP or the CSLD task provides. The CSLD task is the CommonStore server component that processes jobs.

When using the browser viewing feature, the browser shows weird characters instead of the real content.

Answer: You probably forgot to add an entry to file csmimes.properties in order to map the content type to a MIME type to be understood by the browser. Another explanation would be that you used the default mapping (file extension .unk) to map all files to the same content type instead of assigning every file extension its own content type.

Content Manager

If you encounter problems with CMCS and Content Manager, stop the CMCS server. Supply the parameter *TRACE VI* in the CMCS profile and switch folder manager tracing to **ON** by setting the environment variable *FRNFMTRACE* to the value `/p` on the command line (for release 2.3 or later of the Content Manager Client toolkit). Then, manually re-start CMCS using the **archpro** command in the same shell session.

Check the resulting CMCS trace file (usually `archint.trace`) as well as the Content Manager folder manager protocol (`fmtrace.txt`) and consult the Content Manager error logging facility on that library server to which the problematic logical archive refers. Consult the Content Manager publication *Messages and Codes* (publication no. GC31-9065-03) for detailed error descriptions or refer to the *System Administration Guide* (publication no. SC31-7774-03) for collecting further trace information. When in doubt, double-check the Content Manager key field, index class, and workflow definitions.

Part 6. Customizing the Archives

The CSLD server is capable of interfacing with three different archiving systems: Tivoli Storage Manager, Content Manager, and Content Manager OnDemand. Customizing these archiving systems is described in the following chapters.

Chapter 6.1 Base setup

Depending upon the archiving system you are using, proceed according to the steps described in the appropriate section below.

Tivoli Storage Manager

1. Create a Tivoli Storage Manager management class.
2. Create a Tivoli Storage Manager node.

Content Manager

1. Create a Content Manager user account.
2. Create a Content Manager index class.
3. Create a Content Manager workbasket.

CMOD

1. Create a CMOD user account.
2. Create a CMOD application group.
3. Create a CMOD application.
4. Create a CMOD folder.

Chapter 6.2 CSLD related setup

Before documents can be archived, some administrative steps must be performed in the archive. In this chapter, we mainly cover Content Manager and OnDemand. TSM does not support indexing, so there are no special steps to perform besides creating management classes.

CSLD administration in Content Manager and OnDemand includes the following steps:

Creating index classes (Content Manager) and application groups (OnDemand)

In Content Manager and Content Manager OnDemand, documents of the same type are grouped in index classes and application groups, respectively. Documents of the same index class/application group have the same set of descriptive search attributes. CSLD maps Notes document fields (items) to index attributes. For every Notes document type (form) that you want to archive, an index class / application group must be created. For every Notes field that you want to be able to search for in the archive, an attribute must be created. For a given Notes field type, you can determine the Content Manager attribute type from the following table.

Table 1. Content Manager attribute types

Notes field type	Content Manager attribute type
Text	Character or variable character
Number	Integer, Long Integer or Decimal
Date only	Date
Time only	Time
Date and Time	Timestamp

The following table shows the mapping of Notes field types to OnDemand attribute types:

Table 2. OnDemand attribute types

Notes field type	OnDemand attribute type	Format
Text	String	
Number	Integer, Small Integer or Decimal	
Date only	Date	%Y-%m-%d

Table 2. OnDemand attribute types (continued)

Time only	Time	%H.%M.%S
Date and Time	Variable String	Minimum length 30 characters

When the ″restore to original database″ and ″restore by original user″ features are used, the index class/application group must contain the following attributes:

CSLDOrigUser:

Type variable character. Maximum length is the length of the longest Notes user name in canonical format in your Domino domain. Used to store the Notes user name.

CSLDOrigDB:

Type character. Length: 17 characters. Used to store the replica ID of the database the document was archived from.

For OnDemand only, the following attributes must be defined for every application group storing CSLD documents:

Table 3. OnDemand attribute names

Attribute Name	Length	Description
WORKBASKET	128	Name of the ″virtual″ workbasket a document has been archived to
FOLDER	60	Used to store the folder name for Notes folder archiving
CONTENT_TYPE	80	The content type as defined in the content type mapping table of the configuration database. Only used for browser viewing to map content types to MIME types
ORIGFILENAME	254	Used to store and restore the original filename
DOC_ID	60	Stores the document's unique id
ITEMTYPE	254	Describes whether the document is a document or a folder

All these attributes are of type *Variable String*, attribute case *Mixed.*

For every index class/application group to archive Notes documents to, you must define a logical archive in file *archint.ini* to make it known to CSLD. The logical archive ID can then be used in a document mapping as a valid target to archive documents to.

CSLD will login to the archive with the CSLD archive ID specified in *archint.ini*. This ID must be given full rights to the index class/application group. You should create a separate user ID just for CSLD. Whether other archive user IDs should be granted access to the index classes/application groups depends on your requirements. If the archive is only accessed from CSLD, only the CSLD archive ID and the archive administrator should have access to the index class/application group.

Preparing Content Manager and OnDemand for Notes folder archiving:

To archive and restore complete Notes folder structures, the CSLD archive administrator must define a Content Manager index class or OnDemand application group to hold CSLD internal information. Using the archive's security features, the administrator must ensure that the index class/application group is not made accessible to users for folder browsing. No document mapping for the index class/application group may exist to ensure that users cannot access the internal information.

This index class or application group must define the following attributes:

- **CSLDOrigUser** - variable character, extended alphanumeric, should be able to hold the largest possible Notes user name in canonical format; recommendation is 256 characters.
- **CSLDOrigDB** - variable character, extended alphanumeric, holds the replica ID of the Notes database the archived folder comes from. Should be sized to 17 characters.
- **CSLDFolderName** - variable character, extended alphanumeric, holds the name (or hierarchical name structure) of the Notes folder or subfolder. Should be sized to 256 characters, or larger to archive deeper folder structures
- **CSLDFolderAlias** - variable character, extended alphanumeric, holds possible aliases of the archived Notes folder. Should be sized to at least 100 characters. Can be smaller if no aliases are specified in the application.

You can choose an arbitrary name for the index class or application group. Then, create a logical archive ID for the index class/application group in file *archint.ini.* Finally, enter the name of the archive ID in the *Folder Archive ID* field of your archive task's and retrieve task's profile document.

For Tivoli Storage Manager, folders can only be archived without preserving the folder structure.

Part 7. CSLD Administration

Chapter 7.1 Base setup in Lotus Notes

7.1.1 Creating the CSLD user

The CSLD user is the Notes user under whose ID CSLD archiving and retrieval tasks will run. It is a regular user ID, but for security reasons, it should not be the ID of an existing user. Nor should it be the CSLD system administrators ID. Remember that the CSLD user has at least 'Editor' rights to all databases its CSLD task is assigned to, so he can modify all documents. Create the CSLD user like any other user. Follow the steps in the *Domino Administrator's Guide*. You can also create more than one CSLD user ID. This will allow you to run different CSLD tasks under different CSLD user IDs.

7.1.2 Creating the job database

To create the job database, proceed as follows:

1. On a Domino server, create a new database from the `CSLDJobs.ntf` template, located in the data directory under your installation directory. Do not create a local database.
2. Add all users who are allowed to use CSLD as 'Authors' to the job database's ACL. Make sure that all users have the right to delete documents. *Under no circumstances should you issue other rights!* Instead of adding every single user to the ACL, we recommend using a pre-defined group.
3. Add the CSLD user as 'Editor' to the job database's ACL, thus allowing CSLD tasks to search the database for jobs.
4. Add the CSLD User as 'Editor' to the ACLs of all Notes databases that make use of CSLD archiving functionality, thus allowing CSLD tasks to work with the documents. If you use the CSLD setupDB tool to create default forms, the CSLD user must have 'Manager' rights in the database to which setupDB is applied.
5. Assign role `[CSLDUsers]`, which is defined in the job database, to all CSLD user IDs (the IDs under which CSLD tasks run). Under no circumstances should you add regular users to this role. This role allows CSLD tasks to "see" job documents, because in correctly created job documents, there is a readers field containing the role [CSLDUsers] and the job requester. If you do not define and correctly assign this role, all jobs remain unprocessed in the database.

7.1.3 Creating the configuration database

To create the configuration database, proceed as follows:

1. On a Domino server, create a new database from the `CSLDConfig.ntf` template, which is located in the data directory under your installation directory. Do not create a local database.
2. Modify the configuration database's ACL so that it consists only of the CSLD user, the CSLD administrator, and the server IDs.

Chapter 7.2 Configuration Database — Customizing

In a CSLD setup, all configuration is done by managing special documents in the configuration database. Before you can start any archiving or retrieval tasks, you must create configuration documents for these tasks.

This is described in the following sections.

Defining profile documents

Profile documents define the basic parameters for such CSLD archiving and retrieval tasks, e.g. scheduling and database information. When CSLD task executables (in the case of Windows NT, `csld.exe`) are launched, they are given, among other parameters, a profile name. CSLD tasks use this name to search the configuration database for profile documents with their name, and to read parameters from them. For example: Suppose that an archiving task responsible for CD database archiving is named **CDArchiver**. When started up, it looks for a profile document named **CDArchiver** in the configuration database.

To create a profile document, select **Create->CSLD Database Profile** from the Notes menu, or change to the **all Profiles** view and click the **New Profile** button. Then, fill in the following fields:

Profile Name
This is the name that is passed to the CSLD task executable `csld.exe` via the *-p* parameters. We recommend that you use no blanks in the name.

CSLD administrators to send error notification mails to
When severe errors occur (like full disks or lost network connections), notification documents are sent to the people or groups defined in this field. More precisely, the job document is sent that was processed while the error occurred. The job document contains a detailed error message. Usually this field contains the CSLD administrator. The user names must be added by selecting them from your local address book. Do not enter user names manually. The entries would look the same, but Notes will not be able to resolve the name.

On error notify user via e-mail
When set to ″yes″, users will receive an e-mail when a job results in an error. This e-mail contains the job document.

CSLD Task

This field is used to specify whether the task is an archiving task or a retrieval task.

Notes databases

All jobs in database

When this mode is selected, the CSLD task will process all jobs in the job database, no matter from which database they were created. All jobs will be processed sequentially. This mode is not really suitable for a large number of databases and maximum throughput (like with mail archiving) as every user creating a job must wait until all other jobs are completed. Use a number of CSLD tasks combined with the other two modes for more parallelism and higher throughput. **Important:** You cannot start two tasks that are both set to this mode and operate on the same job database, as they would both try to process the same jobs. See "Chapter 2.6 Scalability" on page 31 for details on scaling CSLD.

All jobs coming from selected Domino servers:

When this mode is selected, the task will only process jobs for the Domino servers given in the Servers field below. It will process jobs for all databases on these servers. For every given server one operating system thread will be spawned. That is, jobs for different servers will be processed concurrently. This mode is especially suitable for email archiving, where you should start one task for every mail server.

All jobs coming from selected databases or data directories:

When this mode is selected, the task will only process jobs for the databases entered in the field below. It is also possible to specify Notes data subdirectories, i.e. jobs from all databases in the given subdirectories will be processed. Suppose you have a number of subdirectories below the notes data directory, e.g. mail1, mail2 and mail3. If you enter ″mail1/*, mail2/*″ in the profile of task1 and mail3/* in the profile of task2, all jobs from databases in directories mail1 and mail2 will be processed by task1. Similarly, task3 will take care of all jobs from databases in directory mail3. The asterisk is required, otherwise CSLD will interpret a string like ″mail2″ as a database name and try to open it. It is possible to mix database names and data subdirectories, e.g. ″mail4/user1.nsf, mail3/*, mail2/user2.nsf″. For every database or database subdirectory, the Domino server must be added to the Servers field. If less servers are given than databases or database directories, the databases/database directories without a server are assumed to be on the last server listed in the Servers field.

Important
Server names must be entered in abbreviated format, not canonical format! Otherwise the CSLD task will not find the jobs from databases on these servers.

For every database or database subdirectory, one operating system thread will be spawned. That is, jobs for different databases or subdirectories will be processed concurrently. Machines with multiprocessors will benefit from CSLD multitasking. However, too many databases or subdirectories will result in server thrashing, because the operating system cannot handle too many threads and Notes sessions at the same time. The maximum number of threads/Notes sessions depends on the hardware CSLD is running on.

retrieve documents to original database only
When this flag is set to **YES**, all documents retrieved via single retrieve or queries may be retrieved only to the database from which they were archived.

Note:
To use this feature, the retrieved documents must have been archived with CSLD, and with this flag set to **YES**. Further, the attributes *CSLDOrigDB* and *CSLDOrigUser* must be defined for all index classes or application groups from which documents are retrieved.

retrieve documents by original user only
When this flag is set to **YES**, all documents retrieved via single retrieve or queries may be retrieved only by the user who archived them.

Note:
To use this feature, the retrieved documents must have been archived with CSLD, and with this flag set to **YES**. Further, the attributes *CSLDOrigDB* and *CSLDOrigUser* must be defined for all index classes or application groups from which documents are retrieved.

Job Database Name
This field is used to specify the file name of the job database. The CSLD task will periodically check this database for jobs. The file name

is relative to the Notes data directory. For example, if the absolute path name of the job database is `E:\lotus\notes\data\cslddatabases\CSLDJobs.nsf`, specify `cslddatabases\CSLDJobs.nsf`.

Allow only requester to read result documents.
When set to **yes**, documents retrieved from the archive will be a protected from other users via a Readers field. That is, only the requesting user and the CSLD User ID will be able to see the retrieved documents.

Additional readers for result documents
When a users/group/server is given in this field, retrieved documents will be protected from other users by a Readers field. The readers field will contain the given user list as well as the user retrieving the document(s). You must copy the users/groups/servers from your address book, otherwise Notes will not be able to resolve the entries. Do not type entries manually, even when they look the same as the entries selected from the address book.

Job Database Server
This field is used to specify the name of the server with the job database on it. Must be listed in the public address book.

> **Important:**
> The server name must be given in abbreviated Notes syntax name/Domain, for example: `BOEDOM1/IBM_IDE`. Otherwise, this task will ignore all jobs.

Display query results as single hitlist or multiple result documents
This field is visible only when **Retrieval** is selected. When **single hitlist** is selected, the result of a query will be a single hitlist document containing all the hits in a table. When **multiple result documents** is selected, a query will return a number of result documents, each representing a hit. The result documents are simple documents displaying only the index information of the document in the archive.

Polling Parameters
These parameters schedule a CSLD task to search the job database for jobs. When not scheduled, the task falls asleep and frees system resources.

Polling Interval
Start and stop polling time. The CSLD task will process jobs within this time interval, only. If you have many tasks running at the same time on the same machine, you should,

for performance reasons, make sure that CSLD tasks do not run idle for a long time. For example: It is unlikely that users create retrieval jobs after midnight, so values like ″6 am till 11:59 pm″ might be reasonable for retrieval jobs. If a database is to be archived once a day at midnight, and archiving takes one hour, enter ″00:00 am till 02:00 am″. If you want to poll 24 hours a day, do not enter 00:00 am till 12:00 pm, because these two times are the same, so the time interval is zero. Instead, enter 00:01 am till 11:59 pm Note that jobs are always completed, even when a two hour archiving job is started one second before the polling stop time. Jobs are completed even if the task is shut down.

Polling Frequency
This field is used to specify the number of seconds, minutes, or hours the CSLD task sleeps before it looks for the next job. For minimum response times (e.g. for retrieval tasks), enter `one second`. Note that every job lookup requires a Notes database search, so please do not waste resources by specifying unnecessarily low values. See “Chapter 2.5 Performance” on page 29 for tips on how to improve system performance.

Days of Week
This field is used to specify the weekdays the CSLD task is scheduled to work.

Export Directory
This field is used to specify the directory for temporary files. CSLD tasks and the CMCS server exchange files by writing them to the directory specified in this field. The file system containing this directory should be big enough, should not be a shared or mounted network file system, and should not contain system swap files. To avoid difficulties, every CSLD task should have its own export directory.

Task Port Number
This field is used to specify the number of the TCP/IP port at which the CSLD task will listen for shutdown requests. Can be any valid, non-reserved TCP/IP port number. You cannot assign a single port number to two different tasks.

CommonStore TCP/IP Port
The port configured via the *DOMINOPORT* parameter in the CMCS configuration file `archint.ini`. Can be any valid, non-reserved TCP/IP port number.

CommonStore Host Name
The IP name of CMCS server

Tracing level

When set to **None**, no trace information is written to a file. When set to **Errors Only**, only error information is written to the trace files. When set to **All**, all available trace information is written to a file

Maximum trace and log file size

The maximum size of all trace and log files in KB. When this size is reached, the file is truncated. We recommend setting this value to at least 2 MB.

CommonStore Web port

Port of the CommonStore Web dispatcher. Used for browser viewing only. The Web dispatcher is basically a HTTP server that retrieves documents from the archive and sends them to the browser. When an attachment is archived, it is replaced by an URL. The URL of an archived document is also displayed in search histlists. The http port specified here will be part of this URL. The port specified here must be identical with the WEBPORT parameter in file archinit.ini of the CommonStore server. If you have only one CommonStore server (archpro instance) running on your machine, just leave the default (**8085**).

State Fields:

Determines how the state of archiving is written to the archived Notes document. The existence of a status field allows the application to visualize the document's state in a view or a form. In the mail archiving demo application, a successfully archived document is display with an icon depending on the archiving format. A document that could not be archived is displayed with a red bullet.

Two modes are available. In both modes, for every archived document content, one archive document ID will be written to the CSNDArchiveID field of the Notes document. For example, if five attachments of a document are archived successfully, CSNDArchiveID will contain five document IDs. For documents archived in native format, RTF, ASCII or TIFF, CSNDArchiveID will contain exactly one document ID, as the whole document is converted to one content file.

- **Write state to CSNDArchiveID field (CSLD 2.1 behavior):**

  When this mode is selected, the CSLD task will work as in CSLD version 2.1. When archiving fails, CSNDArchiveID is set to the word "Error". Previous values of CSNDArchiveID are overwritten. That is, success is determined by checking that CSNDArchiveID has a value other than "Error". This mode is less sophisticated than *Write state to special field* and is provided for backward compatibility.

- **Write state to special field:**

Allows writing the archiving state to a custom field. When archiving succeeds, CSLD will write the string entered in field *Success string* to the item entered in field *Status field name*. When archiving fails, CSLD will write the string entered in field *Failure string* to the item entered in field *Status field name*. For example, CSLD could be configured to write the strings ″Success″ and ″Error″ to field *CSLDArchivingState* of the archived document.

Note: In order to use document rearchiving (see “Chapter 3.2 Archiving options” on page 47 for details), this mode must be enabled. Otherwise, in CSLD 2.1 mode, when rearchiving of a document fails, all entries in CSNDArchiveID would get lost as they would be overwritten by the word ″Error″.

For users starting archiving documents with CSLD 7, we recommend using this mode, as it provides more control by the user.

Trace file directory
Trace files will be written to the given directory.

Log file directory
Log files will be written to the given directory.

Important:
CSLD profiles are read in during CSLD task startup. When a profile is modified, the running CSLD task must be shutdown and restarted.

Defining document mappings

Document mappings are defined by creating a document of the **Create->CSLD Document Mapping** form. You can also change to the **all Document Mappings** view and click the **New Document Mapping** button.

The document mapping dialog contains the following fields:

Notes Form name
All documents with this form will be mapped to the archive ID given in the CMCS archive ID field.

Notes Form Aliases
Enter all aliases for the form given in the Notes form name field, separated by a comma. For example: In order to archive e-mails (documents of the form **Memo**), enter **New Memo**, **Document**.

CommonStore Archive ID

Name of CMCS archive ID (as defined in the CMCS configuration file `archint.ini`) to which Notes documents are archived.

Representative Attributes

When users perform queries, they retrieve hitlists. In order to distinguish the different entries of a hitlist, some information about the entries must be given. So for each entry, a number of fields are displayed that are regarded as ″representative″ for that document. One would probably choose e.g. **Artist** and **Album Name** as the representative fields of an **AudioCD** form. The CSLD administrator decides which fields of a Notes form are representative by filling in the field names of representative attributes, separated by a comma.

Due to the maximum width of Notes documents, the maximum number of displayable representative attributes in a hitlist document is limited to 10. The order in which the attributes are entered in the **Representative Attributes** field reflects the order in which they will be displayed in a hitlist. This does not apply to queries returning hitlists as single result documents, as these documents contain all mapped attributes rather than the representative attributes.

Form used to display documents

Archived documents of the form given in the form name field are displayed in this form. This field is only used for retrieving rasterized documents and attachments. When documents archived in the Notes native format are retrieved, they are displayed in their original form (provided that it still exists in the target database).

Pre-archiving agent

Name of a pre-archiving agent (see “Chapter 3.4 Agents for processing archived and retrieved documents” on page 55) that is invoked before a document is archived. The agent is only invoked for documents of the form given in the *form* or *aliases* fields.

Post-archiving agent

Name of a post-archiving agent (see “Chapter 3.4 Agents for processing archived and retrieved documents” on page 55) that is invoked after a document has been archived. The agent is only invoked for documents of the form given in the *form* or *aliases* fields.

Post-retrieval agent

Name of a post-retrieval agent (see “Chapter 3.4 Agents for processing archived and retrieved documents” on page 55) that is invoked after a document has been retrieved. The agent is only invoked for documents of the form given in the *form* or *aliases* fields.

The following figure presents a typical document mapping for an **AudioCD** Notes form.

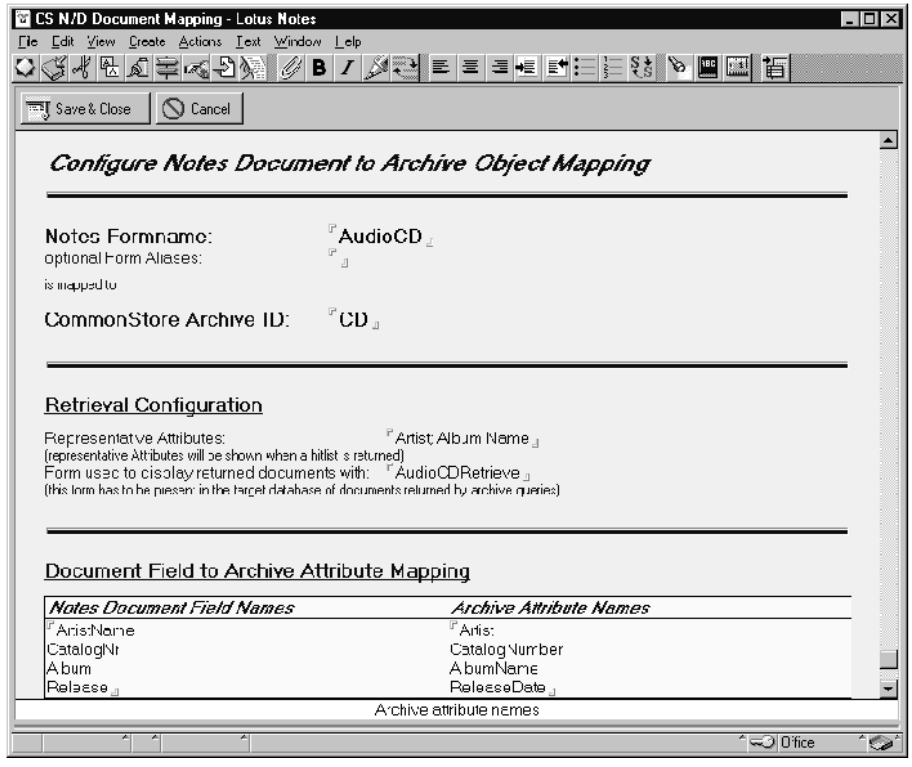

Figure 2. Typical document mapping for an AudioCD Notes form

In this simplified example, the administrator has defined a Content Manager index class named **CD**, consisting of the attributes **Artist**, **CatalogNumber**, **AlbumName**, and **ReleaseDate**. To make this index class known to CSLD, he defined an archive ID `CD` in the CMCS configuration file `archint.ini`. The corresponding Notes form is named **AudioCD**, and defines the fields **ArtistName**, **CatalogNo**, **Album**, and **Released**. When an AudioCD Notes document is archived, the content of the field **ArtistName** is stored in the Content Manager attribute **Artist**, and so on. Since the artist name and the album name of a CD are probably the most descriptive attributes of an audio CD, they are defined the representative attributes of the Notes form **AudioCD** and will be shown in query result lists. When rasterized Notes documents or attachments are retrieved from the index class **CD**, they are displayed with the form **AudioCDRetrieve**. This form has been created by the administrator and is stored in the user database.

Important:

- You can specify only one document mapping for a form. However, you can define one document mapping and a number of special mappings for one form (see "Defining special mappings").
- Document mappings are read in during CSLD task startup. When new document mappings are added, all running CSLD tasks must be re-started.
- Notes form names specified in the document mappings will be treated case sensitive.

Defining special mappings

Special mappings are defined by creating a document of the **CSLD Special Mapping** form. You can also change to the **All Special Mappings** view and click the **New Special Mapping** button The special mapping definition dialog contains the following fields:

Notes form name
: Name of the Notes form for which the special mapping is defined. Documents of this form will be archived to the archive specified in the **CommonStore Archive ID** field if they fulfill the special mapping criteria.

CommonStore Archive ID
: The CMCS archive ID as defined in the CMCS configuration file `archint.ini` to which documents fulfilling the special mapping criteria are archived.

Notes field name
: Name of the field which is checked for fulfilling the special mapping criteria. This field must be defined in the form given in the **Notes form name** field above.

Notes field value type
: Notes value type of the field given in the **Notes field name** field. You can choose between "text", "number", and "date/time." The value type given here must be the same as in the form containing the field. If you enter a wrong value type, an error message will be generated during archiving. We recommend that you always check the form in the user database. For example: If the special mapping defines the **MusicStyle** field as the field to evaluate and **MusicStyle** is of the Notes data type "text," you must select "text." This is necessary because the CSLD configuration database has no information about forms stored in other databases.

Exact value or value range selector

Specifies whether the special mapping is based on an exact value or on a value range. When the mapping is based on an exact value, a ″text″ field will appear to fill in the exact value. When the mapping is based on a value range, two fields for the lower limit and the upper limit of the value range will appear.

Values

When **Exact value** is selected, enter the value to be used in comparing document fields. When a document being archived has the same value, it is archived to the destination given in the **CommonStore Archive ID** field. When **value range** is selected, enter the minimum and maximum value for the value range. When a document being archived has a value between the minimum and maximum value, it is archived to the destination given in the **CommonStore Archive ID** field.

Important!

- *Make sure you have entered the correct type by looking into the original forms*. If you specify a different type than in the form, an error message will be generated.
- *Be aware that the Notes application template designers could have changed the type of a field used in a special mapping without informing the CSLD administrator.*
- *Special Mappings cannot be based on multi-valued or RTF fields.* When a multivalued field is used, CSLD interprets only the first value.
- *The CMCS archive ID defined in special mappings always overwrite the CMCS archive ID of regular document mappings!* Thus, when a document fulfills the criteria defined in a special mapping, the document mapping for the document's form is ignored.
- Special mappings are read in during CSLD task startup. When new special mappings are added, all running CSLD tasks must be re-started.

Defining sophisticated special mappings

You might find that an exact value or a value range are not flexible enough for sophisticated special mappings like ″Store all classical audio CDs of Debussy and Ravel in index class named 'Impressionism'″. In these few cases, you can apply higher-level logic within the Notes application. For example, you can add Notes formulas or Lotus Script code to perform complex field computations. The result can be written to a special field which can then be used in the special mapping dialog. One possible implementation for the above scenario would be to define an additional computed field **MusicStyle** in the form that would be calculated by the Notes formula

```
@if(Artist = "Ravel" | Artist = "Debussy"; MusicStyle = "Impressionism";)
```

You would then define a special mapping based on the value of the **MusicStyle** field.

Defining content type mappings

The figure below presents a sample content type mapping.

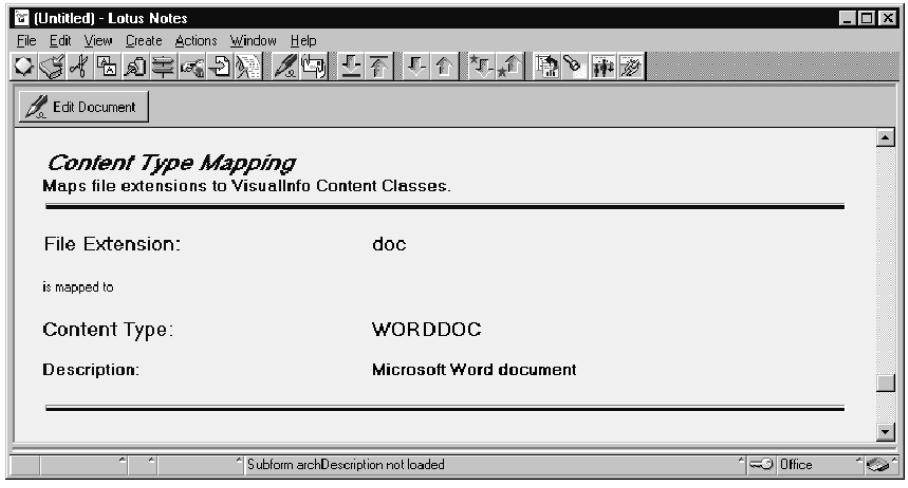

Figure 3. Sample content type mapping

Creating content type mappings is straightforward: From the Notes menu, select **Create->CSLD Content Type Mapping** or go to the **all Content Type Mappings** view and click the **New Content Type Mapping** button. Then, enter a file extension and content type name. Make sure that the content type you enter really does exist. Of course, you can map many different file extensions to the same content type. However, you cannot enter more than one file extension in one mapping dialog.

Note:
File extensions are case-insensitive. That is, you do not have to define a mapping for both `.doc` and `.DOC`. However, Content Manager content types are case-sensitive. We recommend that you always write content types in upper case.

Also, note that the file extension must not contain the leading dot. For instance, for file extension `.txt`, simply enter `txt`. For files with the extension

`.tar.Z` (a common UNIX format), enter `tar.Z`. For more information on content types, see "2.3.3 Content types and filenames" on page 24.

Important:
Content type mappings are read in during CSLD task startup. When new content type mappings are added, all running CSLD tasks must be shutdown and re-started.

The Example Documents view

The CSLD setupDB tool, shipped with CSLD, allows the creation of a simple default query form as well as a form for displaying retrieved documents (if the documents have been archived by means of attachment archiving or rasterizing) for a given form. To create fields with the right data type, setupDB needs one example document for the form it is called for. If you want to call setupDB for form **A**, simply drag and drop one document of form **A** into the configuration database. The setupDB tool will then use the **Example Documents** view to find that document. The document must contain all items defined in the document mapping for the form. There is no need to copy the document's form into the configuration database.

Chapter 7.3 Command Line Administration

7.3.1 Starting and stopping CSLD tasks

CSLD archiving and retrieval tasks are both started by launching the `csld.exe` executable. `csld.exe` must be executed with the following parameters:

-n This parameter specifies the file name of the CSLD configuration database. The file name is relative to the Notes data directory on the server.

-s This parameter is used to specify the name of the server with the CSLD configuration database on it.

-p This parameter is used to specify the CSLD task profile name. Used to find this task's parameter in the CSLD configuration database. See "Defining profile documents" on page 125 for details on profiles

-shutdown
This parameter is optional. When given, it stops the CSLD task having the given profile name. In other words, to shutdown a CSLD task, use the command line in which the task was started and append *-shutdown*.

-i notesinifile
This parameter is optional. When given, the CSLD task runs under the ID specified by the Notes ini file named `notesinifile`. When omitted, the ID file specified in `notes.ini` is taken. You will be prompted for a password.

-f serverpasswd
Specify this parameter to provide a password for the currently active Notes userID. The only other parameter allowed in combination with this one is the `-i parameter`. The password will be stored in encrypted format to file csld.cfg, from which CSLD will read it the next time it starts, instead of bringing up a password prompt.

Note that the information whether a CSLD task is an archiving task or a retrieval task is provided in the task's profile in the CSLD configuration database.

When a CSLD task is shut down, all pending jobs are completed before the task is stopped. This can take a while, but is necessary to ensure the consistency of archiving applications. When the task is currently sleeping (not polling for jobs), shutting down can take up to 30 seconds. You should not

stop CSLD tasks by killing the processes, because this can leave the entire Notes runtime in an unpredictable state.

7.3.2 Starting CSLD tasks without typing in Notes passwords

In CSLD 2.1, when CSLD tasks were started, users had to explicitly type in the Notes password for the ID the task is running under, or they had to leave the password empty. In CSLD 7, Notes passwords can be stored so that CSLD tasks will no longer display the password prompt. Instead, the task will read the password from the password file *csld.cfg*. This does not imply a security hole as the passwords in the file are protected with a sophisticated encryption algorithm.

In order to use this feature, add the line

```
EXTMGR_ADDINS=CSLDExtPwd.dll
```

to your notes.ini file. At the next Notes startup, this will launch the CSLD extension manager password plugin. If the CSLD task is started with a different .ini file using the -i parameter, make sure it contains the line above. The file *CSLDExtPwd.dll* is installed under the CSLD binary directory. This directory is contained in the PATH environment variable, so that the DLL can be found by the operating system. The CSLD task first checks under which ID it is running. Then, it checks the password file if it contains a password for the current ID. If not, it prompts the user to type in the password, and stores it encrypted in *csld.cfg*. Note that the password file can contain more than one password because a CSLD task can run with different IDs. Suppose you have changed the password for a certain ID in Notes, but the password file still contains the old password. To manually set or overwrite a password without starting a task, use the command

```
csld -f  serverpasswd [-i <inifile.ini>]
```

This prompts you for a password and stores it in *csld.cfg*. The word *serverpasswd* is not a placeholder for the actual password. Type in the command exactly as given above. If the optional parameter <inifile.ini> is given, CSLD reads the ID from the given file. Otherwise, the default *notes.ini* file is used. Note that this command works the same as the *archpro -f serverpasswd* command, just with the difference that *csld -f serverpasswd* sets Notes passwords, and *archpro -f serverpasswd* set archive passwords. To erase all stored passwords simply delete file *csld.cfg*.

Note that CSLD uses the Notes extension manager mechanism to set passwords. This means that if you have other Notes applications running on your CSLD server that would prompt for a password, they would read the

password from the *csld.cfg* file. You can even use the *CSLDExtPwd.dll* and *csld.cfg* file together with CSLD 2.1 to prevent the password prompt.

7.3.3 Logging and tracing

CSLD tasks log all errors and events like startup or shutdown to the file `<profilename>.log`, where `<profilename>` is the name of the task's profile. The error logs include date and time, and are identical to the error messages written to the jobs that were executed while the error occurred. Once in a while the size of this file should be checked. In addition, all errors during archiving, retrieval, update or deletion are written to job documents in the job database.

When tracing is turned on in the profile, all trace information is written to file `<profilename>.trace`.

Chapter 7.4 Error handling

CSLD tasks run in three phases:

1. When CSLD tasks are started up, all configuration information is read in from the configuration database. When an error condition is encountered during configuration reading, the task aborts with an error message, which is logged to the `<profilename>.log` file. The error message is also printed on the screen.
2. When the connection cannot be established, the task aborts with an error message.
3. When a CSLD task is running, all errors during archiving, retrieval, updating, or deletion are written to the job error field.

When error conditions are encountered that require actions by the CSLD administrator, the job containing the error message is mailed to the administrator. Examples for such error conditions are as follows:

- Full disks, or disk crash, so that CSLD cannot create temporary files.
- The value type of a field in a special mapping is not the same as the field value type in the form. The administrator must check the document form.
- CSLD tries to archive a document whose form has not yet been mapped in a document mapping.

Most errors during archiving are mailed to the CSLD administrator(s), if configured.

At most five error notification mails are sent to the administrator for a single repeating error condition. For example: When a CSLD task cannot poll the job database for jobs every ten seconds, a maximum of five error mails are sent to the CSLD administrator.

Chapter 7.5 The Raster Exit DLL

The Raster Exit consists of a single function that must be provided through a C-DLL named `CSLDRaster.DLL`. Using this interface CSLD will call the external rasterizer functionality. The interface has the following appearance:

```
int _cdecl _Export

RasterizeNote(    HANDLE         hNote,
                  HANDLE         hDB,
                  unsigned long  ulRasterFormat,
                  unsigned long  ulRasterOptions,
                  unsigned long  addtlFlags,
                  char           *pszOutfile,
                  void           *pHook,
                  char           *pszErrText );
```

The function takes the C handle to the note that needs to be converted as well as the C handle to the database this note resides in. CSLD also passes the format to which the given note should be converted, an option as to which parts and how to convert it and some additional flags telling if and which extra-processing to perform on the note. Further, the complete path name under which the file containing the converted note must be created and a character buffer to which the rasterizer can store error information are passed in by CSLD.

Upon return an integer return code taking some predefined values is expected to be returned by the exit.

Input parameters

HANDLE hNote:	C handle to Notes document to be rastered
HANDLE hDB:	C handle to Notes DB containing above note
unsigned long ulRasterFormat:	constant defining the raster format. CSLDRaster.h defines symbols for allowed values. For the time being only RASTER_TIFF_FORMAT is supported. However, this list can easily be extended.

unsigned long ulRasterOptions:	constant defining an additional option as to what to raster. CSLDRaster.h defines symbols for allowed values. Possible values are: RASTER_NOTE_APPEND_ATTACHMENT - raster both, note and attachments, then append the attachments behind the note. RASTER_NOTE_EMBED_ATTACHMENT - raster both, note and attachments, embed attachments at the position they were attached at. RASTER_NOTE_ATTACHMENTS_ONLY - raster only the attachments of the note, but not the note itself.
unsigned long addtlFlags:	additional set of Ored together flags determining how to raster the document. CSLDRaster.h defines symbols for allowed values. Possible values are: RASTER_ROTATE - if an attached image is wider than high, rotate it to fit the note's width. RASTER_TRIMWHITE - trim leading and trailing whitespaces in note and/or attachments. For the time being the flags are hard-wired to both RASTER_ROTATE | RASTER_TRIMWHITE
char *pszOutfile:	character buffer containing the fully qualified path to output file. Upon return CSLD expects that the rasterized note is found in this file.
void *pHook:	reserved for future use.
char *pszErrText:	character buffer allocated with MAX_MSG_LENGTH (as defined in CSLDRaster.h) Raster exit can fill in an optional error text, if available. This error text will then be printed in the job status field in case rasterizing fails.

Output parameters

int ulRasterRC:	return code from rastering. CSLDRaster.h defines symbols for possible return codes. Based on this return code, CSLD will decide whether rasterizing succeeded or not.

Raster Exit implementation with Compart DocBridge

The first implementation of the CSLD Raster Exit is provided using the Compart DocBridge product. In fact, the Raster Exit was designed and developed in close cooperation with Compart. Compart DocBridge consists of a printer driver and an API DLL. Notes documents will basically be printed to a file that CSLD will then transfer to the archive.

To enable rastering with DocBridge, the product has to be installed and configured. CSLD comes with a DLL named `_CSLDRaster.DLL`. Once Compart DocBridge is working properly, this DLL must be renamed to `CSLDRaster.DLL` (remove the underscore "_"). CSLD will then call this DLL which in turn makes use of the DocBridge APIs to implement the rasterizing functionality.

Note

Although the Raster Exit DLL calling Compart DocBridge comes with the installation of CSLD, DocBridge itself is not part of CSLD and has to be purchased and installed separately.

For further information on Compart and the DocBridge product please contact Compart at:

Mail:	armin.groeger@compart.net
Phone:	(+49) 7031 - 62 05 23

For further information on how to install and configure the DocBridge product please refer to the DocBridge documentation.

Chapter 7.6 The User Exit DLL

The user exit consists of a single function that needs to be provided to CSLD in a C DLL. This C DLL must be named CSExit.DLL and must be loadable by the CommonStore server. The function interface has the following appearance:

```
int GetArchUser( const char* pszNotesUserName,
                 char archUserID[SIZE_VIUSER],
                 char archUserPasswd[SIZE_VIPASSWD] );
```

The function takes the Notes user name in full canonical format (e.g. CN=John Doe/OU=Boston/O=USA) in a character array as the input parameter. Character buffer pszArchiveUserID allocated with a length of SIZE_VIUSER is passed to the function and is expected to contain the archive user ID on return. In a second character buffer pszArchiveUserPasswd with length SIZE_VIPASSWD, the archive user's password is expected to be returned. Make sure that the buffer sizes SIZE_VIUSER and SIZE_VIPASSWD (which you must define yourself) are big enough to hold the longest user name/password as well as the ending null byte.

The function body has to be provided by the user and might, for example, include a database query where all Notes users are mapped to the respective archive users. But however the logic of this function is implemented, it must be pointed out that it might have a major impact on the performance of the archive functions that call it, since it is called for every single request.

There are different ways of implementing the aforementioned function:

- by doing a lookup in a simple Notes database view using the Notes C API
- by reading all database entries into memory at the first request
- by reading mappings from a file and writing them into a hash table structure in memory

Part 8. CSLD Programming Guide

Chapter 8.1 Creating job documents

Making archiving or retrieval requests to CSLD is done by creating so-called *job documents* in the CSLD job database. The job database is the only interface for users to the CSLD processes. Jobs are protected by signing them and including a **Readers** item containing only the job's author and the CSLD users (see "Chapter 2.7 Security" on page 33). This way, no other user can copy information from another job and possibly access documents he/she would normally not be allowed to access.

The following sections describe the fields that have to be set in order to generate a valid CSLD job. Alternatively to writing code that creates such a job document itself and fills in the required fields, CSLD also provides users with a set of script classes that can be used to simplify job creation. The structure and usage of these script classes will be discussed in a separate chapter.

General job parameters

In addition to a set of general parameters, each job document includes some job-specific or request type-specific parameters which must be set in order for the job to be processed correctly by the CSLD processes.

In addition to the fields in a CSLD job, each job document has to be signed in order to identify the requester of the job. Unsigned job documents will be rejected by the CSLD processes.

Further, it is recommended that all job documents be read-protected so as to allow only the job's author and users that have role [CSLDUsers] to read the job documents stored in the job database.

The general fields found in every job in order for it to be processed, no matter what the request type is, are listed in the following table.

Table 4. General fields

Field name	Data type	Usage
requestType	Number	This field determines what type of request the given job will be. The job-dependent fields are determined on the basis of this field. CSLD defines a set of constants, each of which stands for a particular request type. Available request types are: • CSN_ARCHIVE_ATTACHMENTS (attachment archiving) • CSN_ARCHIVE_NATIVE (native archiving) • CSN_ARCHIVE_TIFF (choosing this request type will result in CSLD calling the Raster Exit. This exit will then be responsible to convert the Notes document according to an also given raster option (see field ″rasterOptions″ in Archive Job)) • CSN_ARCHIVE_ASCII_FORMAT (conversion of all document content into a plain text file) • CSN_ARCHIVE_RTF_FORMAT (rasterizing of the Notes document to a Windows RTF file) • CSN_DELETE_BY_ID (deletion request for an archived document) • CSN_UPDATE_INDEX (index update of an archived document) • CSN_REQUEST_BY_ID (single retrieve of an archived document on the basis of its archive ID) • CSN_QUERY (archive search request) • CSN_MOVE_TO_WORKBASKET (moves an archived document to a workbasket) • CSN_REMOVE_FROM_WORKBASKET (removes an archived document from a workbasket) • CSN_LIST_WORKBASKET (lists all documents in a workbasket) • CSN_RESTORE_FOLDER (restores an archived Notes folder based on an also given name or archive document ID)
deleteJob	Text/flag	This field determines whether or not the job document will be automatically deleted from the job database when processing of all documents in the job has been successfully completed. Expected values are ″yes″ or ″no″.

Table 4. General fields (continued)

Field name	Data type	Usage
jobState	Number	This field specifies the numerical value type code that stands for the current job's state. CSLD defines a set of constants representing the possible states: 1. CSN_JOB_TOBEPROCESSED: Initial state of the job. 2. CSN_JOB_INPROCESS: CSLD is now working on the current job. 3. CSN_JOB_SUCCESS: Processing for all documents in the job has been successfully completed. 4. CSN_JOB_ERROR: An error occurred during the processing of one or more of the documents in the job. More detailed information can be found in the job info field. When a job document is created, the job is expected to be in the "waiting to be processed" state. Only job documents in this state will be processed by the CSLD processes.
bodyJobInfo	RTF	This field is used by the CSLD processes to provide more-detailed information on the job state. In the event of the failed processing of documents contained in the job, the system will add an extra line with a detailed error message for each document.

Archiving

When building up an archiving job, information needs to be passed about the document or documents to be archived and the manner in which this should be done.

Archiving requests can include requests for:

- the archiving of all the attachments in the given document(s) in their respective formats;
- the archiving of the document(s) in Notes native format, i.e. in a bytestream format that can, upon retrieval, be restored to that exact Notes document;
- the archiving in Microsoft Rich Text Format, i.e. rasterizing the given Notes document(s) to a RTF image that can be restored as a file attachment of type RTF;
- the archiving in ASCII format, i.e. converting the Notes document into a plain ASCII text file that can be restored as a file attachment in TXT format.
- the archiving in another format using the Raster Exit, thereby calling external rasterizing functionality;
- moving a document to a workbasket after archiving.

Documents to be archived are selected by passing their UNID to CSLD. Up to 1200 documents can be defined in a single archive job. Therefore, not only single documents, but also complete views or the contents of a folder can be stated in a job. If CSLD encounters a UNID representing a folder or view, it will automatically apply the parameters set in the job to all of the documents contained therein. Alternatively to archiving all documents from within a given view or folder, it can also be requested to archive an entire Notes folder structure preserving that structure in the archive.

Other archiving job parameters include whether or not to delete the original document from the Notes database after it has been successfully archived (**deleteOriginal**), whether or not to remove archived attachments from their originating documents (**deleteAttachments**), and whether or not to delete the job document itself after it successfully completes processing (**deleteJob**).

Archiving job fields

If the **requestType** field is set to CSN_ARCHIVE_RTF_FORMAT, CSN_ARCHIVE_ASCII_FORMAT or CSN_ARCHIVE_TIFF_FORMAT, in addition to the general fields, the following fields must be also defined:

Table 5. Required fields for archiving requests

Field name	Data type	Usage
docUNID	Text, multi-valued	The UNIDs of all of those documents whose archiving is defined in the current job. This field may contain a maximum of 1200 document UNIDs. Alternatively, an UNID stored in this field may also identify a view or folder.
keepFolderStruct	Text	optional field. Will only be evaluated if the UNID given in the ″docUNID″ field refers to a folder. Expected values are ″yes″ and ″no″. If not available or set to ″no″ CSLD will archive all documents residing in the given folder. If set to ″yes″ CSLD will archive the entire folder structure.
rasterOptions	Number	will only be evaluated when the request type is set to CSN_ARCHIVE_TIFF_FORMAT. Then, this field can be used to specify which parts of the Notes document should be rastered. CSLD defines a set of constants, each of which stands for a particular raster option. Available values are: 1. CSN_RASTER_NOTE_EMBED_ATTACHMENTS (rasterize the document and its attachments. Attachments will be rastered at the position they were attached to) 2. CSN_RASTER_NOTE_APPEND_ATTACHMENTS (rasterize the document and its attachments. Attachments will be rastered at the end of the document) 3. CSN_RASTER_NOTE_ATTACHMENTS_ONLY (only the document's attachments will be rastered. All attachments will be rastered into a single output file)
sourceDB	Text	Specifies the path name of the database in which the aforementioned documents are located.
sourceSrv	Text	Specifies the name of the server on which the originating database resides. The value of this field in combination with **sourceDB** is used by the CSLD processes to locate the originating database for documents to be archived. CSLD archiving processes are configured to run on a number of source databases, so the selection of job documents one CSLD archiving process is responsible for is done based on the value of this field in combination with **sourceDB**.
deleteOriginal	Text	Determines if the original document will be deleted from its database once archiving has been successfully completed. Expected values are ″yes″ and ″no″.
deleteAttachments	Text	In the case of attachment archiving, this field determines whether or not the archived file attachment(s) will be automatically removed from the originating document. Expected values are ″yes″ and ″no″.
workbasketName	Text	If this field is given and non-empty, the document will be archived to the workbasket with the given name (CM and OD only).

Document updating

Once a Notes document is archived, it can be updated in three different ways:

- Index updating: The index of a document in the archive is updated with the field values of the corresponding Notes document. The request type is CSN_UPDATE_INDEX.
- Moving a document to a workbasket: Membership to a workbasket is a property of an archived document. Therefore, in CSLD, moving documents to a workbasket is handled as a document update. The request type is CSN_MOVE_TO_WORKBASKET.
- Removing documents from their current workbasket: Membership to a workbasket is a property of an archived document. Therefore, in CSLD, removing documents from their current workbasket is handled as a document update. The request type is CSN_REMOVE_FROM_WORKBASKET.

It is not possible to mix case 1 and 2, i.e. you cannot perform an index update in the archive and move a document to a workbasket with one request. All three operations are handled by update jobs. Every update job must contain the following fields:

Document update job fields

If the **requestType** field is set to CSN_UPDATE_INDEX, in addition to the general fields, the following fields must also be defined:

Table 6. Required fields for request types CSN_UPDATE_INDEX, CSN_MOVE_TO_WORKBASKET and CSN_REMOVE_FROM_WORKBASKET

Field name	Data type	Usage
sourceDB	Text	The file path of the Notes Database containing the documents used to update (an) archived document(s).
sourceSrv	Text	The server name of the server on which **sourceDB** resides. The value of this field in combination with **sourceDB** is used by the CSLD processes to locate the originating database for update documents.
docUNID	Text, multi-valued	The document UNIDs of the documents to update. The CSLD processes expect these documents to include a field named **CSNDArchiveID** containing the ID of the archived document that should be updated
workbasketName	Text	When this field is given in an update job, the update will move the document to the workbasket with the given name. It is not possible to move the document to a workbasket on a different archive server. Delete and rearchive the document in this case. For request type CSN_REMOVE_FROM_WORKBASKET, do not specify the workbasketName field, as the documents are removed from their current workbasket.

Deletion

With CSLD, the only way to delete an archived document is on the basis of its archive document ID. Thus, the only parameter needed in a deletion job is this ID. Again, one or more archive document IDs can be stated in the job.

Optionally, the UNID of a document containing the link to the archive document may be stated. This is especially the case when dealing with attachment archiving. If the UNID is specified, CSLD will also remove the link to the archived document from the originating document once the archive document has been deleted.

Deletion job fields

If the **requestType** field is set to CSN_DELETE, in addition to the general fields, the following fields must also be defined:

Table 7. Required fields for request type CSN_DELETE

Field name	Data type	Usage
sourceDB	Text	Specifies the path to a working database.
sourceSrv	Text	Server name on which **sourceDB** resides. Must be given in abbreviated format, otherwise CSLD will not process the job
deletionIDs	Text, multi-valued	This field contains the archive document IDs of all the documents that should be deleted
docUNID	Text	The UNID of a Notes document containing a reference to the archive document to be deleted (i.e. it contains the archive document ID in a field named **CSNDArchiveID**). If set, the CSLD process deleting the archive document specified in the job will delete the reference to this archive document from the given Notes document, too. This parameter is an optional one. It will be considered only if the **deletionIDs** field contains only one archive document ID.

CSNDSpecificJob uses subform **DeleteByID** to display deletion requests. The additional fields defined by this subform are described in the following table.

Table 8. Additional fields defined by subform DeleteByID for browsing the job document

Field name	Data type	Usage
numDelete	Number	The number of archive document IDs included in the delete job. Computed for display based on the number of values in the **deletionIDs** field.
removeLink	Text	Flag telling whether or not the reference to the deleted archive document will be removed from the referencing Notes document. Computed for display based on the availability of the **docUNID** field. Possible values are ″yes″ and ″no″.

Retrieval

In CSLD, there are five ways to retrieve documents from the archive:

- Retrieving a single document by id: If the ID of the archived document is known (i.e. taken from the CSNDArchiveID field of an archived Notes document or from a hitlist), the document can be retrieved.
- Archive search: A search in the archive returns all documents that match a certain search criteria.
- Listing the documents in a workbasket: All documents in a workbasket are returned as a hitlist or multiple result documents

- Listing the content of a Content Manager folder: Besides regular documents, a search can also return folders. Clicking the ″Open″ button in a hitlist will return the folder content in a second hitlist (or as multiple result documents).
- Restoring a Notes folder or folder structure: Archived Notes folder structures are restored to their original position, together with the documents in the folder structure.

The document created by CSLD when archive content is brought back to Notes will consist of the mapped fields containing the index information and an RTF field that has the document content appended as a file attachment. Optionally a retrieved document can be made a response document to another document in the Notes database. Since CSLD returns documents as well as archive folders in queries, this is especially feasible, when views need to be organized as to reflect that connection, e.g. users could create a type of tree structure by making the folder result document parent and all documents from it responses.

Using the setupDB tool, CSLD can provide a default form for browsing retrieved documents.

Single document retrieval

An archived document can be retrieved from the archive on the basis of the archive document ID it was given during archiving. If this cryptic and non-descriptive ID is somehow preserved, this is the fastest way of getting documents back from the archive to the Notes environment. In addition to the aforementioned possible target locations for single document retrieval, even a Notes document in a Notes database may be stated in the job. In this case, the document content will either be appended to a given RTF field within this document as a file attachment or it will be created as a stand-alone file attachment at the bottom of the document. The descriptive information for this archive document will be skipped.

Single document retrieval job fields

If the **requestType** field is set to CSN_REQUEST_BY_ID, in addition to the general fields, the following fields must also be defined:

Table 9. Required fields for request type CSN_REQUEST_BY_ID

Field name	Data type	Usage
archID	Text, multi-valued	This field contains the archive document IDs for all documents that are to be retrieved from the archive according to the parameters set in the job.
targetDB	Text	The path to the database to write the retrieved document to.
targetSrv	Text	Name of the server on which this database resides.
targetFolder	Text	This is an optional field in which the name of a folder within the given target database can be specified to which the resulting documents will be retrieved.
targetDocUNID	Text	The UNID of the Notes document to which the retrieved document content should be appended as an attachment. If this field is set, only the document itself (but not the retrieved index information) will be restored. If this field is set, **targetField** must also be set.
targetField	Text	This field contains the name of the RTF field to which the retrieved document content will be appended. It will be considered only if the **targetDocUNID** field is set.
treatNativeAsNew	Number	optional field. Expected values are 0 or 1. If set to 1, CSLD will create natively archived documents as new documents, i.e. without their original UNID. If not specified or set to 0, CSLD will restore natively archived documents exactly as they were before archiving, i.e. including their UNID.
parentDocUNID	Text	optional field. May contain the UNID of another Notes document within the target database. CSLD will then make the retrieved document a response to the given one. This is especially feasible if you want to implement categorized views, where e.g. an archive folder is parent to all documents residing in it.

Query jobs

The main parameter of a query job is the query string, that represents the query in an SQL-like syntax. This query string will be translated by CSLD into the correct archive query. The query string is made up of query predicates, combined together with **AND**, **OR**, or enclosed in parentheses. Each search predicate is made up of the field name enclosed in brackets, an operator (**<**, **>** ,**=**, **>=**, **<=**, or **like**) and a value. For example: Suppose that the Notes database is a catalog of music. As document content, a sample of a certain recording might be stored in the archive. An archive query looking for all Mozart recordings after 1985 conducted by either Leonard Bernstein or Herbert Karajan would have the following appearance:

```
[Composer] = "Mozart" AND [RecordingDate] > "1985-01-01" AND ([Conductor]
          like "%Bernstein" OR [Conductor] like "%Karajan")
```

The field names are Notes field names and will be replaced with the corresponding archive attribute names. The syntax itself with all combination operators and parentheses will be passed on ″as is″ to the archive, where the respective search engine will resolve the parentheses, do the logical checking and finally evaluate the query. Syntactical checks that will be done by CSLD include checking whether or not the 'like' operator is followed by a string search argument, whether or not all parentheses opened are closed again, and the position of field names, operators and values. The following field value types may be used in a search:

″text″ fields
: All ″text″ field values can be used as search arguments. The value itself has to be enclosed in double quotes in order to be processed, e.g. `"Mozart"`

″number″ fields
: ″Number″ field values can be used as search arguments 'as is'. No enclosure is needed. In the case of decimal values, make sure that a decimal point is used, for example `45.7`.

″date/time″ fields
: Just as in a Notes environment, ″date/time″ fields may be represented as ″date-only,″ ″time-only,″ or as a ″timestamp,″ i.e. a date followed by a time. In all cases, ″date/time″ values are transformed into a standardized string format and then enclosed in single quotes to be processed correctly.

  The correct ″date/time″ format is `yyyy-mm-dd` for dates, where `yyyy` is the year in 4-digits, `mm` is the month and `dd` the day of the month, both as two digits with possible leading zero. ″Time″ values will be represented as `hh.mm.ss`, with `hh` being the hours in 24-hour-representation, `mm` the minutes, and `ss` the seconds, all in two digits with possible leading zero. ″Timestamps″ will be represented by a date and a time value, connected with a dash, and nanoseconds in six trailing digits (the first three of the six digits are microseconds, the last three digits are always zero). The format is: `yyyy-mm-dd-hh.mm.ss.nnnnnn`.

Query job fields

In addition to the general job fields, the following fields must also be defined for a query job:

Table 10. Required parameters for query jobs

Field name	Data type	Usage
targetDB	Text	Specifies the path to a working database.
targetSrv	Text	The name of the server on which this database resides. **targetSrv** together with **targetDB** are used by the CSLD processes to locate the database in which documents returned by retrieval requests will be stored. For documents archived in the Notes native format, a new Notes document will be created in the database, all other archiving types will yield a result document consisting of the index information and the document content appended as an attachment.
targetFolder	Text	An optional field in which the name of a folder within the given target database can be specified to which the resulting documents will be retrieved.
treatNativeAsNew	Number	optional field. Expected values are 0 or 1. If set to 1, CSLD will create natively archived documents as new documents, i.e. without their original UNID. If not specified or set to 0, CSLD will restore natively archived documents exactly as they were before archiving, i.e. including their UNID.
CSNQryString	Text	The complete query string is stored to this field.
maxNumOfDirectHits	Number	The threshold defining up to how many of the documents found by an archive query will be built as a whole with their document content. Since an archive query might return a large number of hits depending on how exact search parameters were defined, this threshold allows to limit data transfer from the archive.
maxNumOfHits	Number	The threshold defining the maximum number of documents that will be returned from an archive query request. This number limits the total number of documents that will be displayed as a search result. If a query returns a number less than or equal to **maxNumOfHits**, **maxNumOfDirectHits** determines whether the documents returned will be built with their document content or whether a hitlist document will be generated that contains only a definable number of representative attributes. The user will then be able to retrieve single documents from the hitlist document.

CSNQueryForm

CSLD provides a default form named **CSNQueryForm** that can be used to create query requests for a particular document type (i.e. form). The form allows users to enter search criteria for each of the mapped attributes. It also defines an action that will take all the search predicates entered, combine them together with **AND** and automatically generate a query job in the job database. This default form can be created for every Notes document type mapped to the archive in every Notes database used as target database for query requests using the CSLD setupDB tool.

This form contains the document type for which the search arguments apply and for each of the mapped attributes a computed field with the field name, a list of relational operators and an entry field for the search argument. The script to create a query job from the search criteria entered is triggered by a form action.

The following example presents a query of all documents including the phrase 'Project' in the **Subject** field.

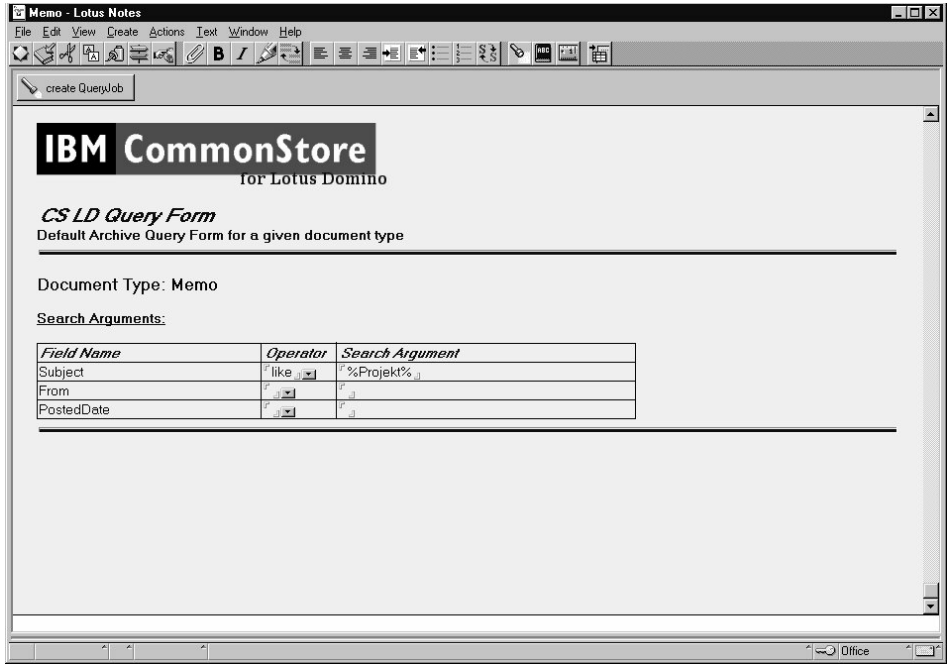

Figure 4. CSLD Query Form

Defining your own query form

The default query form provided by CSLD should only help to set up a database and form for archiving more quickly. The script code can be seen as sample code on how to create a query job. Since in a customer environment, archive queries might need to be set up in completely different fashions, it is of course possible to define your own query forms or to adapt the default form for your own purposes. The following list presents the items contained in the default form. For each of the mapped attributes, the form contains:

field_n
: Computed "text" field containing the name of the document field

op_n
: Keyword list containing the relational operators with which the search argument will be compared to the field value

searchArg_n
Field that has the same value type as the document field in the original form. This is necessary and important since the field type decides how the search argument will be syntactically built.

The script that builds a query job from the field entries will loop over all **field_n** and build a search predicate with the corresponding **op_n** and **searchArg_n**. It will combine all of theses search predicates logically together with **AND**, and store a query job with the corresponding query string set.

When defining your own query forms, it is very important to follow the syntactical rules given above (see "Query jobs" on page 162). Aside from that, customers have extensive freedom in specifying archive query forms.

Result documents

Notes documents created as the result of a retrieval request contain those fields which have been mapped to archive attributes (see "Document mappings" on page 19), the archived document itself appended as a file attachment to an RTF field, and the archive document ID.

The field names of the result document are the same as the field names defined in the mapping. The RTF field to which the document content will be attached is named **bodyDocContent**, and the archive document ID will be provided in a field named **CSNDArchiveID**.

Since such a result document is a good candidate for an update document in update requests, the field types are the same as the field types of the Notes document from which the archive document was created.

Result documents contain two additional items: **requester** and **reqTS**:

requester
The "text" field containing the name of the user who issued the retrieval request or query resulting in this document.

reqTS The time at which this request was made.

These two items are included in the result document primarily so that retrieval results can be more easily associated to a particular request.

Example:
The target database could define a view that presents archive retrieve results categorized by user name (i.e. the requester) and ordered by the request time. In this fashion, a user can locate all of his/her results at once.

Displaying query results

If an archive query returns more documents than specified with the parameter *maxNumOfDirectHits* (see "Query jobs" on page 162), those documents will be returned without their content. The administrator setting up a CSLD system has two choices as to how these results will be displayed. From the profile (see "Chapter 7.2 Configuration Database — Customizing" on page 125 for details), the administrator can select either a single hitlist document (which will contain references to all hits returned by the query) or multiple result documents (each of which will represent a single hit). Both of these choices, together with their advantages and disadvantages, are described in the following sections.

Displaying a query result by means of a single hitlist document

For each returned document, a hitlist document will contain those fields that best describe the archive document as well as a button to automatically generate a retrieval request for it.

Hitlist documents contain

- the requester and request timestamp of the query request that resulted in this hitlist,
- the document type (i.e. the form name) of all the hits contained in the document,
- the hits themselves contained in an RTF item, and also
- a list of all archive document IDs.

CSNHitlistForm: CSLD provides a default form that can be used to browse archive query results. This default form can be created in every Notes database used as target database for retrieval requests using the CSLD setupDB tool. In contrast to the result form, where one result form has to be created for each mapped document type, one instance of this form will suffice per target database, since it can be used to display hitlists for all document types.

This form contains requester, timestamp, and the table of hits as visible fields and the list of archive IDs in a hidden field.

Defining your own hitlist form: The form provided by CSLD should only help to set up database for archiving more quickly. It is, however, possible to define your own forms or to adapt the default result form for your own purposes. Generally, defining your own hitlist form is a little more restricted than creating a result form, since the main information is contained in an RTF field. Thus, the base layout of a hitlist document is not changeable.

The following list presents the items contained in a result document returned by CSLD:

requester
The ″text″ field containing the name of the user who issued the retrieval request or query resulting in this document.

reqTS The ″time″ field containing the date and time at which the request was issued

numHits
The ″number″ field containing the number of hits, i.e. the number of archive document IDs contained in this hitlist

docType
The ″text″ field containing the form name of the original form. This is the document type given in the mapping. The mapped fields are determined on the basis of this form.

bodyHitlist
The ″RTF″ field containing the actual hitlist. This hitlist is set up as a table, with a row for each hit in the document and a column for each representative attribute defined in the document mapping for this form, plus an additional column for the **Fetch** button that can be used to generate a retrieval request for the document described in this row.

CSNDArchiveID
The ″text″ field containing a list of all archive document IDs. This list can be used if some external logic is used to process documents returned by a query request. For example: The 'Fetch All' action defined in the hitlist form uses the content of this field to retrieve all documents in the hitlist.

Displaying a query result by means of multiple result documents

If **Multiple Result Documents** was selected when setting up CSLD, search hits will always be represented by result documents. Whether or not such a result document contains a document content solely depends on the parameter *maxNumOfDirectHits*. If the threshold defined by this parameter is exceeded, the result document consists only of indexing information. With a number of hits lower than or equal to *maxNumOfDirectHits*, the result document will be built with document content appended as an attachment

The advantage of using multiple result documents instead of a single hitlist document is that the hits can be viewed in a database view. This simplifies the selection of hits for which content should be retrieved. It also allows for ordering the result documents based on one or even multiple column values. On the other hand, a possible drawback might be that an archive query might return a possibly large number of result documents. Those documents must all be deleted by hand in order to keep the result database laid out clearly.

CSNResultForm

CSLD provides a default form named **CSNResultForm** that can be used to browse retrieved archive documents. This default form can be created for every Notes document type mapped to the archive in every Notes database used as target database for retrieval requests using the CSLD setupDB tool.

This form contains requester, timestamp, all index information and the document content body field as visible fields and the archive ID as a hidden field.

Defining your own result form

The form provided by CSLD is intended only to provide help in setting up the database for archiving more quickly. It is, however, possible to define your own forms or to adapt the default result form for your own purposes.

Generally, the default form can be changed in any desired fashion as long as the fields names remain the same. But customers may also want to set up completely different forms with a thoroughly different layout or containing additional fields.

The following list presents the items contained in a result document returned by CSLD:

<fields_1> .. <field_n>:
: A result document will contain all mapped fields for the given document type. These fields have the same names and the same types as the fields in the original form (i.e. the form from which they were created).

requester
: The ″text″ field containing the name of the user who issued the retrieval request or query resulting in this document

reqTS The ″time″ field containing the date and time at which the request was issued.

docType
: The ″text″ field containing the form name of the original form. This is the document type given in the mapping. The mapped fields are determined on the basis of this form.

bodyDocContent
: The ″RTF″ field containing the document content appended as an attachment.

CSNDArchiveID
: The ″text″ field containing the archive document ID of this document.

CSNDRequestType
The ″number″ field containing the request type code (see “General job parameters” on page 153) with which this document was originally archived.

Notes folder restore

When an entire Notes folder structure has been archived, that structure can be restored to Notes in one step by specifying a special request type. Notes folders can either be restored by their name or by the archive document ID it was given during archiving. As with single document retrieval, restoring a Notes folder by its archive document ID is faster than restoring it by name.

When restoring subfolders within a hierarchy that has been archived, you must follow the Notes naming conventions for folders. In Notes a subfolder is identified through prepending its root folder's name, e.g. ″RootFolder\SubFolder″.

Entire Notes folder structures can only be restored back to the database they originally came from. This request will recreate the same folder structure that existed before it was archived.

> **Note**
> In order to see the result of a folder restore, the Notes database must first be closed and then reopened.

Notes folder restore job fields

If the **requestType** field is set to CSN_RESTORE_FOLDER, in addition to the general fields, the following fields must also be defined:

Field name	Data type	Usage
folderArchID	Text	Document archive ID that was given to the archived folder when it was archived by CSLD. Either this field or **folderName** must be given
folderName	Text	Name or Alias of the archived folder. When specifying subfolders a naming like ″Folder\Subfolder″ must be used. Either this field or **folderArchID** must be given.
targetDB	Text	Specifies the path to a working database.

targetSrv	Text	The name of the server on which this database resides. **targetSrv** together with **targetDB** are used by the CSLD processes to locate the database in which documents returned by retrieval requests will be stored. For folder restore this must be the database the folder originally was archived from.

Listing documents in a workbasket

Listing the documents in a workbasket is performed via a retrieve job with request type CSN_LIST_WORKBASKET. In addition to the general fields, the following fields must be set:

Field name	Data type	Usage
targetDB	Text	Name of the database to write the hitlist (or multiple result documents) to.
targetSrv	Text	Name of the server hosting the database given by the targetDB field. **Important:** The server name must be given in *abbreviated* format, not in canonical format.
targetFolder	Text	Optional. If this field exists, the hitlist (or multiple result documents) will be added to the folder with the given name. Otherwise the histlist document will simply be written to the database, and the application must provide a custom view to see it.
workbasketName	Text	Name of the workbasket to list.
WBArchiveID	Text	Since CSLD can archive to more than one archive, the archive server containing the target workbasket must be specified by this field. Set this field to the logical archive ID (must be defined in archpro.ini) of the target archive server. The archive ID includes the server. Do not specify the name of an archive server.
parentDocUNID	Text	Optional. The hitlist document (or multiple result documents) with the documents in the workbasket become responses to the document with the universal Notes ID (UNID) specified in this field. This allows creating sophisticated categorized views with hierarchies, as in a discussion database.

Chapter 8.2 Enabling user databases for CSLD

A user database is CSLD-enabled by performing several steps. Follow these steps closely to ensure that your Notes applications are set up correctly

Access rights

All documents are archived or retrieved via CSLD tasks, each running under a particular CSLD user ID. To enable a database for CSLD usage, add the CSLD user to that database's ACL. The access level must be 'Editor'. If you use the CSLD setupDB tool to create default forms, the CSLD user must have 'Manager' rights on the database setupDB is applied to. You can also modify the database template ACL so that the CSLD user is already part of the ACL when a new database is created from that template. This is especially helpful in large companies where mail databases are created frequently. To add the CSLD user to a major number of databases, you can write a Lotus Script, C, C++ or Java Application.

An alternative is to add the CSLD user to a group that is already a member of the database ACL. This way, no changes must be made to the ACL. If you modify the database template ACL, you can update numerous databases at once by running the ″load design″ server add-in task.

When implementing CSLD functionality a set of forms must be defined. The setupDB tool provides the functionality to create defaults for some of these forms. Other forms as well as other design elements are provided as defaults in the CSLD configuration database and can simply be copied. However, all design elements provided by CSLD can be customized as desired. See "Chapter 8.1 Creating job documents" on page 153 for details.

SetupDB

CSLD is shipped with a tool that helps to get a quick start when enabling Notes databases for archiving functionality. This tool is called setupDB and can be found in the tools subdirectory of the installation directory of CSLD.

Generally CSLD works based on Notes forms or document types. When preparing a given Notes form for retrieval with CSLD, basically two kinds of additional documents are needed in the Notes database from which archive queries are issued and to which search results are stored:

1. a query form to enter the parametric search that will be passed to the underlying archive and

2. a form to display archived content with.

A third form, the so-called hitlist document, might also be needed when CSLD is configured to use single hitlists instead of multiple result documents

setupDB is used to automatically create defaults for query and result forms for a given Notes form. To be able to do so, setupDB has to be provided an example document created using the form that is to be set up. This document has to be placed in the configuration database prior to starting setupDB. According to the document mapping for this form setupDB will check the example document's item types and create respective fields in the default forms it creates.

> **Note**
> Since hitlist forms are independent of a certain Notes form, setupDB does not create a default for a hitlist. However, CSLD also provides a default hitlist form. This form, called CSNHitlistDoc can simply be copied from the template of the CSLD configuration database to any Notes database using CSLD.

How to set up a Notes form using the setupDB tool

A setupDB is an independent tool that can be run from the command line. As a prerequisite, a Notes client must be installed on the machine running this tool. SetupDB can be started as follows:

```
setupDB    -cfgdb <cfgDBName>
           [-cfgsrv <cfgSrvName>]
           -db <dbName>
           [-srv <srvName>]
           -form <formName>
```

cfgSrvName, cfgDBName
: Server and database name of CSLD configuration database. This database must contain the field to attribute mappings for the given form. It must also contain an example document of that type, so setupDB can match the datatypes of all mapped fields.

srvName, dbName
: Server and database name of Notes database to which setupDB will store the default forms it created. Make sure that this database contains the CreateCSNJobs script library, since the default query form makes use of methods defined therein.

formName

Notes form name for which to create default forms. A document that uses this form is expected to have been copied to the configuration database (example document).

Initial setup of a Notes database

When setting up a Notes database for archiving with CSLD the following steps have to be performed:

1. Open the template for the CSLD configuration file (CSLDConfig.ntf) in the Domino Designer (for Notes R4 open the design tab).
2. You will find three Lotus script libraries:
   a. CreateCSNJobs - has to be present and correctly configured in every database using CSLD features (at least when using the CSLD default forms).
   b. CSNJobSamples - can be used to adopt the database quicker for archiving, since it defines default actions for the most common tasks.
   c. CSNWebFunctions - must be copied to the Notes database when it should be accessed from a Domino Web client.

   In addition to these script libraries CSLD also provides three script agents, (WebCreateQuery), (WebRetrieveAllInHitlist) and (WebRetrieveSingleDoc). These agents also must be copied to the Notes database to be set up if it will be accessed by Web clients.
3. Open the template for the Notes database to be set up in the Domino Designer and paste at least script library CreateCSNJobs to it.
4. Open script library CreateCSNJobs in the Designer. CreateCSNJobs defines two functions, JobDatabaseName and JobDatabaseServer. Adapt those functions to return the database name or server respectively, where your job database resides.
5. From the template of the CSLD configuration database copy form CSNHitlistDoc and paste it to the template of the Notes database to be set up.

While the above steps must be performed only once for every database to be enabled, the following steps are necessary for every form that should be enabled for archiving.

1. Copy a document created with the form you want to enable for archiving and paste it to the CSLD configuration database.
2. Ensure that each of the fields defined in the corresponding document mapping is available and set in this document.
3. Call setupDB with the appropriate parameters for this form. SetupDB will now create the default forms for archive searches and displaying search results.

This form is now set up for archiving functionality. If the Notes database to be set up includes other forms that should also be mapped, repeat the above steps to set them up accordingly.

Once finished setting up the database and creating the default forms, the Notes application can be customized by adding buttons or agents to invoke CSLD functions, creating special views e.g. to display search results in etc.

Chapter 8.3 Script classes

Lotus Script helper classes

CSLD jobs define several document fields, but none of the different job makes use of all of these fields. For this reason, CSLD provides a Lotus Script Library containing a set of script classes that can be used to simplify job document generation. Basically, for each type of job request, there is a class which encapsulates that job request type and which defines properties to set and get the necessary parameters while setting those parameters that can be gotten from the working environment automatically and therefore ensuring that the job documents generated are consistent.

These helper classes are defined in a Lotus Script library named **CreateCSNJobs** which can be found in the CSLD configuration database. This script library should be imported by any Notes database making use of CSLD archiving capabilities.

In addition to the **CreateCSNJobs** script library, the CSLD configuration database contains another script library named **CSNJobSamples** which defines methods (subs) that implement the **CSNJob** script classes. These methods can be used as sample code to get a quick start when implementing CSLD archiving functionality.

The following sections describe these classes as well as the defined constants and provide examples on how to use them.

Class hierarchy

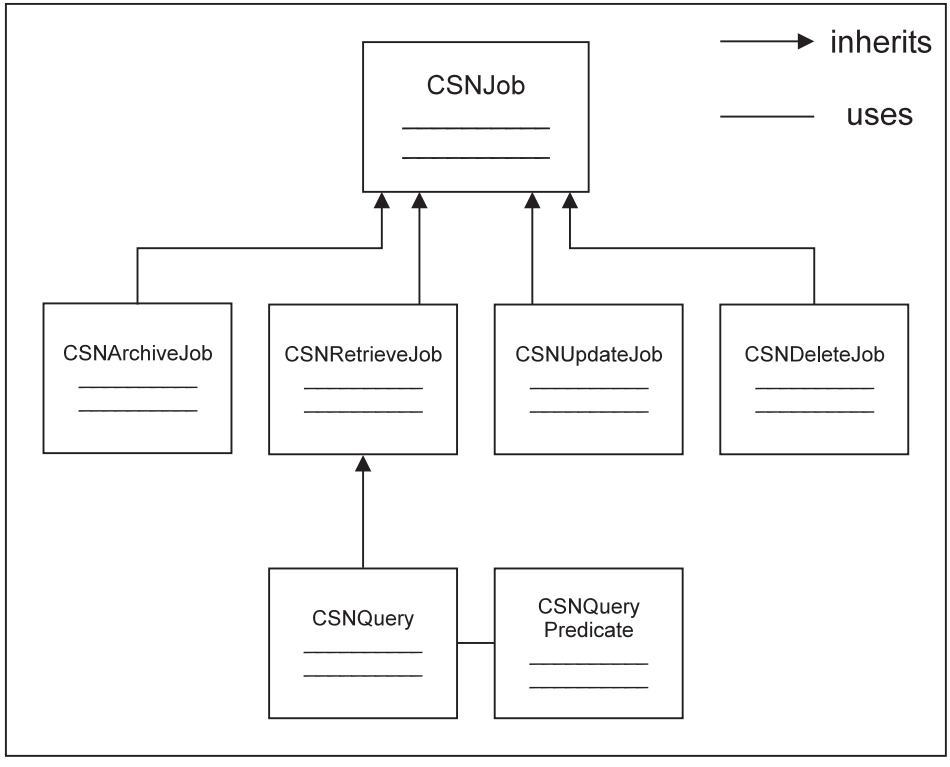

Figure 5. Class hierarchy of the CSLD script class

The base class **CSNJob** contains 'get' and 'set' methods for the general parameters mentioned above. The derived classes **CSNArchiveJob**, **CSNRetrieveJob**, **CSNUpdateJob**, **CSNDeleteJob**, and **CSNQuery** each contain the 'get' and 'set' methods needed for a job document of their respective type. **CSNQuery** also defines methods enabling the user to build up an archive query, i.e. to set the value of the **CSNQryString** within a job document, in a convenient way.

In addition to the 'get' and 'set' methods, each derived class defines a method called 'storeJobDocument', which will create a job document from the job in a given job database.

Thus, users should proceed as follows when creating CSLD jobs: The application making use of CSLD functionality creates a job document in some job database using the script classes. These jobs can then be viewed and their

process be tracked with the forms defined in the job database. Creating jobs manually by filling in the corresponding forms is cumbersome and should probably be avoided.

Constants

Request type

There are symbolic values for the available defined request types that can be used instead of the numerical values themselves. These values are:

- Request Types an Archiving Task will process:

```
Const CSN_ARCHIVE_ATTACHMENTS% = 1
Const CSN_ARCHIVE_NATIVE% = 2
Const CSN_ARCHIVE_TIFF_FORMAT% = 3
Const CSN_ARCHIVE_ASCII_FORMAT% = 4
Const CSN_ARCHIVE_RTF_FORMAT% = 5
Const CSN_DELETE_BY_ID%=6
Const CSN_UPDATE_INDEX%=7
Const CSN_MOVE_TO_WORKBASKET%=8
Const CSN_REMOVE_FROM_WORKBASKET%=9
```

- Requests a Retrieval Task will process:

```
Const CSN_REQUEST_BY_ID% = 50
Const CSN_QUERY% = 51
Const CSN_LIST_WORKBASKET% = 52
Const CSN_RESTORE_FOLDER%=53
```

It is highly recommended to always use the symbolic values instead of the integer values they stand for.

Raster Archiving Options

There are symbolic values defined for the available raster options. Either of these can be specified when the request type is set to CSN_ARCHIVE_TIFF_FORMAT. Again, it is highly recommended to use the symbolic values instead of the actual integers.

```
Const CSN_RASTER_NOTE_APPEND_ATTACHMENT% = 100
Const CSN_RASTER_NOTE_EMBED_ATTACHMENT% = 101
Const CSN_RASTER_NOTE_ATTACHMENTS_ONLY% = 102
```

CSN_RASTER_NOTE_EMBED_ATTACHMENT will be the default option if not otherwise set.

Job state

There are symbolic values defined for the available job states that can be used instead of the numerical values. These values are:

```
Const CSN_JOB_TOBEPROCESSED = 1
Const CSN_JOB_INPROCESS = 2
Const CSN_JOB_SUCESS = 3
Const CSN_JOB_ERROR = 4
```

Job documents that are stored to the database are expected to have their job state set to CSN_JOB_TOBEPROCESSED.

Error codes

There are symbolic error codes defined for the errors that are thrown by the **CSNJob** classes. Possible error codes are:

```
Const CSNERR_VARNOTBOOL%                = CSNERR_BASE + 1
Const CSNERR_INVALIDREQTYPE%            = CSNERR_BASE + 2
Const CSNERR_UNIDNOARRAY%               = CSNERR_BASE + 3
Const CSNERR_NOARCHDOC%                 = CSNERR_BASE + 4
Const CSNERR_NOSOURCEDB%                = CSNERR_BASE + 5
Const CSNERR_DBOPEN%                    = CSNERR_BASE + 6
Const CSNERR_STOREJOBDOC%               = CSNERR_BASE + 7
Const CSNERR_NOTARGETDB%                = CSNERR_BASE + 8
Const CSNERR_NOTARGETFIELD%             = CSNERR_BASE + 9
Const CSNERR_NOARCHID%                  = CSNERR_BASE + 10
Const CSNERR_UNEXPECTED%               = CSNERR_BASE + 11
Const CSNERR_WRONG_ITEMTYPE%            = CSNERR_BASE + 12
Const CSNERR_NOWORKBASKET%              = CSNERR_BASE + 13
Const CSNERR_NOWBARCHIVEID%             = CSNERR_BASE + 14
Const CSNERR_NOFOLDER%                  = CSNERR_BASE + 15
```

CSNJob

CSNJob is the base class for all job classes provided by CSLD. It defines all the general parameters and interfaces for generating job documents.

Methods and properties for **CSNJob** are described in the following sections.

Public properties for CSNJob

DeleteJobAfterSuccess As Variant

IsJobDeletedAfterSuccess As Variant

This is a Boolean flag stating whether or not the job document should be automatically deleted from the job database once the job has successfully completed processing all of the documents in the job. The property is expected to be Boolean. For any other type, error CSNERR_VARNOTBOOL is thrown.

Public subs for CSNJob

New(reqType As Integer, dbName As String, srvName As String)

This is a constructor for job documents. Input parameters:

reqType

This parameter specifies the request type code determining the kind of job encapsulated by this object.

dbName

This parameter specifies the path to the job database in which this job will be stored.

srvName

This parameter specifies the name of the server on which *dbName* resides.

Public functions for CSNJob

StoreJobDocument As String

An abstract method that will be implemented in each of the derived classes. This function should be the last call to one of the **CSNJob** documents. It builds a Notes document based on the properties that have been set and stores the newly-built document to the job database specified in the constructor

CSNArchiveJob

CSNArchiveJob encapsulates a job document describing an archiving job. It includes all necessary 'set' and 'get' methods for the parameters needed to create a valid archiving request to CSLD, while it sets those parameters that can be gotten from the environment automatically. When using the **CSNArchiveJob** class to build up archiving requests, the user can be sure to get valid and consistent archiving job documents.

Public properties for CSNArchiveJob

Set ArchivalDocs As Variant

Get ArchivalDocs As Variant

Used to set/get the array of document UNIDs representing the documents to be archived in this job. Alternatively, a single UNID representing a view or folder may be passed on call. The variant is expected to be an array of strings. For other types, error CSNERR_UNIDNOARRAY is thrown.

Set SourceDBName As String

Get SourceDBName As String

Used to set/get the path name to the database in which the documents to be archived are located. The format in which the path name is expected is analogous to NoteSession's **GetDatabase** sub.

Set WorkbasketName As String

Get WorkbasketName As String

Used to get/set the name of a workbasket within the archive to which the document will be stored.

Set DeleteOriginal As Variant

Get IsOriginalDeleted As Variant

Used to get/set the flag telling the system whether or not the original document is to be deleted from its originating database after it has

been successfully archived. The variant is expected to be Boolean. For other types, error CSNERR_VARNOTBOOL is thrown.

Set DeleteAttachments As Variant

Get AreAttachmentsDeleted As Variant
Used to get/set the flag telling the system whether or not, in the case of attachment archiving, the file attachments are to be deleted from their originating documents after they have been archived. The variant is expected to be Boolean. For all other types, error CSNERR_VARNOTBOOL is thrown.

Set KeepFolderStructure As Variant

Get KeepFolderStructure As Variant
Used to determine whether to archive an entire folder structure or all documents within a folder separately. This flag will only be evaluated if the UNID specified as **archDocUNID** refers to a Notes folder. Variant is expected to be Boolean. For all other types, error CSNERR_VARNOTBOOL is thrown.

Set RasterOption as Integer

Get RasterOption as Integer
Used to set an additional option as to what parts of a note should be rastered. This parameter will only be considered when the **requestType** was set to CSN_ARCHIVE_TIFF_FORMAT. Expected values are

1. CSN_RASTER_NOTE_APPEND_ATTACHMENTS,
2. CSN_RASTER_NOTE_EMBED_ATTACHMENTS or
3. CSN_RASTER_NOTE_ATTACHMENTS_ONLY.

Public subs for CSNArchiveJob

New(reqType As Integer, dbName As String, dbSrv As String)
See description of **CSNJob** constructor. Expected request types are:

- CSN_ARCHIVE_ATTACHMENTS,
- CSN_ARCHIVE_NATIVE,
- CSN_ARCHIVE_TIFF_FORMAT,
- CSN_ARCHIVE_RTF_FORMAT, and
- CSN_ARCHIVE_ASCII_FORMAT.

For all other request types, the constructor throws error CSNERR_INVALIDREQTYPE.

Public functions for CSNArchiveJob

StoreJobDocument As String
Concrete implementation of the sub defined in the base class **CSNJob**.

Example

A typical scenario for making use of the **CSNArchiveJob** functionality is when implementing an agent or action that will act on the selected documents. In the following example, the agent will create one archiving job for all selected documents. The request is to archive the document attachments, delete them from their originating documents upon success, and to delete the job document, too:

```
Dim session As New NotesSession
Dim db As NotesDatabase
Dim docList As NotesDocumentCollection
Dim doc As NotesDocument

Dim job As CSNArchiveJob

Set db = session.currentDatabase
Set docList = db.UnprocessedDocuments

Set job = New CSNArchiveJob( CSN_ARCHIVE_ATTACHMENTS,
                             JobDatabaseName,
                             JobDatabaseServer)

' Create a string array for the UNIDs
' set its upper bound to as many entries as docList.
' Array is zero-indexed, therefore count-1
Dim docIdx As Integer
docIdx = docList.Count-1
Dim unidList(docIdx) As String

Set doc = docList.GetFirstDocument
For i=0 To docIdx
     unidList(i) = doc.UniversalID
     Set doc = docList.GetNextDocument(doc)
Next

' now set all general parameters needed in an
' archiving job
job.ArchivalDocs          = unidList
job.SourceDBName          = ArchiveDatabaseName
job.SourceSrvName         = ArchiveDatabaseServer
job.DeleteOriginal        = false
job.DeleteJobAfterSuccess = true
job.DeleteAttachments     = true
' finally store the job document to the job database
Call job.storeJobDocument()
```

CSNRetrieveJob

CSNRetrieveJob encapsulates a job document describing a request to retrieve single archive documents on the basis of their archive document IDs. It includes all methods to set the required parameters while setting those parameters that can be gotten from the environment automatically. By using **CSNRetrieveJob**, the user can in a simple way create consistent and valid retrieval requests.

Public properties for CSNRetrieveJob

Set DocumentType As String

Get DocumentType As String
Used to get/set the document type, i.e. the form name of the document(s) to be retrieved. It is not necessary to set this property, but for the purpose of clarity, when browsing job documents, you might nevertheless want to set it.

Set TargetDBName As String

Get TargetDBName As String
Used to get/set the database path of the Notes database to which to restore the retrieved document(s). The syntax of database path is the same as for NotesSession's **GetDatabase** sub.

Set TargetSrvName As String

Get TargetSrvName As String
Used to get/set the name of the server on which **targetDB** resides.

Set TargetFolderName As String

Get TargetFolderName As String
Used to get/set the optional folder to which to restore the retrieved document(s). This parameter does not need to be set. If not set, all documents will be restored directly into the given target database. If set, they will be moved to the given folder.

Get TargetDocUNID As String
Used to get the optionally-set UNID of a document to which the retrieved content will be appended as an attachment. The target document can be set using the **setTargetDoc** sub described in "Public subs for CSNRetrieveJob" on page 185.

Get TargetFieldName As String
Used to get the name of the RTF field to which the document content will be appended. This parameter will only be set when a target document was specified. The target field can be set together with the target document using the **setTargetDoc** sub described in "Public subs for CSNRetrieveJob" on page 185.

Set ArchIDs As Variant

Get ArchIDs As Variant
Used to get/set the document archive IDs of the documents that should be retrieved within this job. These archive IDs will be returned by the archiving request once a document has successfully been stored to the archive.

Set ParentDocUNID as String

Get ParentDocUNID
Used to get/set the UNID of a document to which the retrieved document(s) will be made responses. Use this feature to build up categorized views, in which e.g. a folder is parent to all documents residing in it.

Set WorkbasketName As String

Get WorkbasketName As String
Used to set the name of the archive workbasket for which to retrieve its content. When specifying this property, property **WorkbasketArchiveID** must also be specified. This parameter will only be considered for **requestType** CSN_LIST_WORKBASKET.

Set WorkbasketArchiveID As String

Get WorkbasketArchiveID As String
Used to set the CommonStore logical archive ID for the server on which the requested workbasket resides. This property must be set in conjunction with **WorkbasketName**. This parameter will only be considered for **requestType** CSN_LIST_WORKBASKET.

Set NotesFolderName As String

Get NotesFolderName As String
Used to set the name or alias of the Notes folder to be restored. If restoring a subfolder the full hierarchical name (e.g. ″Folder\Subfolder″) must be specified. This parameter will only be considered for **requestType** CSN_RESTORE_FOLDER.

Set FolderArchiveID As String

Get FolderArchiveID As String
Used to set the archive document ID of the Notes folder to be restored. This parameter will only be considered for **requestType** CSN_RESTORE_FOLDER.

Set TreatNativeAsNew As Variant

Get IsTreatNativeAsNew
Used to specify how to treat natively archived documents when they are retrieved. Variant is expected to be Boolean. For all other types error CSNERR_VARNOTBOOL will be thrown. Setting this parameter to TRUE will restore natively archived documents as new documents (i.e. without their UNID), while setting it to FALSE will restore them including their UNID.

Public subs for CSNRetrieveJob

new(requestType As Integer, jobDB As String, jobSrv As String)
See description of base class constructor. The expected request type is:

CSN_REQUEST_BY_ID. For all other request types, the constructor throws error CSNERR_INVALIDREQTYPE.

SetTargetDoc(docUNID As String, bodyField As String)

In the case of attachment archiving, you might want to restore archived content to the original document. In this case, a document can be specified by its UNID and the name of an RTF field to which to append the document content.

Public functions for CSNRetrieveJob

StoreJobDocument As String

Concrete implementation of the method (defined in the base class **CSNJob**) to store the job document (described within the object) to the given job database.

Example

In the following example, the currently selected document on the workspace contains a field named **CSNDArchiveID** whose value is the archive ID of an archived document. The script code will create a **CSNRetrieveJob** from that ID and attach the document content returned back to the original document (i.e. the current one) to an RTF field named **BodyInfo**.

```
Dim ws As New NotesUIWorkspace
Dim wsDoc As NotesUIDocument
Dim doc As NotesDocument
Dim item As NotesItem

Dim job As CSNRetrieveJob

Set job = New CSNRetrieveJob( CSN_REQUEST_BY_ID,
                              JobDatabaseName,
                              JobDatabaseServer)

Set wsDoc = ws.currentDocument
Set doc = wsDoc.document

If( doc.CSNDArchiveID(0) <> "Error" ) Then
     ' set the return document to doc itself into item "BodyInfo"
     Call job.setTargetDoc(doc.UniversalID, "BodyInfo")

     ' get the archive ID(s) stored to the CSNDArchiveID field
     ' and make them the archive IDs for this job
     job.ArchIDs = doc.GetItemValue("CSNDArchiveID")

     ' now set the general parameters for this job
     job.TargetDBName = ArchiveDatabaseName
     job.TargetSrvName = ArchiveDatabaseServer

     ' no necessary parameter, just for clarity...
     job.DocumentType = doc.Form(0)
```

```
        ' finally store the job document to the database
        Call job.storeJobDocument()

    End If
```

CSNDeleteJob

Public properties for CSNDeleteJob

Set DeletionIDs As Variant

Get DeletionIDs As Variant
Used to get/set the archive document IDs of the documents that are to be deleted in this job.

Set SourceDBName As String

Get SourceDBName As String
The database path name of the source database. This parameter needs to be set, since CSLD processes are configured to process requests from one or more Notes production databases. While specifying a source or target database for a deletion request is not necessary, some database name must be set, because otherwise no CSLD process will work on the job.

Set SourceSrvName As String

Get SourceSrvName As String
The name of the server on which **sourceDB** resides.

Public subs for CSNDeleteJob

new(requestType As Integer, jobDB As String, jobSrv As String)
See description of the base class constructor. The expected request type is: CSN_DELETE_BY_ID. For all other request types, the constructor throws error CSNERR_INVALIDREQTYPE.

Public functions for CSNDeleteJob

StoreJobDocument As String
Concrete implementation of the method (defined in the base class**CSNJob**) to store the job document described by this object to the given job database.

Example

In the following example, a given list document contains an item called **ArchiveIDs**. This field contains several archive document IDs. These will be written to a delete job.

```
Dim ws As New NotesUIWorkspace
Dim uidoc As NotesUIDocument
Dim doc As NotesDocument
Dim doc As NotesDocument
```

```
Dim deletionIDsItem As NotesItem

Dim deleteJob As New CSNDeleteJob( CSN_DELETE_BY_ID,
                                   JobDatabaseName,
                                   JobDatabaseServer )

Set uidoc = ws.currentDocument
Set doc = uidoc.Document

Set deletionIDsItem = doc.GetItem("deleteIDs")

' set the job parameters
deleteJob.SourceDBName = ArchiveDatabaseName
deleteJob.SourceSrvName = ArchiveDatabaseServer
deleteJob.DeleteJobAfterSuccess = True

deleteJob.DeletionIDs = deletionIDsItem.Values

' finally store the job document to the
' job given database
Call deleteJob.storeJobDocument()
```

CSNUpdateJob

Public properties for CSNUpdateJob

Set WorkbasketName As String

Get WorkbasketName As String

Sets the name of the workbasket the documents are moved to. If a workbasket is set, no index will be updated. The move to the workbasket has higher priority.

Set UpdateDocs As Variant

Get UpdateDocs As Variant

Used to get/set the document UNIDs of the Notes documents that contain the updated index information for archive documents. A Notes document becomes an update document if it contains an item named **CSNArchiveID** whose value is set to an archive document ID and either has the same form as the archived document or is a result document containing the name of the document type to which it belongs in an item. The variant is expected to be a string array. For all other value types, error CSNERR_UNIDNOARRAY is thrown.

Set SourceDBName As String

Get SourceDBName As String

Used to get/set the database path of the Notes database in which the given document(s) is (are) found. The syntax of database path is the same as for NotesSession's **GetDatabase** sub.

Set SourceSrvName As String

Get SourceSrvName As String
Server name on which **SourceDBName** resides.

Public subs for CSNUpdateJob

new(reqType As Integer, dbName As String, dbSrv As String)
See description of the base class constructor. The expected request type is: CSN_UPDATE_INDEX. For all other request types, the constructor throws error CSNERR_INVALIDREQTYPE.

Public functions for CSNUpdateJob

StoreJobDocument As String
Concrete implementation of the method (defined in the base class **CSNJob**) to store the job document described by this object to the given job database.

Example

The following example demonstrates a view action that acts on all documents currently selected in the view. The action loops over the document collection, filters all those documents containing a **CSNDArchiveID** item, and creates an update job from them

```
Dim db As NotesDatabase
Dim docList As NotesDocumentCollection
Dim doc As NotesDocument

Dim job As CSNUpdateJob

Set db = session.currentDatabase
Set docList = db.UnprocessedDocuments

Set job = New CSNUpdateJob( CSN_UPDATE_INDEX,
                            JobDatabaseName,
                            JobDatabaseServer )

' Create an undimensioned array for the UNIDs
' then redim it to contain as many entries as docList.
' Array is zero-indexed, therefore count-1
Redim dummy(0) As String
Dim docIdx As Integer
docIdx = docList.Count-1

Set doc = docList.GetFirstDocument
For i=0 To docIdx
     ' only take those docIDs for updating that have
     ' a field "CSNDArchiveID" whose value is NOT "Error"
     If( Not doc.GetFirstItem("CSNDArchiveID") Is Nothing _
     And doc.CSNDArchiveID(0) <> "Error" ) Then
          Redim Preserve dummy(i)
          dummy(i) = doc.UniversalID
     End If
```

```
     Set doc = docList.GetNextDocument(doc)
Next

' before setting all job parameters check that at least ONE
' update document was found
If( dummy(0) <> "" ) Then
     job.UpdateDocs            = dummy
     job.SourceDBName          = ArchiveDatabaseName
     job.SourceSrvName         = ArchiveDatabaseServer
     job.DeleteJobAfterSuccess = deleteJob

     ' finally store the job document just built
     Call job.storeJobDocument()
End If
```

CSNQueryPredicate

Public properties for CSNQueryPredicate

Set SearchField As String

Get SearchField As String
Used to get/set the name of the field used as a query predicate.

Set SearchOperator As String

Get SearchOperator As String
Used to get/set the relational operator used. Allowed values are:

- like
- <
- <=
- =
- >
- >=

A validity check of the set operator is not done in the script classes.

Set SearchArgument As Variant

Get SearchArgument As Variant
Used to get/set the value used as search argument for the predicate.

Public functions for CSNQueryPredicate

searchArgAsString() As String
This function is used to convert the value given to its valid string representation. This string representation will then be used to build up the query string.

CSNQuery

Public properties for CSNQuery

Set MaxDirectHits As Integer

Get MaxDirectHits As Integer

Used to get/set the number of hits that should be returned directly by that query. This number defines up to how many hits will be returned as a complete document with document content. In the case of native archiving, this will be the Notes document itself; in all other cases, this will be a result document containing the fields mapped for its document type and the document content appended as a file attachment.

Set MaxTotalHits As Integer

Get MaxTotalHits As Integer

Used to get/set the number of hits this query will return at most. This number defines the absolute maximum number of hits to be returned. It is supposed to be higher than or equal to the number of direct hits. If the number of hits the given query produces lies between *maxDirectHits* and *maxTotalHits*, a single hitlist document will be created containing a table with representative fields for the given document type and one row for each hit returned.

Set DocumentType As String

Get DocumentType As String

Used to get/set the only document type this query yields to. The document type defines which archive container the query applies to. By using the set property, a single document type will be set. If the current query should yield to more than one archive container, use the **addTargetDocType** sub instead (for description see "Public subs for CSNQuery").

Public subs for CSNQuery

addPredicate(p As CSNQueryPredicate)

Adds a new predicate to the query (see also "CSNQueryPredicate" on page 190).

and()
or()

The **and()**, **or()**, **openParentheses()**, and **closeParentheses()** subs are used to combine the given predicates together. The order in which calls to **addPredicate()** and the combination operators are made defines the way in which predicates are logically combined.

openParentheses()
closeParentheses()

Just as with the **and()** and **or()** subs, the **openParentheses** and

closeParentheses subs, too, are used to logically build up the query. They allow the precedence in which the operators will be evaluated by the archive search engine to be changed.

addTargetDocType(formName As String)
In contrast to the **setTargetDocType** property, this sub is used to add additional document types the current query should apply to. The document types will determine the archive containers in which the archive search engine will perform the query. All given document types are expected to have the field(s) referenced by the query's predicate(s). If any of the given target document types does not include any of the given fields, the search request will return an error.

new(reqType As Integer, dbName As String, dbSrv As String)
See description of the base class constructor. The expected request type is: CSN_UPDATE_INDEX. For all other request types, the constructor throws error CSNERR_INVALIDREQTYPE. New will create an empty query. At least one call to **addPredicate** has to be made in order to be able to store a valid query job.

Public functions for CSNQuery

StoreJobDocument As String
Concrete implementation of the method (defined in the base class **CSNJob**) to store the job document described by this object to the given job database.

Example
Suppose that the Notes database is a catalog of music. The document type is called **Recording**. An archive query looking for all recordings after 1985 conducted by either Leonard Bernstein or Herbert Karajan would be constructed as follows:

```
Dim query as New CSNQuery( CSN_QUERY,
                           JobDatabaseName,
                           JobDatabaseServer )

Dim recDatePred as New CSNQueryPredicate
Dim conductorPred1 as New CSNQueryPredicate
Dim conductorPred2 as New CSNQueryPredicate

' and set the query's only target document type to "Recording"
query.TargetDocType = "Recording"
' start with building all the predicates

' initialize a date Variant to current date then set it
Dim recDate as Variant
recDate = Date
recDate.Year = 1985
recDate.Month = 1
recDate.Day = 1
```

```
recDatePred.SearchField    = "RecordingDate"
recDatePred.SearchOperator = ">"
recDatePred.SearchArgument = recDate

conductorPred1.SearchField    = "Conductor"
conductorPred1.SearchOperator = "like"
conductorPred1.SearchArgument = "%Bernstein"

conductorPred2.SearchField    = "Conductor"
conductorPred2.SearchOperator = "like"
conductorPred2.SearchArgument = "%Karajan"

' now combine all these predicates into one query
Call query.addPredicate(composerPred)
Call query.and()
Call query.addPredicate(recDatePred)
Call query.and()
Call query.openParentheses()
Call query.addPredicate(conductorPred1)
Call query.or()
Call query.addPrecicate(conductorPred2)
Call query.closeParentheses()

' set the other parameters needed for a query job
query.TargetDBName  = WorkingDatabaseName
query.TargetSrvName = WorkingDatabaseServer
' build a maximum of 10 documents complete with content
' and return up to 50 hits in total
query.MaxDirectHits = 10
query.MaxTotalHits  = 50

' finally store the job document just built
Call query.StoreJobDocument
```

Part 9. Appendixes

Appendix A. CMCS Server Profile Keywords

General remarks

- Please do not use the name of keywords as values. It is especially important that you not use `VI` as an archive ID (keyword *ARCHIVE*).
- The character # is the comment symbol. When a line in the CMCS server ini file is started with #, this line is skipped completely.

The keywords *ARCHAGENT, ARCHAGENTVI, ARCHAGENTOD*, *DISPATCHER*, *ARCHWIN*, and *ARCHREG* are obsolete and are therefore no longer documented here. Nevertheless, CMCS server still accepts these keywords when found in the ini file, though a warning will be displayed. These obsolete keywords should be replaced by *BINPATH* (see corresponding entry on page 199), which specifies the directory in which all CMCS server binaries reside.

ACCESS_CTRL *YES|NO*

Additional keyword for the **ARCHIVE** statement. Specifies that the `CSExit.DLL` is called in order to replace the provided Lotus Domino user by another VI user. The `CSExit.DLL` is to be provided by the companies themselves.

DEFAULT

NO

EXAMPLE

```
ACCESS_CTRL YES
```

.

ADSMAGENTS *number*

Using this TSM-specific keyword, you can specify the total number of parallel Tivoli Storage Manager Client sessions (name: `archagent`) the CMCS Server establishes. For a direct archive/retrieve operation on tape drives, keep the following in mind: The number of sessions must be equal to or lower than the number of tape drives available for archiving. For performance reasons, it is recommended that you use as many agents as there are tape drives available. The default is **0**.

EXAMPLE:

```
ADSMAGENTS   3
```

ADSMNODE *nodename*

Using this TSM-specific keyword, you can specify the Tivoli Storage Manager node name for the Tivoli Storage Manager log-in procedure.

If Tivoli Storage Manager's *Password Generate* option is used, the node name must not be specified in the CMCS server configuration profile.

> **Note:**
> This keyword refers to the corresponding **ARCHIVE** statement and is used only for Tivoli Storage Manager archives.

APPGROUP *group*

Using this CMOD-specific keyword, you can specify the name of the CMOD application group. All application group names containing spaces must be enclosed in single quotation marks.

EXAMPLE:

```
APPGROUP   'CSLD Mail Demo'
```

> **Note:**
> This keyword refers to the corresponding **ARCHIVE** statements and is used only for CMOD archives.

APPLICATION *app*

Using this CMOD-specific keyword, you can specify the name of the CMOD application. All application names containing spaces must be enclosed in single quotation marks.

EXAMPLE:

```
APPLICATION   'CSLD Mail Demo'
```

> **Note:**
> This keyword refers to the corresponding **ARCHIVE** statement and is used only for CMOD archives.

ARCHIVE *archive_ID*

archive_ID specifies the logical archive ID (e.g. A1). The archive ID must be unique. It is used by CMCS to identify the requested archive. All keywords required to access this archive are combined in the so-called **ARCHIVE** statement. The keyword *STORAGETYPE* must be specified as the second keyword. All following keywords depend on the storage type. Keywords belonging to different storage types must not be combined in a single **ARCHIVE** statement.

A default archive could be specified.

An example storage type is shown below.

```
ARCHIVE  A2
  STORAGETYPE                   VI
  LIBSERVER                     LIBSERV2
  INDEX_CLASS                   TestClass
  VIUSER                        FRNUSER
```

> **Note:**
> You must not use `VI` as an archive ID. In general, keywords of the CMCS configuration file `archint.ini` must not be used as names.

> **Note:**
> Once set up, these settings should be saved and not changed; this is because any restore operations will depend upon them.

BINPATH *path*
Specifies the complete path to the CMCS server binary files.

EXAMPLE:

```
BINPATH   c:\ibm\csld\ (Windows)
```

> **Note:**
> The binaries *must not* be renamed. Further, it is expected that they reside in a *single* directory.

CHECK_ARCHIVE_SERVER *ON|OFF*
Specifies whether or not the CMCS server will start up in case an archive is not available. When set to **ON**, all archives for the configured agents must be available. and have the mandatory attributes defined at startup; otherwise, CMCS will refuse to start. When set to **OFF**, CMCS prints a warning if an archive is not available; nevertheless, it will continue to start up. The default is **ON**.

> **Important:**
> For Lotus Notes archiving, this parameter must be set to **OFF**.

COMMENT
This is an additional keyword for the **ARCHIVE** statement. Using this

general CMCS keyword, you can specify a comment for this archive ID. The comment is displayed by the HTTP command server info.

EXAMPLE:

```
COMMENT   'Archive is used for testing only'
```

CONFIG_FILE *filename*

Specifies the configuration file for the CMCS Server to store all variable parameters such as passwords, user names, and current version number. *filename* specifies the full path and the name of the file. This keyword is required.

EXAMPLE:

```
CONFIG_FILE   c:\ibm\csld\archint.cfg
```

Note:
The configuration file is encrypted.

DOMINODPS *number*

Specifies the number of parallel sessions for the CSLD dispatcher. If not specified, the default is **0**. This keyword must be present for CSLD use.

EXAMPLE:

```
DOMINODPS   2
```

DOMINOPORT *port*

Specifies the TCP/IP port for connecting to the CSLD dispatcher. This keyword must be present for CSLD use.

EXAMPLE:

```
DOMINOPORT   47111
```

END

Using this general CMCS keyword, you can specify the end of the parameter definitions. When *END* is encountered, the CMCS server stops searching its configuration profile for keywords.

ERRORLOG_FILE *filename*

Specifies a directory and file where all errors occurring in a CommonStore operation will be recorded. The error log file is a text file. The entries in the error log file consist of one section per failed operation. The first error in a failed operation is recorded in the error log file. The entries in the error log file contain the following information:

1. date and time when the error occurred

2. component where the error occurred (Tivoli Storage Manager, Content Manager, CMOD, R/3, etc..)
3. return code, extended return code if present, reason code if present
4. error description obtained from API or generated by Common

The error log file grows without size limitation.

DEFAULT
value of INSTANCEPATH + ″csserror.log″

EXAMPLE
ERRORLOG_FILE `c:\ibm\csld\log\srverror.log`

INDEX_CLASS *index_class_name*
Specifies the Content Manager index class which the CMCS server uses to archive the document content. This index class must be defined on the corresponding Content Manager library server.

EXAMPLE:

```
INDEX_CLASS   TestClass1
```

Note:
This parameter refers to the corresponding **ARCHIVE** statement and is used only for Content Manager archives.

INSTANCEPATH *path*
Specifies the directory where the instance related files (profile, config file, ...) are located. On AIX for every CommonStore server instance a user should be created. Then the INSTANCEPATH should point to the users home directory. All instance related files are maintained in the users home directory. On Windows for every CommonStore server instance a sub-directory should be created. The INSTANCEPATH of every single instance should point to the related subdirectory. All instance related files are maintained in these directories.

The error log file grows without size limitation.

DEFAULT
value of BINPATH

EXAMPLE
INSTANCEPATH `/home/csadm1`

LIBSERVER *server_name*
Specifies the name of the Content Manager Library Server, to which a connection with the subsequent definitions will be established. The `FRNTABLE` contains all necessary communication parameters for Content Manager.

EXAMPLE:

```
LIBSERVER   LIBSRVESD
```

> **Note:**
> This parameter refers to the corresponding **ARCHIVE** statement and is used only for Content Manager archives. It is needed only if the scenario of late archiving with barcode for Content Manager has to be supported.

> **Note:**
> Check to see if you have a valid `FRNTABLE`.

LOG *ON|OFF*

If set to **ON**, a log file containing information about all archived and retrieved data for each day will be created.

The log files are generated in the following format:

```
aiyyyymmdd.log
```

where

- `yyyy` = the year,
- `mm` = the month, and
- `dd` = the day.

LOGPATH *path*

path defines the complete path of the log file. The log files' file names are generated automatically.

EXAMPLE:

```
LOGPATH   /usr/lpp/csld/bin/
```

MGMT_CLASS *management_class*

Using this TSM-specific keyword, you can specify the Tivoli Storage Manager management class which the CMCS server uses to archive the documents. The parameter string can consist of up to 16 characters. If there is no management class, it uses the Tivoli Storage Manager default management class.

Note:
This keyword refers to the corresponding **ARCHIVE** statement and is used only for Tivoli Storage Manager archives. Further, keep in mind that the Tivoli Storage Manager expiration date (specified under the management class in the Tivoli Storage Manager archive storage pools) should be checked; this is because Tivoli Storage Manager automatically deletes all files as soon as the expiration period is reached.

MULTIPART *ON/OFF*

Applies to Content Manager/VisualInfo archives only. Specifies if documents should be stored in the Content Manager server in multiple parts or not. It is recommended to store the document in one part, as documents in multiple parts can cause problems, when displaying them in the Content Manager viewer. Setting the value of MULTIPART to OFF or NO will store any document in one single part. Setting the value of MULTIPART to ON or YES stores documents in multiple parts in the Content Manager server.

DEFAULT

OFF

EXAMPLE

MULTIPART ON

ODAGENTS *number*

Using this CMOD-specific keyword, you can specify the maximum number of parallel CMOD sessions (name: `archagentod`). If not specified, the default is **0**.

EXAMPLE:

```
ODAGENTS   1
```

ODHOST *hostname*

Using this CMOD-specific keyword, you can specify the host name or IP address of the CMOD library server.

EXAMPLE:

```
ODHOST   asterix
```

Note:
This keyword refers to the corresponding **ARCHIVE** statement and is used only for CMOD archives.

ODUSER *username*

Using this CMOD-specific keyword, you can specify an CMOD user permitted to view, add, and delete documents contained in the application group specified using the keyword *APPGROUP* and who has access to the folder specified using the keyword *FOLDER*.

EXAMPLE:

```
ODUSER   admin
```

> **Note:**
> This keyword refers to the corresponding **ARCHIVE** statement and is used only for CMOD archives.

PROTECTION *prot_flags*|*OFF*

This is an additional keyword for the **ARCHIVE** statement. Using this general CMCS keywords keyword, you can specify the default protection for this archive ID. `prot_flags` is a combination of the letters 'r' (read), 'c' (create), 'u' (update) 'd' (delete). The default is `rcud`.

When set to **OFF**, this archive is not protected, i.e. all operations are allowed. Typically, **OFF** is used in connection with the CMCS Web Access.

EXAMPLE:

```
PROTECTION       rcu
```

In this example, READ, CREATE, and UPDATE are enabled, while DELETE is *not* enabled.

REPORT *ON*|*OFF*

If set to **ON**, the CMCS server produces some additional information. The output is written to `stdout`, which is normally the console. If not specified, the default is **OFF**.

> **Note:**
> **ON** should be used for tracing purposes (e.g. setup of the CMCS server or in case of errors).

SERVER *server_name*

Using this TSM-specific keyword, you can specify the name of the Tivoli Storage Manager server to which a connection with the subsequent definitions will be established. The `dsm.sys` file contains all necessary communication parameters for Tivoli Storage Manager.

EXAMPLE:

```
SERVER   ADSMSERV01
```

> **Note:**
> This keyword refers to the corresponding **ARCHIVE** statement and is used only for the Tivoli Storage Manager archives.

SERVICE_TRACEFILE *filename*
Specifies an additional trace file to record the startup and shutdown of the CMCS service. Useful only for analyzing problems with the CMCS service on Windows NT. If not specified, no service trace file is written.

STARTUP_TRACEFILE *filename*
Specifies the full file name of the so-called startup trace file. When a non-empty file name is specified, all CMCS executables record messages during the initial startup phase in this file. This trace file is very useful in case of initial communication problems among the server executables. For all other problems, it is typically of no help. If not specified, no startup trace file is written.

EXAMPLE:

```
STARTUP_TRACEFILE   c:\ibm\csld\startup.trace
```

> **Note:**
> The startup trace file is re-written on each start of the CMCS server.

STORAGETYPE *VI*
Defines whether or not documents assigned to the specified logical archive are to be stored in Content Manager.

EXAMPLE:

```
STORAGETYPE   VI
```

> **Note:**
> This parameter belongs to the **ARCHIVE** statement and is used for all archives. Any further archive parameters which you specify depend upon this setting.

SYSTEMTYPE *SAPSYSTEM|DOMINOSYSTEM*
Specifies whether CommonStore is used for SAP and/or Lotus Domino. These settings affect only the LUM licensing part.

Default:

both SAPSYSTEM and DOMINOSYSTEM

EXAMPLE:

```
SYSTEMTYPE   SAPSYSTEM
```

TEMPPATH *path*

Specifies the directory in which the CMCS server writes temporary files needed for processing.

If this setting is missing in your profile, the environment variable *TMPDIR* will be checked. If this variable is not set either, the temporary files will be written to the system's temporary directory.

EXAMPLE:

```
TEMPPATH   c:\temp
```

TRACE *ON|OFF*

If set to **ON**, the CMCS server writes trace information into the trace file.

If you specify the following parameters, you can force the CMCS server to write only specific trace information:

- *DISP* (for information on the CMCS dispatcher program)
- *FILEIO* (for information on the input/output file activities)
- *ARCHPRO* (for information coming from the CMCS `archpro` program)
- *AGENTS* (for information coming from the Content Manager agents)
- *DEBUG* (for debug information on the CMCS server)

The value **ON** includes all of these parameters except *DEBUG*.

Examples:

```
TRACE   ON

TRACE   ARCHPRO DEBUG
```

Note:

This parameter should be used only for the purpose of detecting problems. If not specified, the default is **TRACE OFF**. Do not delete the trace file while the CMCS server is running as this will affect further writing to this file.

TRACEFILE *filename*

Specifies the trace file for the CMCS server, where all the trace information will be stored. *filename* specifies the full path and name of the file. This setting will be used only if tracing has been activated. If not specified, the default is `archint.trace`.

EXAMPLE:

```
TRACEFILE   c:\ibm\csld\archint.trace
```

> **Note:**
> It is not allowed to delete any trace file while the CMCS server is running. Rather, the CMCS server must first be stopped using the **archstop** command.

TRACEMAX *number*

Specifies the maximum size (in KB) of the CMCS server trace file.

EXAMPLE:

```
TRACEMAX   500
```

VIAGENTS *number*

Using this Content Manager-specific keyword, you can specify the total number of parallel Content Manager Client sessions (name: `archagentvi`) which the CMCS server establishes. If not specified, the default is **0**.

EXAMPLE:

```
VIAGENTS   3
```

VIUSER *username*

Using this Content Manager-specific keyword, you can specify the Content Manager user name for the Content Manager log-in procedure.

> **Note:**
> This keyword refers to the corresponding **ARCHIVE** statement and is used only for Content Manager archives.

WEBDPS *number*

Specifies the total number of parallel sessions for the CMCS Web Access. If not specified, the default is **0**.

EXAMPLE:

```
WEBDPS   5
```

> **Note:**
> By default, no Web Access to the CMCS archive is possible. To enable Web Access, the keyword *WEBDPS* must be specified.

WEBPORT *port*

Using this general CMCS keyword, you can specify the TCP/IP port number used to access the Web dispatcher using a Web browser. The port number must match the port number specified under the `~archive` setting in the Web Access service file for the SAP Internet Transaction Server (ITS). .

EXAMPLE:

```
WEBPORT   5501
```

> **Note:**
> The HTTP protocol used for communication with the Web dispatcher uses the TCP/IP port **8085** by default. If no Web server is running on the machine on which the CMCS server is running, the default TCP/IP port **8085** can be used.

WEBROOT *path*

Specifies the directory where the HTTP interface of the CommonStore server expects the files necessary for the Web operations, e.g. Browser Viewing.

EXAMPLE:

WEBROOT `/home/csadm1/webroot`

Appendix B. CMCS Server Commands

archadmin

This program allows a connection to a CMCS server to be opened in order to view the messages issued by the CMCS server. It is possible to open the connection across machine and platform boundaries.

Usage

archadmin [-m <machine>] [-p <port>] [-i <ini file>] [-h]

Options

-m specifies the machine or IP address where archpro is running

-p specifies the fixed port used by archpro

-i specifies the full ini file name used by archpro (local machine only)

-h show this help

Remarks

The connection to archpro is established using the specified (remote) machine and fixed port. When no machine is specified, localhost is assumed. The fixed port could also be read from the specified (local) ini file. When neither *-p* nor *-i* is specified, the standard (local) ini file is read. Only port numbers above 5000 are accepted. Connections between different machine types (Windows and AIX) are supported!

Examples

archadmin
connects to archpro using port from standard ini file

archadmin -p 5510
connects to archpro running on fixed port 5510

archadmin -m obelix -p 5510
connects to archpro running on 'obelix' on port 5510

archadmin -m obelix -p 5510
connects to archpro running on 'obelix' on port 5510

archadmin -m 9.164.10.20 -p 5510
connects to archpro on '9.164.10.20' on port 5510

archadmin -i c:\ibm\csld\archint.ini
connects to local archpro using port from ini file

Note:
When using `archadmin` on a Windows NT machine on which the CMCS server is not installed, you must also copy the message catalog file `CSSrvMsg.dll` to the directory containing `archadmin`.

archpro

`archpro` is the continuously-running CMCS main program which controls all of the other CMCS components.

Usage

`archpro [-i ][-f serverpasswd | license][-n ][-h]`

Options

-i **`<ini file>`**
full file name of CMCS server ini file

-f **`serverpasswd [<srv> [<node> [<passwd>]]]`**
enter Tivoli Storage Manager, Content Manager, Content Manager OnDemand password(s) (setup, only)

-f **`license`**
enroll a production license

-n instance name of the CMCS service

-h show this help

Remarks

Specify the ini file name when CMCS is not installed in a single path.

It must be specified before any other option.

Examples

`archpro`
start the CMCS server with the default CMCS configuration profile

`archpro -i c:\ibm\csld\archint.ini`
start the CMCS server with the specified profile.

`archpro -f serverpasswd`
ask for all archive passwords

`archpro -f serverpasswd SRV`
ask password for server SRV and all nodes/users

archpro -f serverpasswd SRV USR
ask password for server SRV and node/user USR

archpro -f serverpasswd SRV USR PWD
accept password for server SRV and node/user USR

archpro –f license
ask for license certificate file containing a productive license and enroll the license

archservice (only on Windows NT)

This program contains the whole service functionality of CMCS on Windows NT.

Usage

archservice install [-i <ini file>] [-n <name>]
installs the service

archservice remove [-n <name>]
removes the service

archservice start [-n <name>]
starts the service

archservice stop [-n <name>]
stops the service

archservice status [-n <name>]
queries the service's status

archservice -h
show this help

Options

-i **<ini file>**
full file name of CMCS ini file

-n **<name>**
instance name of the CMCS service

Remarks

Specify the ini file name when CMCS server is not installed in a single path.

The instance name permits multiple installations of CMCS server on a single machine.

The corresponding service instance is labeled as `CommonStore_<name>`.

Examples

archservice install
installs CommonStore as a service

archservice install -n 2
installs CommonStore_2 as a service

archservice remove -n 2
removes the CommonStore_2 service

archservice start -n 2
starts the CommonStore_2 service

archservice stop -n 2
stops the CommonStore_2 service

archservice status -n 2
queries the status of the CommonStore_2 service

> **Note:**
> **archservice** must not be executed from the command line *WITHOUT* parameters. This way of executing **archservice** is restricted to internal NT calls.

archstop

This program completely stops CMCS server by means of a regular shutdown.

Usage

archstop [-p <port>] [-i <ini file>] [now] [-h]

Options

-p **<port>**
specifies the fixed port used by archpro

-i **<ini file>**
specifies the full ini file name used by archpro

now stop immediately without waiting for completion of jobs

-h show this help

Remarks

The connection to archpro is established using the specified fixed port. This fixed port could also be read from the specified ini file. When neither -p <port> nor -i <ini file> is specified, the standard ini file is read.

Only port numbers above 5000 are accepted.

Examples

archstop
stops archpro using the port from the standard ini file

archstop -p 5510
stops archpro on port 5510 after all jobs have been completed

archstop -p 5510 now
stops archpro on port 5510 immediately

archstop -i c:\ibm\csld\archint.ini
stops archpro using port from this ini file

Appendix C. Sample Profiles

In the following sections, the sample profiles for Tivoli Storage Manager, Content Manager, and Content Manager OnDemand are presented. If you decide to use archives of more than one type, you can for example adapt the sample file for Content Manager by adding Tivoli Storage Manager archives as described in the sample file for Tivoli Storage Manager.

CMCS server on Windows NT using Tivoli Storage Manager

```
##############################################################################
#                                                                            #
#                 Sample configuration profile for                           #
#                                                                            #
#       IBM Content Manager CommonStore for Lotus Domino 7.1                 #
#                                                                            #
#              on Windows using Tivoli Storage Manager                       #
#                                                                            #
#           Copyright IBM Deutschland Entwicklung GmbH                       #
#                                                                            #
##############################################################################

#----------------------------------------------------------------------------#
# TIP:
#
# If a directory or a filename in a path in this ini includes a blank
# you must enclose the path with single quotes (').
# Example:
#
# BINPATH  'C:\Program Files\csld'
#
#----------------------------------------------------------------------------#

#----------------------------------------------------------------------------#
# Path (directory) of the CommonStore binaries
#----------------------------------------------------------------------------#
BINPATH  'c:\ibm\csld\bin'

#----------------------------------------------------------------------------#
# Path (directory) of the CommonStore instance.
# Should point to the instance directory of the server installation.
# All instance related files will be located in this directory, e.g.
#  trace files, log files, config files, etc.
#----------------------------------------------------------------------------#
INSTANCEPATH            c:\ibm\csld\instance01

#----------------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
```

```
#------------------------------------------------------------------------#
REPORT ON

#------------------------------------------------------------------------#
# TRACE     DISP FILEIO ARCHPRO AGENTS DEBUG ADSM RFC ON OFF
# Switches trace levels
#
# Examples for usage:
# TRACE         ON
# TRACE         AGENTS RFC DEBUG
#------------------------------------------------------------------------#
TRACE OFF

#------------------------------------------------------------------------#
# Size of the trace file in KB
#------------------------------------------------------------------------#
TRACEMAX 500

#------------------------------------------------------------------------#
# LOG           ON|OFF
# Switches the writing of the log file on or off
#------------------------------------------------------------------------#
LOG ON

#------------------------------------------------------------------------#
# Path for temporary files
#------------------------------------------------------------------------#
TEMPPATH               c:\temp\

#------------------------------------------------------------------------#
# Turn the SAP-specific feature off which checks whether all index
# classes have the right structure for SAP R/3 archiving.
#------------------------------------------------------------------------#
CHECK_ARCHIVE_SERVER    OFF # ON|OFF


#------------------------------------------------------------------------#
# Archive ID definitions
#
# We just define one archive ID required for the CSLD_MAILDEMO management
# class.
# The archive is ADSM/TSM.
# The name of the management class is CSLD_MAILDEMO.
# The name of the library server is ADSMSERV.
# Log into the archive with user ID CSLD.
#------------------------------------------------------------------------#
ARCHIVE A1
STORAGETYPE ADSM
SERVER               ADSMSERV
MGMT_CLASS           CSLD_MAILDEMO
ADSMNODE             CSLD

#--------------------------------------------------------------------
# Number of parallel instances of the CommonStore child programs
#--------------------------------------------------------------------
```

```
ADSMAGENTS 2
VIAGENTS 0
ODAGENTS         0
DOMINODPS 5
WEBDPS 2
#------------------------------------------------------------------------#
# TCP/IP port numbers used by CommonStore.
# ARCHWIN_PORT is used by archstop
# DOMINOPORT is the port CommonStore server listens for requests
# from CSLD tasks. Must be identical to the port number in profile
# document of configuration database.
# WEBPORT is the port CommonStore server listens for HTTP requests.
# Must be identical to the port number in profile
# document of configuration database.
#------------------------------------------------------------------------#
DOMINOPORT 47111
WEBPORT 8085
ARCHWIN_PORT         8012

#------------------------------------------------------------------------#
# LUM
#------------------------------------------------------------------------#
SYSTEMTYPE DOMINOSYSTEM

#--------------------------------------------------------------------
# End of profile
#--------------------------------------------------------------------
END
```

CMCS server on UNIX using Tivoli Storage Manager

```
##########################################################################
#                                                                        #
#                Sample configuration profile for                        #
#                                                                        #
#       IBM Content Manager CommonStore for Lotus Domino 7.1             #
#                                                                        #
#               on UNIX using Tivoli Storage Manager                     #
#                                                                        #
#           Copyright IBM Deutschland Entwicklung GmbH                   #
#                                                                        #
##########################################################################

#------------------------------------------------------------------------#
# Path (directory) of the CommonStore binaries
#
# On HP-UX and Sun Solaris:
# BINPATH /opt/csld/bin
#------------------------------------------------------------------------#
BINPATH         /usr/lpp/csld/bin/

#------------------------------------------------------------------------#
# Path (directory) of the CommonStore instance.
# Should point to the home directory of the instance admin user.
```

```
# All instance related files will be located in this directory, e.g.
#  trace files, log files, config files, etc.
#-----------------------------------------------------------------------#
INSTANCEPATH    /home/csadm1/

#-----------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
#-----------------------------------------------------------------------#
REPORT          ON

#-----------------------------------------------------------------------#
# TRACE     DISP FILEIO ARCHPRO AGENTS DEBUG ADSM RFC ON OFF
# Switches trace levels
#
# Examples for usage:
# TRACE         ON
# TRACE         AGENTS RFC DEBUG
#-----------------------------------------------------------------------#
TRACE           OFF

#-----------------------------------------------------------------------#
# Size of the trace file in KB
#-----------------------------------------------------------------------#
TRACEMAX        500

#-----------------------------------------------------------------------#
# Path for temporary files
#-----------------------------------------------------------------------#
TEMPPATH        /tmp/

#-----------------------------------------------------------------------#
# LOG           ON|OFF
# Switches the writing of the log file on or off
#-----------------------------------------------------------------------#
LOG             ON

#-----------------------------------------------------------------------#
# Turn the SAP-specific feature off which checks whether all index
# classes have the right structure for SAP R/3 archiving.
#-----------------------------------------------------------------------#
CHECK_ARCHIVE_SERVER    OFF # ON|OFF


#-----------------------------------------------------------------------#
# Archive ID definitions
#
# We just define one archive ID required for the CSLD_MAILDEMO management
# class.
# The archive is ADSM/TSM.
# The name of the management class is CSLD_MAILDEMO.
# The name of the library server is ADSMSERV.
# Log into the archive with user ID CSLD.
#-----------------------------------------------------------------------#
ARCHIVE A1
```

```
STORAGETYPE ADSM
SERVER               ADSMSERV
MGMT_CLASS           CSLD_MAILDEMO
ADSMNODE             CSLD

#------------------------------------------------------------------
# Number of parallel instances of the CommonStore child programs
#------------------------------------------------------------------
ADSMAGENTS 2
VIAGENTS 0
ODAGENTS         0
DOMINODPS 5
WEBDPS 2
#-------------------------------------------------------------------------#
# TCP/IP port numbers used by CommonStore.
# ARCHWIN_PORT is used by archstop
# DOMINOPORT is the port CommonStore server listens for requests
# from CSLD tasks. Must be identical to the port number in profile
# document of configuration database.
# WEBPORT is the port CommonStore server listens for HTTP requests.
# Must be identical to the port number in profile
# document of configuration database.
#-------------------------------------------------------------------------#
DOMINOPORT 47111
WEBPORT 8085
ARCHWIN_PORT         8012

#-------------------------------------------------------------------------#
# LUM
#-------------------------------------------------------------------------#
SYSTEMTYPE DOMINOSYSTEM

#------------------------------------------------------------------
# End of profile
#------------------------------------------------------------------
END
```

CMCS server on Windows NT using Content Manager

```
#########################################################################
#                                                                       #
#                Sample configuration profile for                       #
#                                                                       #
#       IBM Content Manager CommonStore for Lotus Domino 7.1            #
#                                                                       #
#                 on Windows using Content Manager                      #
#                                                                       #
#           Copyright IBM Deutschland Entwicklung GmbH                  #
#                                                                       #
#########################################################################

#-------------------------------------------------------------------------#
# TIP:
#
```

```
# If a directory or a filename in a path in this ini includes a blank
# you must enclose the path with single quotes (').
#
# Example:
#
# BINPATH  'C:\Program Files\csld'
#
#-------------------------------------------------------------------------#

#-------------------------------------------------------------------------#
# Path (directory) of the CommonStore binaries
#-------------------------------------------------------------------------#
BINPATH  'c:\ibm\csld\bin'

#-------------------------------------------------------------------------#
# Path (directory) of the CommonStore instance.
# Should point to the instance directory of the server installation.
# All instance related files will be located in this directory, e.g.
#  trace files, log files, config files, etc.
#-------------------------------------------------------------------------#
INSTANCEPATH            c:\ibm\csld\instance01

#-------------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
#-------------------------------------------------------------------------#
REPORT ON

#-------------------------------------------------------------------------#
# TRACE     DISP FILEIO ARCHPRO AGENTS DEBUG ADSM RFC ON OFF
# Switches trace levels
#
# Examples for usage:
# TRACE         ON
# TRACE         AGENTS RFC DEBUG
#-------------------------------------------------------------------------#
TRACE OFF

#-------------------------------------------------------------------------#
# Size of the trace file in KB
#-------------------------------------------------------------------------#
TRACEMAX 500

#-------------------------------------------------------------------------#
# LOG           ON|OFF
# Switches the writing of the log file on or off
#-------------------------------------------------------------------------#
LOG ON

#-------------------------------------------------------------------------#
# Path for temporary files
#-------------------------------------------------------------------------#
TEMPPATH                c:\temp\

#-------------------------------------------------------------------------#
```

```
# Turn the SAP-specific feature off which checks whether all index
# classes have the right structure for SAP R/3 archiving.
#--------------------------------------------------------------------------#
CHECK_ARCHIVE_SERVER    OFF # ON|OFF


#--------------------------------------------------------------------------#
# Archive ID definitions
#
# We just define one archive ID required for the CSLD_MAILDEMO indexclass.
# The archive is VI.
# The name of the index class is CSLD_MAILDEMO.
# The name of the library server is LIBSVI.
# Log into the archive with user ID CSLD.
#--------------------------------------------------------------------------#
ARCHIVE A1
STORAGETYPE VI
INDEX_CLASS CSLD_MAILDEMO
LIBSERVER LIBSVI
VIUSER CSLD

#-------------------------------------------------------------------
# Number of parallel instances of the CommonStore child programs
#-------------------------------------------------------------------
ADSMAGENTS 0
VIAGENTS 2
ODAGENTS         0
DOMINODPS 5
WEBDPS 2

#--------------------------------------------------------------------------#
# TCP/IP port numbers used by CommonStore.
# ARCHWIN_PORT is used by archstop
# DOMINOPORT is the port CommonStore server listens for requests
# from CSLD tasks. Must be identical to the port number in profile
# document of configuration database.
# WEBPORT is the port CommonStore server listens for HTTP requests.
# Must be identical to the port number in profile
# document of configuration database.
#--------------------------------------------------------------------------#
DOMINOPORT 47111
WEBPORT 8085
ARCHWIN_PORT         8012

#--------------------------------------------------------------------------#
# LUM
#--------------------------------------------------------------------------#
SYSTEMTYPE DOMINOSYSTEM

#-------------------------------------------------------------------
# End of profile
#-------------------------------------------------------------------
END
```

CMCS server on UNIX using Content Manager

```
###############################################################################
#                                                                             #
#                 Sample configuration profile for                            #
#                                                                             #
#       IBM Content Manager CommonStore for Lotus Domino 7.1                  #
#                                                                             #
#                  on UNIX using Content Manager                              #
#                                                                             #
#            Copyright IBM Deutschland Entwicklung GmbH                       #
#                                                                             #
###############################################################################

#------------------------------------------------------------------------------#
# Path (directory) of the CommonStore binaries
#
# On HP-UX and Sun Solaris:
# BINPATH /opt/csld/bin
#------------------------------------------------------------------------------#
BINPATH         /usr/lpp/csld/bin/

#------------------------------------------------------------------------------#
# Path (directory) of the CommonStore instance.
# Should point to the home directory of the instance admin user.
# All instance related files will be located in this directory, e.g.
#  trace files, log files, config files, etc.
#------------------------------------------------------------------------------#
INSTANCEPATH    /home/csadm1/

#------------------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
#------------------------------------------------------------------------------#
REPORT          ON

#------------------------------------------------------------------------------#
# TRACE     DISP FILEIO ARCHPRO AGENTS DEBUG ADSM RFC ON OFF
# Switches trace levels
#
# Examples for usage:
# TRACE         ON
# TRACE         AGENTS RFC DEBUG
#------------------------------------------------------------------------------#
TRACE           OFF

#------------------------------------------------------------------------------#
# Size of the trace file in KB
#------------------------------------------------------------------------------#
TRACEMAX        500

#------------------------------------------------------------------------------#
# Path for temporary files
#------------------------------------------------------------------------------#
TEMPPATH        /tmp/
```

```
#-------------------------------------------------------------------------#
# LOG           ON|OFF
# Switches the writing of the log file on or off
#-------------------------------------------------------------------------#
LOG             ON

#-------------------------------------------------------------------------#
# Turn the SAP-specific feature off which checks whether all index
# classes have the right structure for SAP R/3 archiving.
#-------------------------------------------------------------------------#
CHECK_ARCHIVE_SERVER    OFF # ON|OFF


#-------------------------------------------------------------------------#
# Archive ID definitions
#
# We just define one archive ID required for the CSLD_MAILDEMO indexclass.
# The archive is VI.
# The name of the index class is CSLD_MAILDEMO.
# The name of the library server is LIBSVI.
# Log into the archive with user ID CSLD.
#-------------------------------------------------------------------------#
ARCHIVE A1
STORAGETYPE VI
INDEX_CLASS CSLD_MAILDEMO
LIBSERVER LIBSVI
VIUSER CSLD

#-------------------------------------------------------------------
# Number of parallel instances of the CommonStore child programs
#-------------------------------------------------------------------
ADSMAGENTS 0
VIAGENTS 2
ODAGENTS         0
DOMINODPS 5
WEBDPS 2

#-------------------------------------------------------------------------#
# TCP/IP port numbers used by CommonStore.
# ARCHWIN_PORT is used by archstop
# DOMINOPORT is the port CommonStore server listens for requests
# from CSLD tasks. Must be identical to the port number in profile
# document of configuration database.
# WEBPORT is the port CommonStore server listens for HTTP requests.
# Must be identical to the port number in profile
# document of configuration database.
#-------------------------------------------------------------------------#
DOMINOPORT 47111
WEBPORT 8085
ARCHWIN_PORT                  8012

#-------------------------------------------------------------------------#
# LUM
#-------------------------------------------------------------------------#
```

```
SYSTEMTYPE DOMINOSYSTEM

#-------------------------------------------------------------------
# End of profile
#-------------------------------------------------------------------
END
```

CMCS server on Windows NT using Content Manager OnDemand

```
#########################################################################
#                                                                       #
#                Sample configuration profile for                       #
#                                                                       #
#       IBM Content Manager CommonStore for Lotus Domino 7.1            #
#                                                                       #
#             on Windows using Content Manager OnDemand                 #
#                                                                       #
#           Copyright IBM Deutschland Entwicklung GmbH                  #
#                                                                       #
#########################################################################

#-----------------------------------------------------------------------#
# TIP:
#
# If a directory or a filename in a path in this ini includes a blank
# you must enclose the path with single quotes (').
#
# Example:
#
# BINPATH  'C:\Program Files\csld'
#
#-----------------------------------------------------------------------#

#-----------------------------------------------------------------------#
# Path (directory) of the CommonStore binaries
#-----------------------------------------------------------------------#
BINPATH  'c:\ibm\csld\bin'

#-----------------------------------------------------------------------#
# Path (directory) of the CommonStore instance.
# Should point to the instance directory of the server installation.
# All instance related files will be located in this directory, e.g.
#  trace files, log files, config files, etc.
#-----------------------------------------------------------------------#
INSTANCEPATH            c:\ibm\csld\instance01

#-----------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
#-----------------------------------------------------------------------#
REPORT ON

#-----------------------------------------------------------------------#
# TRACE     DISP FILEIO ARCHPRO AGENTS DEBUG ADSM RFC ON OFF
```

```
# Switches trace levels
#
# Examples for usage:
# TRACE         ON
# TRACE         AGENTS RFC DEBUG
#--------------------------------------------------------------------------#
TRACE OFF

#--------------------------------------------------------------------------#
# Size of the trace file in KB
#--------------------------------------------------------------------------#
TRACEMAX 500

#--------------------------------------------------------------------------#
# LOG           ON|OFF
# Switches the writing of the log file on or off
#--------------------------------------------------------------------------#
LOG ON

#--------------------------------------------------------------------------#
# Path for temporary files
#--------------------------------------------------------------------------#
TEMPPATH                c:\temp\

#--------------------------------------------------------------------------#
# Turn the SAP-specific feature off which checks whether all index
# classes have the right structure for SAP R/3 archiving.
#--------------------------------------------------------------------------#
CHECK_ARCHIVE_SERVER    OFF # ON|OFF


#--------------------------------------------------------------------------#
# Archive ID definitions
#
# We just define one archive ID required for the CSLD_MAILDEMO.
# The archive is ONDEMAND.
# The name of application, applicationgroup and folder is CSLD_MAILDEMO.
# The name of the library server is ODSERVER.
# Log into the archive with user ID CSLD.
#--------------------------------------------------------------------------#
ARCHIVE A2
    STORAGETYPE         ONDEMAND
    ODHOST              ODSERVER
    APPGROUP            'CSLD_MAILDEMO'
    APPLICATION         'CSLD_MAILDEMO'
    FOLDER              'CSLD_MAILDEMO'
    ODUSER              CSLD

#--------------------------------------------------------------------
# Number of parallel instances of the CommonStore child programs
#--------------------------------------------------------------------
ADSMAGENTS 0
VIAGENTS 0
ODAGENTS         2
DOMINODPS 5
```

```
WEBDPS 2
#-------------------------------------------------------------------------#
# TCP/IP port numbers used by CommonStore.
# ARCHWIN_PORT is used by archstop
# DOMINOPORT is the port CommonStore server listens for requests
# from CSLD tasks. Must be identical to the port number in profile
# document of configuration database.
# WEBPORT is the port CommonStore server listens for HTTP requests.
# Must be identical to the port number in profile
# document of configuration database.
#-------------------------------------------------------------------------#
DOMINOPORT 47111
WEBPORT 8085
ARCHWIN_PORT                  8012

#-------------------------------------------------------------------------#
# LUM
#-------------------------------------------------------------------------#
SYSTEMTYPE DOMINOSYSTEM

#-------------------------------------------------------------------
# End of profile
#-------------------------------------------------------------------
END
```

CMCS server on UNIX using Content Manager OnDemand

```
#########################################################################
#                                                                       #
#                Sample configuration profile for                       #
#                                                                       #
#       IBM Content Manager CommonStore for Lotus Domino 7.1            #
#                                                                       #
#              on UNIX using Content Manager OnDemand                   #
#                                                                       #
#           Copyright IBM Deutschland Entwicklung GmbH                  #
#                                                                       #
#########################################################################

#-------------------------------------------------------------------------#
# Path (directory) of the CommonStore binaries
#
# On HP-UX and Sun Solaris:
# BINPATH /opt/csld/bin
#-------------------------------------------------------------------------#
BINPATH         /usr/lpp/csld/bin/

#-------------------------------------------------------------------------#
# Path (directory) of the CommonStore instance.
# Should point to the home directory of the instance admin user.
# All instance related files will be located in this directory, e.g.
#  trace files, log files, config files, etc.
#-------------------------------------------------------------------------#
INSTANCEPATH    /home/csadm1/
```

```
#-------------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
#-------------------------------------------------------------------------#
REPORT          ON

#-------------------------------------------------------------------------#
# TRACE     DISP FILEIO ARCHPRO AGENTS DEBUG ADSM RFC ON OFF
# Switches trace levels
#
# Examples for usage:
# TRACE         ON
# TRACE         AGENTS RFC DEBUG
#-------------------------------------------------------------------------#
TRACE           OFF

#-------------------------------------------------------------------------#
# Size of the trace file in KB
#-------------------------------------------------------------------------#
TRACEMAX        500

#-------------------------------------------------------------------------#
# Path for temporary files
#-------------------------------------------------------------------------#
TEMPPATH        /tmp/

#-------------------------------------------------------------------------#
# LOG           ON|OFF
# Switches the writing of the log file on or off
#-------------------------------------------------------------------------#
LOG             ON

#-------------------------------------------------------------------------#
# Turn the SAP-specific feature off which checks whether all index
# classes have the right structure for SAP R/3 archiving.
#-------------------------------------------------------------------------#
CHECK_ARCHIVE_SERVER    OFF # ON|OFF


#-------------------------------------------------------------------------#
# Archive ID definitions
#
# We just define one archive ID required for the CSLD_MAILDEMO.
# The archive is ONDEMAND.
# The name of application, applicationgroup and folder is CSLD_MAILDEMO.
# The name of the library server is ODSERVER.
# Log into the archive with user ID CSLD.
#-------------------------------------------------------------------------#
ARCHIVE A2
    STORAGETYPE         ONDEMAND
    ODHOST              ODSERVER
    APPGROUP            'CSLD_MAILDEMO'
    APPLICATION         'CSLD_MAILDEMO'
    FOLDER              'CSLD_MAILDEMO'
```

```
    ODUSER              CSLD

#-------------------------------------------------------------------
# Number of parallel instances of the CommonStore child programs
#-------------------------------------------------------------------
ADSMAGENTS 0
VIAGENTS 0
ODAGENTS         2
DOMINODPS 5
WEBDPS 2
#-------------------------------------------------------------------------#
# TCP/IP port numbers used by CommonStore.
# ARCHWIN_PORT is used by archstop
# DOMINOPORT is the port CommonStore server listens for requests
# from CSLD tasks. Must be identical to the port number in profile
# document of configuration database.
# WEBPORT is the port CommonStore server listens for HTTP requests.
# Must be identical to the port number in profile
# document of configuration database.
#-------------------------------------------------------------------------#
DOMINOPORT 47111
WEBPORT 8085
ARCHWIN_PORT         8012

#-------------------------------------------------------------------------#
# LUM
#-------------------------------------------------------------------------#
SYSTEMTYPE DOMINOSYSTEM

#-------------------------------------------------------------------
# End of profile
#-------------------------------------------------------------------
END
```

Appendix D. CMCS Server Error Codes

CS_RC_OK (0)
Operation completed successfully (no error).

CS_RC_CLOSE_SOCKET (-1)
This return code indicates that the corresponding socket will be closed. Typically this does not mean that an error occurred.

CS_RC_CHILDINIT_FAILED (-3)
Initialization of a CommonStore child process failed. This indicates a startup problem of a CommonStore child process.

CS_RC_CHECKARCHIVE_FAILED (-5)
A CommonStore agent could not be started since the startup check of the corresponding archive failed.

CS_RC_VERSION_ERROR (-6)
CommonStore does not start since a child process with a wrong version has been detected. You must replace the corresponding executable by one of the correct version.

CS_RC_CHILD_TERMINATED (-10)
CommonStore has detected that a child process has terminated unexpectedly. The child process will be restarted automatically.

CS_RC_SHUTDOWN (-99)
CommonStore was shut down by a shutdown request from archstop.

CS_RC_SHUTDOWN_NOW (-100)
CommonStore was shut down by an immediate shutdown request from archstop.

CS_RC_NOMEM (-110)
A CommonStore process is running out of memory.

CS_RC_NOTFOUND (-115)
The requested data was not found. Please see CommonStore trace for further details.

CS_RC_ERRDELETE (-116)
An archived document or component could not be deleted. This error code could also occur when data is appended to a component or a component is updated.

CS_RC_NOTSUPPORTED (-118)
The requested operation is not supported. For example, an index transfer request sent to ADSM agent or an invalid action for an update operation.

CS_RC_FILENOTFOUND (-200)
A file or a document could not be found.

CS_RC_UNKNOWNDOC (-201)
The requested document was not found in the archive.

CS_RC_QUERYNOTFOUND (-202)
The document was not found in the archive (query request only).

CS_RC_ACCESSDENIED (-203)
Insufficient access rights for archived document.

CS_RC_DOCEXISTS (-204)
The document could not be archived since the same document already exists in the archive.

CS_RC_ERRCERT (-205)
Failure when administering a certificate.

CS_RC_COMPNOTFOUND (-206)
Issued in an append request when the corresponding component does not exist in the archive. When data is appended to a component it is necessary that this component already exists in the archive.

CS_RC_CONTREP_NOTFOUND (-207)
The specified content repository (archive ID) was not found in the CommonStore profile (ARCHINT.INI).

CS_RC_INVALIDOFFSET (-208)
Issued in a part retrieve request when the specified offset is beyond the end of document.

CS_RC_FREESEARCH_NOTFOUND (-210)
The specified pattern was not found in a free search request.

CS_RC_ATTRSEARCH_NOTFOUND (-211)
The specified pattern was not found in an attributed search request.

CS_RC_OK_VERSION1 (-213)
A document archived with CommonStore version 1 was found (no error).

CS_RC_READONLY (-214)
Document cannot be modified in the archive since it was stored with CommonStore version 1.

CS_RC_LOGSYS_NOTFOUND (-215)
The DESTINATION statement in the CommonStore ini file does either not contain the specified logical system or does not contain any logical system.

CS_RC_NO_ATTR_ARCHIVED (-216)
The plain document data were archived successfully, but all attributes

provided in the attribute list were dropped since the archive cannot store the provided attributes. Issued typically by the CommonStore TSM agent when processing an archive request with additional attributes created by CSLD.

CS_RC_NOCOPYGROUP (-217)
CommonStore cannot use the specified TSM management class due to a problem with the copy group.

CS_RC_SAP_ATTR_NOTALLOWED (-240)
An archive operation failed since the attribute list in the archive request contains one of the reserved SAP attributes. These SAP attributes are filled automatically by CommonStore and hence are not allowed to be specified a second time.

CS_RC_SAP_ATTR_MISSING (-241)
Certain SAP attributes required by CommonStore are missing in the index class or application group.

CS_RC_FILEOPEN_ERROR (-250)
Error when opening a file or requesting the status of a file.

CS_RC_FILEREAD_ERROR (-251)
Error when reading data from a file.

CS_RC_FILEWRITE_ERROR (-252)
Error when writing data to a file.

CS_RC_ADSM_ERROR (-260)
Some error has occurred in the CommonStore TSM agent, but the related TSM API call did not fail.

CS_RC_VI_ERROR (-261)
Some error has occurred in the CommonStore CM agent, but the related CM API call did not fail.

CS_RC_OD_ERROR (-262)
Some error has occurred in the CommonStore OD agent, but the related OD API call did not fail.

CS_RC_HTTP_REQUEST_WRONG_VERSION (-500)
The HTTP request sent to CommonStore HTTP dispatcher did contain a newer version. This request is not yet supported by the current CommonStore HTTP dispatcher.

CS_RC_HTTP_REQUEST_WRONG_METHOD (-501)
The HTTP request sent to CommonStore HTTP dispatcher did contain wrong HTTP method.

CS_RC_HTTP_REQUEST_MISSING_PARAMETER (-502)
The HTTP request sent to CommonStore HTTP dispatcher did not contain all (or empty) mandatory parameters.

CS_RC_HTTP_REQUEST_MISSING_ENTITY (-503)
The HTTP request sent to CommonStore HTTP dispatcher did not contain a body.

CS_DO_WRONG_SEARCHATTR (-1003)
The specified search attribute pattern is not correct.

CS_DO_ARCHIVELIST_EMPTY (-1004)
A search operation failed since the provided list of archive IDs is empty.

CS_DO_FOLDER_ISEMPTY (-1006)
The folder operation cannot be completed since the specified folder is empty.

CS_VI_RETRIEVE_ERROR (-1112)
The retrieve operation failed although the called API functions did not return an error. The error was detected by additional consistency checks.

CS_VI_TOO_MANY_HITS (-1114)
When searching a folder with the specified doc ID in the archive index class more than one hit was found.

CS_VI_PARTS_INCONSISTENT (-1116)
The part numbers returned by the CM API are inconsistent. There is a gap in the returned part numbers.

CS_VI_INDEXCLASS_NOTFOUND (-1118)
The specified index class was not found on the specified CM server.

CS_VI_CONTENTCLASS_NOTFOUND (-1120)
The specified content class was not found on the specified CM server.

CS_VI_NO_DATA_RETURNED (-1121)
An API function did not return the requested data. The API function itself did not fail, however.

CS_BC_RFC_ABORT_BY_USER (-7095)
Archbc processing was stopped by a user (no error).

CS_BC_RFC_ERROR_TABLE (-7096)
Creating the barcode entry table failed. CommonStore was unable to send the barcodes to SAP.

CS_BC_RFC_ERROR_INSERT (-7097)
Sending the barcodes to SAP was not successful, since the remote function call ARCHIV_BARCODE_INSERT_RFC resp. BAPI_BARCODE_SENDLIST failed.

CS_BC_RFC_ERROR_CONNECTION (-7098)
CommonStore could not open the connection to SAP. The remote

function calls ARCHIV_BARCODE_INSERT_RFC resp. BAPI_BARCODE_SENDLIST were not executed. No barcodes were sent to SAP.

CS_BC_RFC_NO_DATA_TO_SEND (-7099)
The barcode table was empty. There are no barcodes to be sent to SAP.

CS_VI_ITEM_IN_WRONG_INDEXCLASS (-7111
An item was requested to be re-indexed but the item did neither reside in the scan index class nor in the target component index class.

CS_VI_ARCHIVE_HAS_NO_SCAN_IC (-7125)
The operation failed since there is no scan index class defined for the specified content repository (archive ID).

CS_VI_ARCHIVE_HAS_NO_SCAN_WB (-7127)
The operation failed since there is no scan workbasket defined for the specified content repository (archive ID).

CS_VI_ARCHIVE_HAS_NO_ERROR_WB (-7128)
The operation failed since there is no error workbasket defined for the specified content repository (archive ID).

CS_RC_CWI_R3_CONNECTION_ERROR (-7201)
The create workitem request failed since CommonStore could not connect to SAP.

CS_RC_CWI_R3_FAILED (-7202)
SAP returned an error when receiving the create workitem request.

CS_RC_CWI_ARCHIVE_FAILED (-7203)
The create workitem request failed since the archive operation failed. The create workitem request was not sent to SAP.

CS_RC_SOCKET_PROBLEM (-10000)
A problem with the socket communication in CommonStore occurred. Please see CommonStore trace file for further details.

CS_RC_NO_HANDLER (-10001)
A CommonStore process received a request for which no message handler was installed.

CS_RC_INTERNAL_ERROR (-10002)
An internal error in CommonStore or in the socket communication occurred. Please see CommonStore trace file for further details.

CS_RC_WRONG_TYPE (-10003)
When parsing a CommonStore message sent over a socket a wrong data type is encountered.

Appendix E. The Mail Archiving Sample Application

CSLD is shipped with a sample mail archiving application that demonstrates most of the CSLD features. During CSLD installation, the template of this application is created as file "<installation directory>\data\CSLDStdMail.ntf". We took the standard mail template of Notes release 4.6, and added a number of design elements to it. For example, we added LotusScript libraries, a number of actions, views, folders and agents.

We chose the Notes R4.6 template because the R5 mail template does not run on Notes 4.6. The mail sample application does not make any use of binary code like LSX classes. Remember that all a CSLD-enabled application has to do is to create CSLD job documents, which, in this case, is completely done via LotusScript using the CSLD LotusScript libraries.

Do not expect the mail archiving demo to be a ready-to-use archiving application! The goal of this sample application is to exploit most of the features of CSLD, so technical details are not hidden from the user. For example, some dialogs might look a little "overloaded", and users might not understand what certain design elements are good for.

It depends heavily on your environment and requirements what archiving functionality should be implemented into a CSLD-enabled mail database. For example, some companies would require interactive mail archiving, where users simply have to click an "archive" button. Other companies would implement scheduled or event-triggered agents that archive only attachments, based on some selection criteria. Some companies would store their documents as TIFF documents for legal requirements, others would use the Notes native format in order to be able to process the document later in Lotus Notes. Also, most companies already use their own customized version of the mail template.

For these reasons, it is almost impossible to find a "least common denominator" for all requirements, and to create a ready-to-use archiving application that can be deployed immediately in your company. It is up to the CSLD application developer to define and implement the desired behavior of your mail archiving database.

To CSLD-enable your application, you do not have to start implementing everything from scratch. We have already implemented most standard scenarios of interactive archiving, so feel free to simply cut & paste the script code from the sample application to your database. There is no copyright on the sample code.

Note

Do not modify the code in script library *createCSNJobs*. If you do, we take no responsibility if your application does not work!

Getting started with the CSLD Sample Mail Application

To get started with this application, perform the following steps (see the documentation for details on each step):

- Install CSLD.
- Copy all templates from directory <installation path>\data into your Notes data directory., so they will appears in the template selection box when creating a new database.
- Create the mail archiving sample database from template *CSLDStdMail.ntf*.
- Create the job database from template *CSLDJobs.ntf*.
- Create the configuration database from template *CSLDConfig.ntf*.

Content Manager:

- Create an index class (e.g. ″Maildemo″) with (at least) the following attributes
  - ″MailSubject″ (variable character/extended numeric)
  - ″FromSender″ (variable character/extended numeric)
  - ″PostedDate″ (Timestamp)
  - CSLDOrigUser (variable character/extended numeric)
  - CSLDOrigDB (variable character/extended numeric)
- Create content types (data formats) for every document type you want to archive.
- For the Notes folder archiving feature, create the Notes folder index class as described in “Preparing Content Manager and OnDemand for Notes folder archiving:” on page 119.

OnDemand:

- create an application group (e.g. ″Maildemo″) with (at least) the following attributes:
  - ″MailSubject″ (variable character/extended numeric)
  - ″FromSender″ (variable character/extended numeric)
  - ″PostedDate″ (timestamp)
  - CSLDOrigUser (variable character/extended numeric)
  - CSLDOrigDB (variable character/extended numeric)

  - the required attributes as listed in Table 3 on page 118.
- For the Notes folder archiving feature, create the Notes folder application group as described in "Preparing Content Manager and OnDemand for Notes folder archiving:" on page 119. Name it "NF" (Notes folders).

Tivoli Storage Manager

- Since TSM does not provide an index that allows to search for documents in the archive, you cannot delete the original Notes mails after archiving. The search form cannot be used with TSM.

  For attachment archiving, you must keep the document that contained the attachments. For all other formats, you must leave a document stub of the archived Notes document (see "Chapter 2.4 Archiving and retrieval tasks" on page 27 for details). This stub contains a link to the archived document, and will allow you to retrieve the archived content.

  The only configuration step is to create a management class.

- Copy file *archint_CSLD_sample.ini* to archint.ini.
- In file *archint.ini*, define the archive ID "MD" (Maildemo), pointing to the Maildemo CM index class, OD application group or TSM management class, respectively
- For the folder archiving feature, define an archive ID "NF" (Notes Folders), pointing to the Notes folder index class.
- In the CSLD configuration database, create a document mapping for form Memo: Map Notes documents of form "Memo" to archive ID "MD".
  - In field *Notes field to display in hitlist*, enter "MemoShell".This is the name of the form retrieved documents will be displayed with.
  - Add "Reply,Document" to the aliases field (so documents of Form "Reply" and "Document" will be archived as a regular email/Memo).
  - Map Notes field "Subject" to VI attribute "MailSubject".
  - Map Notes field "From" to VI attribute "FromSender".
  - Map Notes field "PostedDate" to VI attribute "PostedDate".
  - Enter "Subject; From; PostedDate" as the representative Notes fields to be displayed in hitlists.
- Create an archiving and retrieval database profile in the configuration database:
  - For easiest setup, select "All jobs in job database" (for mail archiving in a real environment you would select the second option and enter a particular mail server).
  - Enter name and server of your job database.

- Set the polling interval to once a second, every weekday (for fastest response times), from 1 am till 11:59 pm.
- Create a directory for temporary CSLD files (e.g. C:\temp\commonstore) and enter this directory in the *Export Directory* field.
- Turn on the two security features ″restore to original database only″ and ″retrieve by original user only″, so that CSLD users cannot retrieve other user's mail documents.
- In the Notes folder field, enter ″NF″. Notes folders will be stored in the archive defined by archive ID ″NF″.
- Set field *CommonStore host name* to ″localhost″ (assuming that the CS server runs on the same machine). Make sure the DOS shell command ″ping localhost″ succeeds.
- Set field *CommonStore Web port* to 8085, so that the browser viewing feature will contact the CMCS HTTP dispatcher at this port.
- Set field *CommonStore TCP/IP port* to 47111, so that the task knows the port to send requests to.
- In file *archint.ini*, set the DOMINODPS parameter to 47111 (should be set to that value already).
- In the ″Advanced″ section, set the trace level to ″All″, and set trace file size to at least 2 MB. The traces will help you discover problems with your setup.

- In the configuration database, create content type mappings for all possible file extensions of attachments that you want to archive
- Create the required content type mappings for file extensions .csn, .rtf and .txt
- In the job database ACL, add the CSLD user ID, and assign it the [CSLDUsers] role.
- Tell the mail sample database where to create job documents:
  - In function *JobDatabaseName* of script library *createCSNJobs*, enter the name of your job database.
  - In function *JobDatabaseServer* of script library *createCSNJobs*, enter the server hosting your job database.
- For the CS server (archpro), set license and archive password(s).
- Start the archpro executable.
- Start CSLD archive and retrieve task:

  In the binary directory of your installation path, you will find the two files demoarchive.bat and demoretrieve.bat. Adjust these scripts to contain your profile names, configuration database name/server, and optionally a Notes ini file.

  Run the two scripts.

- Perform actions in the mail archiving database (click archive and retrieve buttons).
- **Important: If you run into problems, please read the documentation (especially the troubleshooting section) before you consult the CommonStore support!** You will find answers for almost every standard problem.

Changes and additions made specifically to enable this database for CSLD

The Memo Form

In order to call some of the CSLD functionality directly from within an open Memo document, a number of actions have been added to the Memo form. When you open a Memo that has not yet been archived you will find an *"Archive"* button (for more detailed information on what happens when this button is pressed, see "Archive Selected Documents" below). Documents that have been archived will show a *"Retrieve"* button and possibly a *"Show Job"* button. The *"Show Job"* button will open up the job document that processed this Memo and can be used e.g. to find out the reason for a failed archiving request.

The code in *"Show Job"* makes use of the *CSLDJobUNID* item that CSLD adds to an archived document in case of failure.

The Inbox Folder

Most actions can be triggered from the Inbox Folder. The CSLD MailDemo template defines the following actions, which can be found in the menu under **Actions->CommonStore**. The code that will be executed can for most actions be found in script library *"CSNJobSamples"*In addition to the standard Inbox columns, there are three additional columns. The first two columns allow to categorize and order the documents by their state, i.e. non-archived documents will appear in Category *"Normal"* while archived documents will go to category *"Archived Notes"*. The third column displays different icons for each of the archiving request types:

Red Ball
: An archiving attempt failed.

Document with pencil
: The document was successfully archived in Notes native format

Document with check mark
: The document's attachments have been successfully archived. The paper clip icon next to the archived icon signals whether the attachment was removed after archiving.

Document with glasses
The document was archived in raster format (i.e. as RTF, ASCII file or as TIFF using Compart DocBridge).

The following sections will explain the actions that can be selected.

Archive Selected Documents

This action will pop up a dialog box allowing the user to select:

- the archiving format
- whether to remove the document after successful archiving from Notes
- whether to have CSLD detach archived attachments from the container document
- whether to leave the job in the job database after successful job completion

The dialog box is defined as hidden form *"(ArchiveDialog)"* (see below). According to the selections made in the dialog box, either a single archiving job containing all documents to archive or, in case of attachment archiving, one archiving job for each document containing one or more attachments will be created in the job database defined in script library *"CreateCSNJobs"*. We highly recommend to take a look at the code of this function to get an impression how easily CSLD jobs can be created.

Form (ArchiveDialog)

The form defines radio buttons to select the kind of archiving to perform on the selected documents. Each selection stands for a single request type. With the check boxes displayed below the user can select whether or not the archived document will be removed from the Notes database and in case of attachment archiving whether or not to remove the archived attachment from its originating document (checkbox is activated by default). The last check box *"Leave job in job database"* lets the user choose if the job document should remain in the job database even when the job has finished successfully.

Archive All Documents in View/Folder

The action basically does the same as the *"Archive Selected Documents"* action described above. But instead of creating a job with every single document contained in the view or folder, it will only pass the view/folder's universal identifier (UNID) as a job parameter. To determine the UNID of the current view/folder, we used a little trick: The PostOpen event of the Inbox folder stores the folder's UNID in a global variable, which is then used in function *ArchiveSelectedDocuments*. When copying this action to another database, make sure you do not forget the PostOpen code. In contrast to action "Archive current Notes folder structure", this action does not preserve the folder structure. That is, all documents are archived as individual documents.

Retrieve Selected Documents

This action will create a retrieve job for all the documents selected from the Inbox folder. Of course this will only work when the original document (or a stub of this document) is left in the database. Depending on the archiving type, the retrieve request will have different results: If the document's attachments were archived the retrieve request will reattach them back to the original document. For documents archived in raster format, a result document of type *"MemoShell"*will be created. For documents archived in native format, a copy of the original document will be restored to the database. For more information on retrieving documents from the mail sample application, see the description of agent *"Create stubs from Native Documents"*.

Update Index Information

This action will create an update job containing the UNIDs of all documents selected in the Inbox that have been successfully archived (i.e. they can be found in the *"Archived Notes"* category). CSLD does not support updating documents that have been archived in Notes native format.

Move Selected Documents to Workbasket

This action takes all documents that have been successfully archived, and moves them to a workbasket. A dialog box pops up asking for the target workbasket name. This feature is not supported for TSM. For OnDemand, the workbasket name will be a "virtual" one.

Search in Archive

This action will open a new document using form *"Query for 'Memo'"*, i.e. a CSLD query form. Filling in the search fields in this form and executing the *"create Query job"* action will create a search request.

Delete Selected Documents in the Archive

For every selected Notes document that has been successfully archived, the corresponding archived documents will be deleted from the archive. For every document one delete job will be created. *The CSNDArchiveID* item will be removed from the selected documents. Thus, upon successful deletion from the archive, and after pressing F9 to refresh the view, the documents should move to the *"Normal"* category. All archive state icons will disappear.

Remove Selected Documents from Workbasket

Removes all documents that have been archived successfully from their current workbasket. The user does not have to know in which workbasket the document resides. If a document currently does not reside in a workbasket, the job completes without any action. The action creates an update job of request type CSN_REMOVE_FROM_WORKBASKET, containing the document IDs of all selected documents.

List Documents in Workbasket

This action pops up a dialog box asking for the name of a workbasket, and creates a ″list workbasket″ job, containing the workbasket name. Returns a hitlist with all the documents in the workbasket. The hitlist (or the multiple result documents) will be displayed in the search and retrieve results view.

Since CSLD can support multiple archive servers, one parameter to the list workbasket request is the archive ID (defined in archint.ini) that specifies the CM server with the desired workbasket. For simplicity, the script behind this action assumes the workbasket to be on ″hardwired″ archive ID ″SM″. Please adjust this value if you want to list workbaskets on a server with a different archive ID. You could also write code that pops up a dialog window asking for the server archive ID on which to list the workbasket.

Archive Current Notes Folder Structure

This action archives the current view or folder with all its subfolders, and stores the folder structure in the archive. The Inbox folder has no subfolders. To test this feature, switch to folder ″RootFolder″ which has a subfolder named ″SubFolder″. Both folders inherit their design from the Inbox folder, so both have the same set of actions. When you delete documents from the folder structure, and you retrieve them back via action ″Restore Complete Folder Structure″, the documents will be retrieved to their original position within the folder structure. When you remove a number of subfolders of the original folder structure, it will be restored starting from the folder from which you created the request. Suppose you archive ″RootFolder″. All documents in ″Subfolder″ will be archived as well. Suppose you switch to ″Subfolder″ and click ″Restore Complete Folder Structure″. Then, CSLD will restore the documents in ″Subfolder″ only. If you have deleted ″SubFolder″, and you restore ″RootFolder″, then ″SubFolder″ will be recreated. Don’t forget to close and reopen the database to make the new folder visible.

The code creates an archive job containing the UNID of the folder you want to archive. The job has the ″preserve folder structure″ flag set.

Restore Current Notes Folder Structure

This action assumes that you have archived a folder structure using the ″Archive current Notes Folder Structure″ action, and that you have not removed the root folder of the folder structure from Notes.

Restores a complete folder structure by ID. That is, this action reads the document ID from the archived folder, and restores all documents and subfolders in this folder. A retrieval job with the folder ID is created. If you have deleted the folder and you do not have its parent folder available, you can still restore it by name

Restore Notes Folder Structure by Name

This action assumes that you have archived a folder structure using the ″Archive current Notes Folder Structure″ action, and that you have deleted the folder from Notes after archiving. In this case, since you can not read the folder's ID, the folder must be retrieved by name. A dialog box pops up, asking for the name of the (sub) folder to restore. A retrieve job is created with the folder name in it. You can also retrieve only a subfolder of the folder you archived. Use syntax ″folder\subfolder\subsubfolder″ to specify a particular folder

Important:
Since new folders are created in the database, you must close and reopen the database to make them visible.

″Create Stub from Native Document″ agent

Creates a stub from a document that has been archived in Notes native format. This makes especially sense when archiving documents in native format into TSM.

Since TSM has no indexing, there is no way to search for archived documents in the archive. However, with this agent, you can leave a stub of an archived document in Notes. Since stubs still contain a few descriptive fields of the original document, you can search for archived documents within Notes, and retrieve a number of stubs.

When clicking the ″*Retrieve*″ button, the archived document is retrieved. When the Retrieve button is clicked from the Inbox view, a copy of the archived document is retrieved, which can be found in the ″*Search & Retrieve Results*″ view. When the Retrieve button is clicked within an open stub document, and the document was archived in notes native format, the stub document is overwritten with the entire document. However, you must close and reopen the document to see that the stub has become the original document.

This agent should only be invoked via a post archiving agent. Do not invoke it manually. See “Chapter 3.4 Agents for processing archived and retrieved documents” on page 55 for details on stubs.

″Create Stubs from Native Documents Manually″ agent

This agent runs on all documents contained in the hidden view ″*(Native Archived Documents)*″ and creates stubs of them. The difference from ″Create Stub from Native Document″ is that this agent is invoked manually, while the other agent is invoked via a CSLD post archiving agent. See “Chapter 3.4 Agents for processing archived and retrieved documents” on page 55 for details.

In a productive environment, you would automatically create stubs via a post archiving agents. However, automatic things are hard to demonstrate, so for demonstration purposes (e.g. customer presentations) we have provided this agent. Archive your documents in native format, and show the difference of before and after invocation of this agent. Then, restore the original document by pressing the *"Retrieve"* button in an open stub document. Close and reopen the document to see that the stub has become the original document.

The "Archive Profiles" View

The *"Archive Profiles"* view is intended to provide the end-user with a simple means of defining his or her own archiving strategy. The view contains only documents created with form *"CSLDArchiveProfile"*, plus the actions to define new archiving profiles or the selected ones. Action *"Run Archive Agent"* is included for demo purposes and manually starts agent *"CSLDMailArchAgent"*. You could also trigger this agent automatically.

CSLDArchiveProfile form

By creating documents from this form a simple archiving strategy for the end-user can be created. The form defines fields to enter archiving criteria *("Archive mail older than x days.")* as well as an archiving type. Documents created with this form provide the *"CSLDMailArchAgent"* with its selection criteria and the parameters used to automatically start the background archiving of mail documents.

CSLDMailArchAgent agent

This agent reads the CSLDArchiveProfile documents, loops over all new and modified documents since its last run and archives them according to the criteria specified in the profile. In a working environment this agent should be run on a scheduled basis, e.g. once a day. Per default it is run manually.

The "Queries" View

To search for documents in the archive, you create a document of form "Query for Memo".

Since this is a regular Notes document, you can save frequently used queries. All queries are displayed in the Queries view.

The "Search & Retrieve Results" View

This view will be used to display the documents resulting from CSLD retrieve and search requests. The view is categorized by the user that issued the request and by the timestamp when the request was issued. In the first column the document type of the resulting document is displayed. In the mail demo there are basically three document types:

MemoShell:	When the retrieved document content is not in Notes native format and is not written to a target document, a result document of this type is created.

Hitlist:	When a query returns more hits than the number of documents directly built or no hits at all, a hitlist document is returned.
Memo, Reply, ...:	For all documents archived in native format, the document will be completely reconstructed.

The "*Search & Retrieve Results*" view defines actions to define archive queries, retrieve selected documents, update index information of archived documents, and delete documents in the archive. You will find these actions in the toolbar or the Actions menu.

The "Archived Document" and "Non-archived Documents" Views

These two views are included in the mail demo just for browsing purposes. They do not define any new actions. You could use an agent on the "Non-archived Documents" views to archive documents that have not been archived yet.

Have fun using our sample application. We highly appreciate any feedback. Just write email to

```
cstore@de.ibm.com
```

Appendix F. Frequently Asked Questions

Question: We have database replicas distributed over several Domino servers. When I archive documents with the ″retrieve documents to original database, only″ feature enabled, can I restore them to one of the replicas?

Answer: Yes, as long as those replicas are server replicas, you can archive from one replica and retrieve to another one.

Question: Can I run multiple instances of CMCS on one server?

Answer: Yes, you can. CMCS can be started in multiple instances on the same system. You only have to take care of distinct values of some ini parameters (e.g. *port* and *trace / log files* should be different). Thus, you must use distinct parameters for all used ini files. The executables can be unique and do not have to be copied.

Question: Is Lotus Domino R5 supported, too?

Answer: Yes, it is.

Question: Is CSLD DBCS-enabled?

Answer: Yes. Lotus Notes itself fully supports DBCS (double-byte character sets) on document/item base. CMCS's internal communication is done in Unicode.

Question: When archiving in the Notes native format, will the document's UNID be restored when I retrieve an archived document?

Answer: Yes, if no document with the original UNID has been created in the meantime.

Question: What does ″Folder archiving″ mean?

Answer: In terms of CSLD, *folder archiving* means archiving all documents residing in a certain folder within a Notes database. The documents to be archived are identified through the folder containing them rather than each separately by its UNID.

Question: Does CSLD always use the ″Notes″ content type for natively archived documents?

Answer: CSLD will use whatever content class a certain file extension was mapped to in the configuration database. This means that the administrator is responsible for mapping the correct files to their respective content classes. For natively archived Notes documents, there is no predefined content class in Content Manager. The administrator will therefore have to create a new one and map that to the file extension `csn` in the CSLD configuration database, thereby making sure that all natively archived Notes documents will be archived into this particular content class.

Question: While starting up a task, or during job processing, I (sometimes) receive the error message, ″Cannot establish connection to CommonStore Server...″, but most of the jobs are processed correctly. What should I do?

Answer: There are two possibilities:

1. Probably, the CMCS server (`archpro`) has not been started.
2. The number of CSLD dispatchers (keyword *DOMINODPS* in file `archint.ini`) is too low. Increase the number step by step until you do not receive any more messages.

List of Abbreviations

The abbreviations used in this document are listed below.

ACL	Access Control List
ADK	Archive Development Kit
ADSM	ADSTAR Distributed Storage Manager
AIX	Advanced Interactive Executive (IBM implementation of UNIX)
ALF	Advanced List Format
API	Application Program Interface
CM	Content Manager
CMCS	Content Manager CommonStore
CMOD	Content Manager OnDemand
COLD	Computer Output on Laser Disk
CSLD	CommonStore for Lotus Domino
DLL	Dynamic Link Library (files with the extension `.dll`)
ECL	Edit Control List
GUI	Graphic User Interface
IBM	International Business Machines Corporation
ITS	Internet Transaction Server
NFS	Network File System
NT	New Technology (Microsoft operating system Windows NT)
OCR	Optical Character Recognition
OD	OnDemand
OLE	Object Linking and Embedding
OTF	Output Text Format (files with the extension `.otf`)
PDF	Portable Document Format (files with the extension `.pdf`)
R/3	SAP R/3 system
RTF	Rich Text Format (files with the extension `.rtf`)
S/390	System/390
SAP	Systems, Applications, Products in Data Processing (software vendor)
TCP/IP	Transmission Control Protocol/Internet Protocol
TIFF	Tag Image File Format (files with the extension `.tif`)
TSM	Tivoli Storage Manager
UNID	Universal Notes Identifier
UNIX	An operating system developed at Bell Laboratories
URL	Uniform Resource Locator

Index

L

S

T

U

W

Readers' Comments — We'd Like to Hear from You

IBM Content Manager CommonStore for Lotus Domino
IBM Content Manager CommonStore for Lotus Domino
Administrator's and Programmer's Guide
Version 7 Release 1

Publication No. SC33-7053-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SC33-7053-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Fold and Tape **Please do not staple** Fold and Tape

SC33-7053-00

Cut or Fold
Along Line

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-7053-00

Spine information:

IBM Content Manager
CommonStore for Lotus
Domino

IBM Content Manager CommonStore for Lotus Domino Administrator's and Programmer's Guide

Version 7
Release 1