IBM Content Manager

CommonStore for Lotus Domino Administrator's and Programmer's Guide

Version 8.1

SH12-6742-00

IBM Content Manager

CommonStore for Lotus Domino Administrator's and Programmer's Guide

Version 8.1

SH12-6742-00

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix I, "Notices" on page 263.

First Edition, September 2002

This edition applies to Version 8.1 of IBM Content Manager CommonStore for Lotus Domino, program number 5724-B86, and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this book

This book provides detailed information on the following subjects:

- Installing and configuring CommonStore for Lotus® Domino™
- Administering CommonStore for Lotus Domino
- Customizing archiving processes
- Application programming for CommonStore

Reading this book enables you to set up, control, and customize CommonStore for Lotus Domino. It is divided into the following parts:

- Part 1, "Overview" on page 1. This part provides an introduction to CommonStore for Lotus Domino.
- Part 2, "Concepts of CommonStore for Lotus Domino" on page 9. This part describes the concepts of CommonStore for Lotus Domino.
- Part 3, "CSLD features" on page 39. This part gives you a detailed description of the product features.
- Part 4, "Installing CommonStore for Lotus Domino" on page 69. This part deals with the installation of CommonStore for Lotus Domino.
- Part 5, "The CommonStore Server" on page 85. This part covers the administration of the CommonStore Server.
- Part 6, "Configuring the archives" on page 105. This part describes the customization of the supported archive systems in order to make them work with CommonStore for Lotus Domino.
- Part 7, "CSLD administration" on page 133. This part explains how to administer the task component of CommonStore for Lotus Domino. During the installation, the task component is added to the Lotus Domino server. It interacts with the archiving engine on the CommonStore Server.
- Part 8, "CSLD programming guide" on page 163. This part explains the programming features of CommonStore for Lotus Domino. It mainly deals with job documents and Lotus Script programming tools. This part is of interest to programmers who want to customize existing Lotus Notes® applications for use with CommonStore for Lotus Domino.

Who should read this book

This book is intended for administrators and application programmers. These should have in-depth knowledge of Lotus Domino, Lotus Notes, Lotus Notes application development, and the archive system that they intend to use.

This book is not primarily intended for end users. However, it contains some conceptual information that is relevant for users involved in archiving procedures.

What's new in this version

CommonStore for Lotus Domino Version 8.1 offers you the following new features:

1. Policy-driven archiving. See "Archiving policies" on page 27.
2. Policy-driven deletion. See "Deletion policies" on page 28.
3. Administrator-triggered retrieval. See "Administrator-triggered retrieval" on page 28.

4. Support for Content Manager for iSeries. See Chapter 28, “Configuring Content Manager for iSeries” on page 119.
5. Support for Content Manager views or subsets (archive access restriction). See Chapter 29, “Using Content Manager views (subsets)” on page 123.
6. Content Manager Version 8 support.

Conventions and terminology used in this book

This section describes the abbreviations, acronyms, and highlighting conventions used in this book.

Product names

To facilitate reading, the product name CommonStore for Lotus Domino is most of the times shortened to CommonStore. Content Manager OnDemand is often abbreviated to CMOD. Likewise, the names ADSM or TSM refer to ADSTAR Distributed Storage Manager and Tivoli® Storage Manager.

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.

Since SAP Release 3.1, the names of some program components and products have changed. The change is hinted at on the first occurrence of such a name.

Highlighting conventions

Highlighting is necessary to set product-related terms, product elements, and code examples off from the text flow. This book uses the following highlighting conventions

Throughout this book, *italics* are used for
- Book titles
- Emphasis
- Options / variables / parameters / keywords

Boldface is used for the following elements:
- 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

`Monospace` is used for
- Coding examples
- Entered data
- Group and user IDs
- Message text
- Transaction codes (T-codes)

Underlined bold indicates default values.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other CommonStore documentation:

- Visit our home page at http://www.ibm.com/software/data/commonstore/. There, you find the feedback page where you can enter comments and send them.
- Send your comments by e-mail to swsdid@de.ibm.com. Be sure to include the name of the book, the part number of the book, the version of CommonStore, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).
- Fill in one of the forms at the back of this book and return it by mail, by fax, or by giving it to an IBM representative. The mailing address is on the back of the Readers' Comments form. The fax number is +49-(0)7031-16-4892.

List of Abbreviations

The abbreviations used in this document are listed in Table 1.

Table 1. Abbreviations used in this book

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
AS/400	Application System/400®
COLD	Computer Output on Laser Disk
DLL	Dynamic Link Library (files with the extension `.dll`)
ECL	Edit Control List
GUI	Graphical User Interface
ITS	Internet Transaction Server
NFS	Network File System
NT	New Technology (Microsoft operating system Windows NT®)
OCR	Optical Character Recognition
OLE	Object Linking and Embedding
OS/2®	Operating System/2®
OS/390®	Operating System/390
OS/400®	Operating System/400®
OTF	Output Text Format (files with the extension `.otf`)
PDF	Portable Document Format (files with the extension `.pdf`)
S/390®	System/390®
TCP/IP	Transmission Control Protocol/Internet Protocol
TIFF	Tagged Image File Format (files with the extension `.tif`)
UNID	Universal Notes Identifier
UNIX®	An operating system developed at Bell Laboratories
URL	Uniform Resource Locator

Part 1. Overview

Chapter 1. What is CommonStore for Lotus Domino?

This chapter gives a rough overview of the components in a CommonStore for Lotus Domino system. The components shown in Figure 1 are briefly described in the following sections.

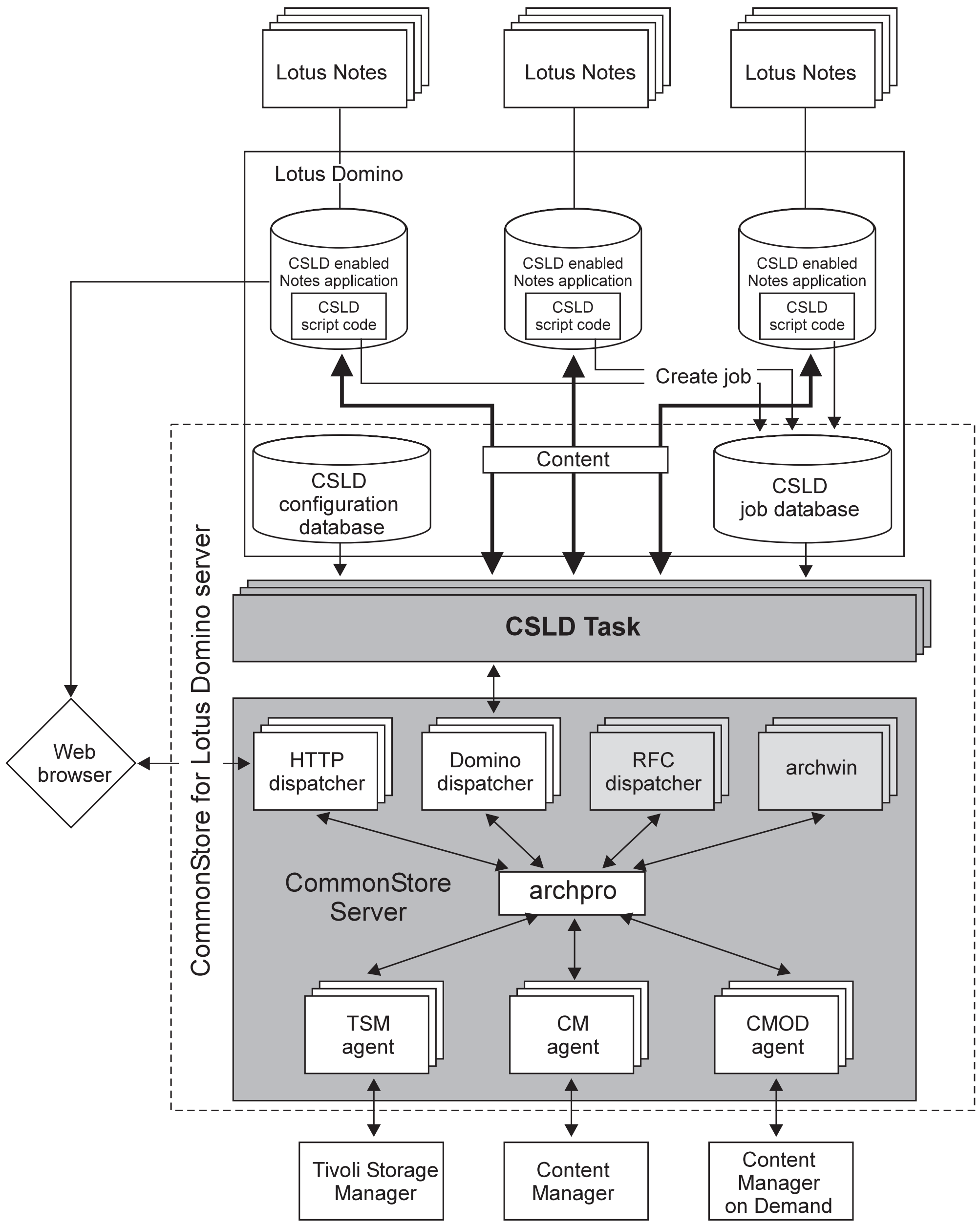

Figure 1. Components in a CommonStore for Lotus Domino system

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

The archpro program

The archpro program 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 program. For example, whenever content is to be archived, an archiving request is sent to the archpro program, along with the content and the content's descriptive information.

The archpro program 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 program is responsible for handling Domino-related content, it has nothing to do with Notes/Domino, that is, it does not call any Notes/Domino related code.

The archpro program does not handle the communication with the archives itself, that is, the archpro program 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 program. When all agents are busy, the request is held in the queue of the archpro program 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 program and are automatically started by the archpro program.

Archives

CommonStore for Lotus Domino works with the following archive systems, which are accessed by the agents:

- Tivoli Storage Manager
- Content Manager
- Content Manager for iSeries
- Content Manager OnDemand

Domino dispatcher

The Domino dispatcher is the interface between the application-independent archpro program and the CSLD Task, the Domino dependent part of the system. It translates Domino requests into the language of the archpro program. 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 program 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 program, and is automatically started by the archpro program.

HTTP dispatcher

The HTTP dispatcher implements a set of http commands which map the main functions of the archpro program 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. Hit lists and result documents resulting from a query also contain such a URL link.

The HTTP dispatcher runs on the same machine as the archpro program and is automatically started by the archpro program.

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 program. In CSLD, there are archiving tasks and retrieval tasks. Archiving tasks process requests that add or modify content in the archive (for example, archiving, updating, deleting). Retrieval task process requests that retrieve content from the archive (for example, 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 program, 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 run time. 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 program. 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 Lotus Script 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.

Crawler

The crawler comes into play when you want to employ the automation features of CommonStore for Lotus Domino. This independent program is the central application for the following functions:

- Policy-driven archiving
- Policy-driven deletion
- Administrator-triggered retrieval

When used for policy-driven archiving or policy-driven retrieval, the crawler checks whether documents in the specified Lotus Notes databases meet a set of selection criteria, which is called a policy. If documents do meet those criteria, the crawler creates archiving or deletion jobs for them.

When used for administrator-triggered retrieval, the crawler creates retrieval jobs for all documents that were previously archived from a specified set of databases.

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 2. Features and scenarios

The following list contains the features of CommonStore for Lotus Domino. It also gives you an impression of the range of tasks that it can handle:

- Archive or offload any kind of Lotus Notes documents.
- Archive or offload attachments, optionally detach attachments after archiving.
- Archive documents in binary or native Lotus Notes format. Allows to restore content in Lotus Notes document format. This way, you can process the documents as before.
- Archive "images" of Lotus Notes documents in Microsoft RTF or ASCII format for long-term archiving.
- Optionally delete archived documents.
- Store descriptive fields in archive index fields for search purposes.
- Archive Lotus Notes views and folder structures (folders with all their subfolders).
- Perform searches over the archive.
- Archive interactively or scheduled.
- Retrieve search results as one hit list document or multiple result documents, each representing one hit.
- User interface is 100% Lotus Notes. You need not install client code on the users' workstations.
- Interface from user databases to CSLD is fully Notes based. You can send all requests to CSLD as regular Notes documents.
- Configuration managed from Lotus Notes through a configuration database.
- Define target archives for Lotus Notes documents based on document types (forms) or specific document field values.
- "Private data": Protect your archived documents from other users. Very helpful for mail archiving.
- Protect retrieved data from being seen by other users through readers' fields.
- User security exit: Map Lotus Notes user IDs to archive user IDs.
- Ready-to-use Lotus Script libraries make CSLD functionality available in your application.
- Mail archiving sample application containing example code, actions, and other design elements for standard archiving tasks.
- Heavy multithreading: Archive or retrieve from multiple databases concurrently for maximum throughput.
- Archive from different Lotus Notes platforms with just one CSLD server.
- Intelligent scheduling of CSLD request processing.
- Use CSLD as your Lotus Notes front-end to access documents that were archived by other applications, for example SAP or scanning applications.
- Content Manager folder browsing: The contents of Content Manager folders can be displayed in a hit list.
- Browser viewing: Documents archived with CSLD can be displayed in a browser without prior retrieval.
- Tasks can be configured to process jobs of entire Domino servers and database directories.

- Archiving and restoring complete folder structures: Entire Lotus Notes folders including subfolders can be archived and completely restored while preserving the folder structure.
- Restoring of document UNID: When you retrieve documents that were archived in native format, the original document UNID is preserved.
- Customizable agents for the automatic processing of archived, updated, retrieved, or deleted documents.
- Content Manager workbasket support and support for virtual workbaskets and virtual folders in Content Manager OnDemand.
- Document re-archiving: You can archive the attachments of a document, attach new ones, and then add the new ones to the archived document by another archiving process.
- CSLD Tasks can be started without typing in passwords.
- All documents except for signed documents can be restored.
- User exits for rasterizing allow you to convert documents and their attachments to various formats like TIFF.
- Default content type: Attachments with unknown file extensions can be mapped to a default content type.
- Documents can be made response documents to the hit lists from which they were retrieved or to the queries that were used to find them.
- Simplified configuration for Tivoli Storage Manager archives.
- CSLD can be configured to send users an e-mail when their jobs fail.

Part 2. Concepts of CommonStore for Lotus Domino

Chapter 3. User IDs and roles

CSLD defines the following IDs and roles:

CSLD administrator
The Lotus 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 program (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 (Content Manager), application groups (Content Manager OnDemand) and management classes (Tivoli Storage Manager) for CSLD.
5. Managing the archint.ini file which contains archive-related definitions.

The CSLD user ID
The Notes specific part of the CommonStore Server is called CSLD Task. It is a regular Lotus Notes 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 e-mail archiving, this means that the CSLD user ID can read the e-mail 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. That is, 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 Lotus Notes production database for CSLD use. A CSLD application developer typically adds design elements like views (search results for example), actions (for example, 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 4. Using 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 (for example, archiving a certain view every day at midnight) or manual (for example, by selecting a number of documents in a view and starting an agent, for example, in the Sample mail application via the path **Actions → CSLD → archive selected documents**);
- be triggered interactively via actions (for example, in the Plug ’n’ Go Mail Demo by pressing the **Archive** button in a document’s action bar);
- be triggered automatically via Notes events (for example, **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 5. Configuration database

To get CommonStore for Lotus Domino to work in your environment, you must define mappings in the configuration database. This chapter introduces the various mapping concepts.

Mapping Lotus Notes forms to archives

By mapping Lotus Notes forms to archives, you determine the target archive of documents that use a certain form. You define document mappings for that purpose, and special mappings to handle exceptions.

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 following data types:

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

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

At the same time, all documents in a Content Manager or Content Manager 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 in 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 *NewMemo* 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 **<form name>**, the document mapping for **<form name>** 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 **<form name>** but there is no document mapping for **<form name>** defined, CSLD checks whether or not **<form name>** is an alias defined in a document mapping for form **<other form>**. In this case, the document mapping for **<other form>** 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. 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. Or, at a record company, the administrator might want to store classical, rock and jazz 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 (for example, 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, and so on.

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.

Mapping document fields to archive attributes

The next step in the mapping process is to map document fields to archive attributes. By defining such a mapping, you select certain fields of a Lotus Notes to serve as index information. The fields become part of the archived document. During a query or retrieval operation, the content of these fields is searched to identify documents in the archive.

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. See the following list of supported data types:

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.

Content types and original file names

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 format of the document, 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. To determine the content type of a document, CSLD uses the content-type mapping table in the configuration database. This table maps file extensions to content types. The archives always store the content-type information that they find in this table.

If your archive system is Content Manager, it is recommended that you manually map *all* possible file extensions to the proper content types. You must at least define a default content type, otherwise CommonStore does not archive documents to Content Manager. See “How CommonStore determines content types” on page 22 for more information. For Content Manager OnDemand or Tivoli Storage Manager, this is not required.

You must manually define content-type mappings in the following cases:

- If you want to view archived content in the Content Manager viewer.
- If you want to view archived content in a Web browser. CommonStore uses content-type information to determine the correct MIME type. You must map the content types to the appropriate MIME types in a file called csmimes.properties. See Chapter 20, “Browser viewing of archived content” on page 67 for more information.

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.

Tivoli Storage Manager
CSLD stores document content types internally. The information is invisible to the user.

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).

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

Content Manager for iSeries
Like Content Manager, Content Manager for iSeries 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 index class that CSLD archives documents to. This allows CSLD to reconstruct the filename of archived attachments when these are restored to Lotus Notes.

Content Manager for iSeries stores the content type provided by CSLD natively with the document. That is, no attributes must be defined to store the content type.

Content Manager OnDemand
Like Content Manager, CMOD 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 that CSLD archives documents to. This allows CSLD to reconstruct the filename of archived attachments when these are restored Lotus 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.

How CommonStore 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 it finds an entry, the document is stored under the configured content type.
4. If it does not find an entry, CSLD checks whether the content type mapping table contains an entry for the ″default file extension″ .unk (unknown).
5. If it finds an entry for the default extension, the content type given here is used.
6. If it cannot even find an entry for the default extension, CSLD reacts in one of the following ways, depending on your archive system:
   - If you use a Content Manager archive, CommonStore does not archive the document and returns an error message indicating that it could not determine the content type. If the notification feature is enabled, an e-mail is sent, asking the CSLD administrator to create a mapping for the file extension.
   - If you use a Tivoli Storage Manager or Content Manager OnDemand archive, CommonStore internally defines the file extension as the content type.

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 cannot 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 6. 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 Task 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. The CSLD user's notes.ini file is passed 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 (for example, 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 Lotus 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 139 for more details.

Chapter 7. Automation features

The term automation features summarizes policy-related functions and administrator-triggered retrieval. These features employ the crawler program to automate archiving, deletion, and retrieval processes. The crawler is a separate program with the name csc.exe.

When you create policies, you define sets of selection criteria that the crawler program uses to identify documents. It then creates archiving or deletion jobs for these documents.

Administrator-triggered retrieval facilitates bulk retrieval: You select a number of databases and the crawler creates retrieval jobs for all the documents previously archived from these databases.

Policies

Policies contain the parameters for policy-driven processes. To define policies, you use a form in the configuration database, which lets you enter these parameters.

Policies are carried out by the crawler, which is an independent program. The crawler scans the specified databases on the Lotus Domino server, and checks whether the criteria defined in the policies apply to the documents in these databases. When a document fulfills the criteria, the crawler creates an archiving or deletion job for it.

CommonStore for Lotus Domino offers the following types of policies:
- Archiving policies
- Deletion policies

Archiving policies

When you create an archiving policy, you specify a schedule for policy-driven archiving and, optionally, a number of archiving rules. Each time policy-driven archiving starts, it is checked whether the rules apply to the documents in the assigned databases. For documents that meet the criteria stated in the rules, the specified actions are carried out.

You can base archiving rules on document sizes and dates. If you specify rules on the basis of document sizes, you define the minimum size of the documents you want to archive in KB.

If you specify rules on the basis of dates, you define limits with respect to the age or creation time of the documents to be archived. For example, you can specify that all messages older than 5 days or all messages received after July 4 are archived.

You can also combine the determining factors of different rule types to formulate a compound rule. For example, you can say that you want to archive all documents larger than 50 KB and older than 5 days.

Deletion policies

Deletion policies allow you to eliminate great numbers of documents-no-longer-needed from your archives. As it is very uncertain what kind of documents these might be and since the characteristics might vary, this task cannot be automated by a predefined set of rules. Therefore, you define selection criteria by creating a Lotus Notes selection formula each time you want to delete documents. The crawler then uses these selection criteria to identify the documents to delete.

Administrator-triggered retrieval

This feature allows you to restore great numbers of documents in one go. You specify one or more Lotus Notes databases, and the crawler retrieves all the documents that were previously archived from these databases. This saves considerable time when restoring large numbers of documents because you need not select the documents before you can start the process.

You put this function into action by starting the crawler program with an additional parameter. Note that it takes the work schedule of the affected CSLD Task into account, but disregards any policies that you might have defined for the selected databases.

Chapter 8. Performance

The overall performance of CSLD depends heavily on 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 servers 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 CommonStore agents
See also Chapter 9, “Scalability” on page 31.

Performance of the archive server
See also Chapter 9, “Scalability” on page 31.

Chapter 9. Scalability

CSLD supports various methods of scaling, which are explained in the subsequent sections:

1. Using more than one job database
2. Using more than one CommonStore agent/dispatcher
3. Scaling on the CommonStore server level

Using more job databases

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.

Using more agents and dispatchers

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

The Domino dispatcher
The Domino dispatcher is a component which takes requests and routes them to the CommonStore server. The number of dispatchers started is determined by the parameter DOMINODPS in the CommonStore server configuration profile (usually 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 CommonStore 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 CommonStore server might occur, resulting in erroneous jobs.

The CommonStore agent
The CommonStore agent is the entity responsible for communicating with the archive. For every supported archive (that is, Content Manager, CMOD and Tivoli Storage Manager, there is a special type of agent. CommonStore 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 CommonStore 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 CommonStore Server

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

Chapter 10. Security

CommonStore for Lotus Domino offers a range of security features on various levels, allowing you to tailor the system security to your needs.

Security on the database level

There are several security features built into the most important Lotus Notes databases in a CommonStore for Lotus Domino setup. The security features in these databases are discussed in the following sections.

Job database

Normally, all users have *Author* access rights to the CSLD job database, so they can delete or modify their own job documents only. If the CSLD administrator wants to deny certain users the use of CSLD services, he merely needs to remove these users from the access control list (ACL) of the job database. 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 to be the requester of the job. 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]. See Chapter 4, "Using CSLD services" on page 15 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.

Accessing the archives

In CSLD, all operations on an archive are performed via CommonStore agents. Such agents are archive clients, and must therefore be started with a certain user ID. At the same time, CSLD users log on to CSLD user databases using their Lotus 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 to log on to 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 CommonStore 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 user ID. This user ID 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 CommonStore (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, you must design th e Lotus Notes database in a way that prevents illegal access to the documents in the archive.

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 through 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 on to 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 CommonStore agent whenever. data is accessed in the archive. You can switch on the security user exit in the CommonStore 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 CommonStore must log off and log in again when user IDs are switched. However, CommonStore always tries to find an agent that is currently logged in under the user ID making the request.

Restricting access to archived documents per user and database

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 the following security modes, which are described in the subsequent sections:

- *Retrieve documents by original user only*
- *Retrieve documents to original database only*

These two modes are described in the following sections.

Retrieve documents by original user

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**.

Retrieve documents to original database only

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

Read the following list carefully if you plan to use the security modes. It contains a few points to consider and a number of valuable hints and tips:

- *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 139 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 Lotus Script, the CSLD job

format, and other users' environments can do is 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*.
- In Content Manager for iSeries, the attribute length is limited to 40 characters. Therefore, you cannot create user names in canonical format that are longer than 40 characters.

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.

Important

Turn on security modes at archiving time to make sure that the required attributes are available at retrieval time. The security attributes must be created at archiving time. You cannot activate them later.

Part 3. CSLD features

Chapter 11. Archiving methods

CSLD supports the following archiving methods:

- Attachment archiving
- Archiving in Lotus Notes native format
- Long-term archiving by *rasterizing* documents

These 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 20, “Browser viewing of archived content” on page 67. This message is removed when attachments are restored back to a container document.

Important

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 Lotus Notes Release 3 and higher reside in an RTF field. CSLD also supports attachments created with a Lotus Notes version prior to release 3 that reside in a Lotus Notes item rather than in an RTF field. In the Notes client, such attachments are usually displayed at the bottom of a document. These attachments are not replaced with a URL.

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 Lotus 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 native Lotus Notes 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.*

Important

- 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.
- You can archive encrypted documents, but encryption imposes some limitations because CSLD is running under a regular Lotus Notes user ID. Encryption prevents unauthorized users from altering encrypted parts of a document. Only users in possession of a decryption key are allowed to do this. Often, it is practically impossible to maintain all the decryption keys that the CSLD user ID might need because many different users archive documents using CSLD. The following features do not work for encrypted documents unless the CSLD user ID is equipped with the required decryption key:
  - Attachment archiving or archiving in rasterized format.
  - Converting the source documents into placeholders (stubbing) after content archiving (invoked by the job paramenter *createDocumentStub*)
  - Extracting index information from the encrypted documents. This depends on the level of encryption. If only rich-text fields are encrypted, CSLD can extract index information. If all the fields are encrypted, it cannot.

Long-term archiving by rasterizing documents

When thinking of long-term archiving, that is, 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, for example, 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, that is, 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 (TIFF for example) 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 rasterizing 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 rasterizing 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 36, "The raster-exit DLL" on page 159.

Chapter 12. 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, that is, 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 necessarily archive documents only once. It depends on the logic of a CSLD-enabled database whether a user can archive a document twice or more. After completing an archiving request, CSLD stores the request type (archiving in native format, attachment archiving, or archiving of rasterized documents) with the archived documents. In addition, it writes a status value to the **CSNDArchiveID** item or a specially defined field. You can define the status field values in the configuration database.

If the status field of a document does not contain the value you defined as the error value, you can consider the document to be archived. If a status field does not exist, you can likewise consider a document to be archived when the **CSNDArchiveID** item contains the document archive ID rather than the value *Error*. If the **CSNDArchiveID** item does not exist, the document was either not archived or deleted from the archive.

For long term archiving, that is, 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 more information on archiving states, see “Defining profile documents” on page 139.

Controlling document rearchiving

Normally, CommonStore for Lotus Domino does not check the existence or the value of the **CSNDArchiveID** item inside a Lotus Notes document. As this is the item containing the CSLD document IDs, this means that CSLD does not check if a document has been archived already. Hence, you can archive the same document over and over again. CSLD handles documents that you already archived according to the request type and the job parameters.

Rearchiving attachments or rasterized documents

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″, for example, 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, that is, 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 does not restore a certain version of a Lotus Notes document, nor does it keep information about different versions of a document. Through the **CSNDArchiveID** item, the Notes document always links to all the attachments in the archive that have ever been archived from that document.

Attention

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

Rearchiving documents in native Lotus Notes format

When you archive documents in native Lotus Notes format or rasterized format more than once, the document content is not appended to the documents that you already archived. Instead, a new version is placed in the archive, and the content of the **CSNDArchiveID** item in the Lotus Notes document is replaced with the value of this new version. Thus, you can only retrieve the versions of the most recent archiving run directly. To retrieve other, older versions, you must employ other tools with search and retrieval capabilities, like, for example, the Content Manager Client.

Example:

Suppose you have archived a number of documents in native Lotus Notes format. Each archived document leaves a placeholder document or stub in the Lotus Notes database. When you rearchive these documents, you actually archive the content of the placeholder documents. The previously archived documents, which include the original content, are still in the archive, but you can no longer retrieve them directly because the values of the **CSNDArchiveID** item have changed. They were replaced with the values generated during the second archiving process. Thus, the IDs in the **CSNDArchiveID** field give you access to the archived placeholders rather than to the original content.

To prevent unwanted results, that is, forfeit the chance to retrieve archived documents directly from Lotus Notes by using the IDs in the **CSNDArchiveID** field, you must carefully design your application logic.

Checking the archive integrity

Using the *checkArchiveIntegrity* parameter, you can change the usual behavior of rearchiving processes, which gives you more control over these processes.

When you set the *checkArchiveIntegrity* parameter, CSLD checks if the IDs in the **CSNDArchiveID** items of your Lotus Notes documents already exist in the archive. If matching IDs are found, CommonStore for Lotus Domino performs an additional checks before it starts the rearchiving process. The outcome of these checks determines if or how the documents are rearchived. The logic that CSLD applies depends on the request type (attachment archiving, archiving of rasterized documents, or archiving of documents in native Lotus Notes format) of the documents that were already archived. See the following sections for a description of the check logic.

Example:

A Lotus Notes application archives the document attachments in a database after 20 days to free up space quickly. Every six months, another application archives the documents themselves in native format to remove all the information that is no longer needed. This rearchiving process obscures the document IDs of the archived attachments because the content of the **CSNDArchiveID** items in the archive is overwritten with new values.

If you decided to remove the attachments from their original documents after successfully archiving them, you might have a problem accessing the attachments in the archive. After the rearchiving process, you can only access and retrieve the native documents directly, from which the attachments were previously removed.

The *checkArchiveIntegrity* parameter can prevent this loss of direct access because its use in the given context would have stored the IDs of the attachments in a new item called **CSNDArchiveIDAttach**. In addressing this item instead of **CSNDArchiveID**, you can still access the attachments directly.

Important

- If you use the *checkArchiveIntegrity* parameter, you can no longer archive documents as described in "Rearchiving attachments or rasterized documents" on page 46 and "Rearchiving documents in native Lotus Notes format" on page 47.
- CSLD does not check if attachments are newer than the ones in the archive, that is, if an update is required or not.

Procedure for the rearchiving of attachments and rasterized documents

If you set the *checkArchiveIntegrity* parameter, and want to rearchive documents whose request type in the archive is *attachment archiving* or *archiving of rasterized documents*, CSLD applies the following logic:

- If the current request type is *native archiving*, CSLD performs the following steps:
  1. CSLD transfers the existing CSLD archive ID from the **CSNDArchiveID** item to a new item called **CSNDArchiveIDAttach**.
  2. It archives the Lotus Notes document in native format.
  3. It stores the new document ID in the **CSNDArchiveID** item.
  4. It changes the existing request type to *native archiving*.

  This way, the previous document IDs are preserved rather than overwritten.
- If the current request type is again *attachment archiving* or *archiving of rasterized documents*, CSLD assumes that the document has already been archived and does not archive it again. It merely performs post-processing operations, which might consist of the following steps:
  - Removing the attachments from the Lotus Notes documents after the archiving process has finished
  - Starting a post-processing agent

Procedure for the rearchiving of documents in native Lotus Notes format

If you set the *checkArchiveIntegrity* parameter, and want to rearchive documents whose request type in the archive is *native archiving*, CSLD applies the following logic:

- If the current request type is *attachment archiving* or *archiving of rasterized documents*, CSLD checks if the item **CSNDArchiveIDAttach** exists. Depending on the outcome of this check, CSLD performs one of the following steps:
  - If **CSNDArchiveIDAttach** exists, CSLD assumes that the documents have already been archived, and does not archive them again. It merely performs post-processing operations, which might consist of the following steps:
    - Removing the attachments from the Lotus Notes documents after the archiving process has finished
    - Starting a post-processing agent
  - If **CSNDArchiveIDAttach** does not exist, CSLD rejects the archiving request and returns an error message saying that it cannot fulfill the request.
- If the current request type is again *native archiving*, CSLD assumes that the documents have already been archived and does not archive them again. It merely performs post-processing operations, for example, starting a post-processing agent.

Chapter 13. Retrieval methods

To choose a suitable retrieval method, it is important to understand the archiving behavior of CSLD: When the original documents are not deleted after successful archiving operations, the document IDs of archived attachments or document bodies are stored in the **CSNDArchiveID** or **CSNDArchiveIDAttach** items of the original document. When you archive a native or rasterized document, only one document is created in the archive. Thus, the **CSNDArchiveID** item contains only a single entry.

If you archive attachments, one document is created in the archive for each attachment. Consequently, the **CSNDArchiveID** item contains as many entries as there are attachments. When you rearchive attachments, the document IDs in the **CSNDArchiveID** item accumulate because the IDs generated during the rearchiving process are added to the existing ones.

You can suppress unwanted rearchiving processes and thus avoid confusion through an abundance of document archive IDs by using the *checkArchiveIntegrity* parameter. For details see "Controlling document rearchiving" on page 46.

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 (that is, taken from the CSNDArchiveID field of an archived Notes document or from a hit list), 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 hit list 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 hit list will return the folder content in a second hit list (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, that is, the document the native content was created from, still resides in the database, users can choose if they want to replace the original with the retrieved content or store the retrieved content as a new document.If the user specifies a target document, the retrieved native document will replace this document, that is, upon successful restoration, it will have that document's UNID.
- *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 175 for a description of result documents.

In addition, you can turn retrieved documents into response documents or save them in a target folder of your choice. These options are independent of the retrieval method that you choose.

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 17 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 hit list 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 hit-list document

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

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

It is up to the application logic to decide what to do with hit-list 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, for example, 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 hit list is generated. Instead, all hit documents are directly retrieved from the archive and written to a target location. Otherwise, a hit list is created.

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

Every hit-list document or retrieved document contains two fields:
- the name of the person who issued the query (that is, 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 hit-list document has an item named **CSNDArchiveID** containing the archive IDs for all hits in the list.

Chapter 14. Agents for processing archived documents

CSLD supports the automatic invocation of custom Notes agents for processing archived documents. These agents can be Lotus Script, 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.
- **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.
- **Post-update agents** become active after you updated a document in the archive. You can employ these agents, for example, to trigger additional processes or to set the value of state fields.
- **Post-deletion agents** start working after you deleted a document from the archive, but only if the shell document remains in the database. You can use these agents to remove the values from custom state fields that were filled during the archiving process.

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 Lotus Script, you would access the document via the *DocumentContext* property of class *NotesSession*.

This behavior does not change when CSLD processes documents that contain more than one attachment. The custom agents are invoked only once, for the document containing the attachments.

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 e-mail (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.

Chapter 15. 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 "Mapping Lotus Notes forms to archives" on page 17 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 Lotus Notes native format. The reason is simple: Index field 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.
- 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 16. 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 hit-list document, the **CSNDArchiveID** item contains the archive IDs of all hits in the list.

Consistency of Lotus 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 application developer's responsibility to ensure consistency between the archive and CSLD.

In addition to a removal request for the CSNDArchiveID item, a deletion job can contain the request to remove the Lotus Notes document together with the archived document.

Chapter 17. 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 acknowledgement 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 hit list. As a Content Manager workbasket can contain documents from different index classes, the documents in the hit list 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 18. Archiving of Notes folder structures

In Lotus Notes documents can be organized in folders, for example, 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, refer to “Preparing Content Manager OnDemand for folder archiving” on page 130 or “Preparing Content Manager for folder archiving” on page 113.

Restoring archived Notes folder structures

Unlike folders in a Content Manager archive, Notes folders are only identified by a name, that is, 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, and so on, 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 rasterized 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 (that is, 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 (that is, 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 19. 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, and so on. 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, and so on). All documents concerning that employee would then virtually be filed into that folder. Thus each employee folder would contain references, for example, 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.

Notes

- 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.
- 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.

Hit-list representation

A hit list 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 hit list, 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 20. 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 hit-list 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 CommonStore 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, CommonStore 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. Installing CommonStore for Lotus Domino

Chapter 21. Installing the CommonStore Server

This chapter describes how to install the CommonStore Server component. The server contains, among other modules, the archiving engine, that is, the archpro program. This program performs the true archiving jobs, that is, it stores or retrieves messages and attachments from an archive.

Server prerequisites

You must prepare the environment before you can install the CommonStore Server. Therefore, make sure that all of the software listed under “Operating system and additional software” and at least one of the packages listed under “Archive client software” and “Archive server software” on page 72 are already installed.

Operating system and additional software

The operating system and additional software listed in Table 2 must be installed before you can install the CommonStore Server.

Table 2. Operating system and software requirements

Platform	Installation prerequisites
AIX	AIX 4.3.3 (or later) Minimum hardware: 256 MB system memory
Windows® 2000	Windows 2000 including Service Pack 2 (or later) Minimum hardware: CPU speed 1000 MHz, 512 MB system memory, 40 GB hard-drive capacity
Lotus Domino	Lotus Domino R5 including Lotus Notes R5 clients
All	Java™ Runtime Environment (JRE) or Java Development Kit (JDK) version 1.3 or later. You can download it at http://www.ibm.com/java/jdk/download Add the path to the JRE or the JDK to the system path statement by setting the PATH environment variable accordingly.

Attention

Before installing Service Pack 2 for Windows 2000, check Microsoft information sources for possible problems with existing applications. Carefully decide whether you want to install the service pack if there are known problems.

Archive client software

Table 3 on page 72 shows the required release level of the supported archive client software. Before installing CommonStore, the listed version or a later release of your client software must be installed on the workstations connected with the archive server.

Table 3. Prerequisites for the client software of your archive system

Archive	Prerequisites
Tivoli Storage Manager	Tivoli Storage Manager client (API) version 4.2 The Tivoli Storage Manager clients can be downloaded from the following server: http://www.tivoli.com
Content Manager	Content Manager client 7.1 including fixpack 1 (7.1.0.1)
Content Manager Version 8	Enterprise Information Portal (EIP) Version 8.1 with local connector for Content Manager Version 8.1
Content Manager OnDemand	Content Manager OnDemand for Multiplatforms 7.1.0.5 (same as server package; this package includes the APIs)

The CommonStore Server supports a wide variety of hardware platforms from which customers are free to choose based on the requirements of the given application scenario. While CommonStore itself runs on AIX, Windows NT, or Windows 2000, 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 a zSeries™ (S/390) mainframe computer. Through the flexible hardware support offered by Tivoli Storage Manager, CommonStore also allows the storage of documents on quite arbitrary storage media, including hard disks, tape drives, optical media, or corresponding library systems.

Archive server software

Table 4 shows the required release level of the supported archive server software. Before installing CommonStore, the listed version or a later release of your server software must be installed on the archive server.

Table 4. Prerequisites for the server software of your archive system

Archive	Archive server platform	Prerequisites on archive server
Tivoli Storage Manager	AIX, HP-UX, Sun Solaris, Windows NT, z/OS™ (OS/390)	Installed, tested, and operational Tivoli Storage Manager 3.7 Tivoli Storage Manager is configured with defined management classes and nodes — which should be used for archiving (see "Basic setup" on page 87).
Content Manager	AIX, Windows, z/OS, OS/390	Completely installed, tested, and operational Content Manager 7.1 including fixpack 1 (7.1.0.1) Configured with defined index classes, workbaskets, and Content Manager users — which should be used for archiving (see "Basic setup" on page 87).
Content Manager Version 8	AIX, Windows, Sun Solaris	Content Manager Version 8.1
Content Manager for iSeries	iSeries™, OS/400	Content Manager for iSeries Version 5.1 (only with CommonStore Server on Windows 2000)

Table 4. Prerequisites for the server software of your archive system (continued)

Archive	Archive server platform	Prerequisites on archive server
Content Manager OnDemand for Multiplatforms	AIX, HP-UX, Sun Solaris, Windows	Completely installed, tested, and operational Content Manager OnDemand for Multiplatforms 7.1 including fixpack 5 (7.1.0.5) Configured for CommonStore as described in "Basic setup" on page 87.
Content Manager OnDemand for iSeries	iSeries, OS/400	Content Manager OnDemand for iSeries Version 5.1
Content Manager OnDemand for z/OS and OS/390	z/OS, OS/390	Content Manager OnDemand for z/OS and OS/390 Version 7.1

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

Installation steps

This section describes how to install the CommonStore Server package on an AIX or a Windows system. For a basic configuration of CommonStore, see "Basic setup" on page 87.

Installing the CommonStore Server package on AIX

To install the CommonStore Server on AIX, perform the following steps:

- Install the CommonStore software package
- For every instance of the CommonStore Server, proceed as follows:
  - Create a CommonStore instance user.
  - Log on as the CommonStore instance user.
  - Modify the notesenv.sh file to set your Lotus Notes run-time environment.
  - Modify the *.profile* file of the CommonStore instance user.

Providing access to Content Manager archives

If your archive system is Content Manager, you must add the details for connecting to the library server to your Content Manager network table. If you have not configured the network table during the installation of Content Manager, you can do it now by following these steps:

1. Log in as user root.
2. Enter `frnnetcfg`.
3. Enter the details of your library server in the dialog box that opens.

When you close the dialog, the frnolint.tbl file is updated, which contains the Content Manager network table.

Installing Lotus Domino

Integrating the Lotus Domino server is uncomplicated because CommonStore only needs the Lotus Notes run-time environment to connect. You do not even have to start or configure the Domino server. To install Lotus Domino R5 for use with CommonStore, follow these steps:

1. Log in as user root.
2. Add a volume group to the computer hosting the Domino server.
3. Add the following logical volumes to the new volume group:
   - dom50lv
   - notes50lv
4. Add large file-enabled journal file systems (JFSs) for the following purposes:
   - One JFS for the binary Lotus Domino code in the logical volume dom50lv, for example:
     ```
     /usr/dom50
     ```
   - One JFS for the Notes data directory in the logical volume notes50lv, for example:
     ```
     /notes50
     ```
5. Mount the new file systems.
6. Create a group called notes.
7. Create a user *notes* and make this user a member of the notes group.
8. Start the Lotus Domino installation routine, for example by entering the following commands:
   a. `cd /cdrom/ibmpow`
   b. `./install`

   Before the installation process starts, you must enter the following information:
   a. Server type: `Mail Server`
   b. Program directory: `/usr/dom50`
   c. More than one Domino server on this computer: `No`
   d. Data directory: `/notes50`
   e. UNIX user: `notes`

After the installation, you find the notes.ini file of the Lotus Domino server in the /notes50 subdirectory.

Installing the software package

To install the CommonStore Server on AIX, follow these steps:

1. Log in as user root.
2. Call SMIT.
3. Install and Update Software.
4. Install/Update Selectable Software (Custom Install).
5. Install Software Products at Latest Level.
6. Install New Software Products at Latest Level.
7. Input device/directory for software.
8. Select your CD-ROM drive and then choose the components that you want to install.

   The CommonStore software will be installed in the directory **/usr/lpp/csld**. This directory is created if it does not exist.

Post-installation steps

After installing the software package, you must manually prepare the CommonStore for Lotus Domino Server in the following way:

1. Log in as user root.
2. Create an AIX user for CSLD, for example *csld01*. This creates a home directory (/home/csld01) and a profile document (/home/csld01/.profile) for this user on the CommonStore Server.
3. Set an initial password for the CSLD user.
4. Log in with the new user ID (csld01 in this example), and create a Lotus Notes data directory as a subdirectory of the CSLD user's home directory. Name this directory *notesdata*, for example.
5. You must provide a personal Lotus Notes address book (names.nsf) in the notesdata directory. Since the newly installed Lotus Domino server has not been configured or started, you cannot copy it from there. Therefore, copy a names.nsf file from a Lotus Notes client installation on a Windows system. Before you copy the file, check and modify it as described:
   a. Make sure that the port of the first location document in the names.nsf file is TCP/IP-enabled.

      The CSLD Task uses TCP/IP to connect to the Lotus Domino server. It takes the connection information from the first location document in the names.nsf file. This is why you must make sure that this location document is configured for TCP/IP. To check, you open the first location document and click the port tab.

   b. Add a connection document for each Domino server accessed by CommonStore for Lotus Domino to the names.nsf file. This is not a must, but it ensures a working communication between the CSLD Task and all Domino servers it might access.
6. Copy the modified names.nsf address book to the notesdata directory of the AIX computer hosting the CommonStore Server.
7. Copy the csld.id file to the data directory on the CommonStore Server (/home/csld01/notesdata in this example).

   This file holds a valid Lotus Domino user ID so that CommonStore can access the Domino server. You find the csld.id file in the notes or notes\data directory of the workstation that you used to create the ID.
8. Log on to the AIX computer hosting the CommonStore Server using your CSLD user ID (csld01 in this example). Copy the notes.ini file created during the installation of the Lotus Domino server to the home directory (/home/csld01 in this example).
9. Modify the notes.ini file so that it contains the following entries:

```
[Notes ]
Directory=/home/csld01/notesdata #Modified for CSLD (user csld01)
KitType=2
SetupDB=setupweb.nsf
UserName=CompanyName=
NotesProgram=/usr/dom50/lotus/notes/5080/ibmpow
#
#Added for CSLD
#
TCPIP=TCP,0,15,0
Ports=TCPIP
KeyFilename=/home/csld01/notesdata/csld.id
EXTMGR_ADDINS=libextpwd.a
```

10. If you modified the csmimes.properties file, for example because you had to add a content-type-to-MIME-type association, copy the modified file to the directory that the INSTANCEPATH keyword points to. This is usually the home directory.

Preparing the run-time environment

CommonStore for Lotus Domino needs access to the Notes C++ API and the Content Manager Client toolkit. In addition, it must be able to find the Java Runtime Environment (JRE). Thus, you must include the paths to these components in the environment settings for the CSLD AIX user (following the example from previous installation steps, this is user csld01). Proceed as follows:

1. Log on to the computer hosting the CommonStore Server with the AIX user ID for CSLD (following the example in "Post-installation steps" on page 75, this is csld01).
2. Copy the notesenv.sh file to this user's home directory:
   ```
   cp /usr/lpp/csld/bin/notesenv.sh
   ```
3. Modify the notesenv.sh file according to your environment. See the following example. The comments give you information on how to modify the settings:
   ```
   #!/bin/sh
   #
   #This script helps setting up the notes environment
   #needed to run CommonStore for Notes Domino

   #Set the path environment for the user
   #Adjust this for the local installation
   PATH=$PATH:/opt/lotus/notes/latest/ibmpow:/opt/lotus/notes/latest/ibmpow/res/C

   #Add the users notes data directory to the path
   PATH=$PATH:$HOME/notesdata

   #Settings for the notes client
   #Adjust this for the local installation
   Notes_ExecDirectory=/opt/lotus/notes/latest/ibmpow
   LOTUS=/opt/lotus

   #Users notes data directory
   NOTES_DATA_DIR=$HOME/notesdata

   #Export all settings
   export PATH Notes_ExecDirectory LOTUS NOTES_DATA_DIR
   ```
4. Copy the csenv.sh file to the home directory. This file is provided during the installation of CommonStore for Lotus Domino. From the home directory, enter the following command:
   ```
   cp /usr/lpp/csld/bin/csenv.sh
   ```
5. Edit the csenv.sh file and add or modify the entries as shown in the following example. Again, it is assumed that the CSLD user ID is csld01:
   ```
   #set CommonStore environment
   PATH=$PATH:/usr/lpp/csld/bin
   LIBPATH=$LIBPATH:/usr/lpp/csld/bin
   NLSPATH=$NLSPATH:/usr/lpp/csld/nls/%L/%N
   CSNBASE=/usr/lpp/csld/nls/$LANG
   CSNINSTANCEPATH=$HOME

   #Added for csld01
   FRNLOCAL=/home/csld01

   export PATH LIBPATH NLSPATH CSNBASE CSNINSTANCEPATH
   ```

6. Edit and modify the .profile file. This file is already in the CSLD user's home directory. Add lines as shown in the following example. Note that in the example, the CSLD user ID is csld01.

   ```
   PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.

   export PATH

   if [-s "$MAIL"] #This is at Shell startup.In normal
   then echo "$MAILMSG" #operation,the Shell checks
   fi  #periodically.
   #
   #Add CM Client toolkit environment
   ./home/cltadmin/frn/frnsetup.clt
   #
   #Add Notes environment
   ./home/csld01/notesenv.sh
   #
   #Add CSLD environment
   ./home/csld01/csenv.sh
   ```

7. Set the classpath environment variable so that it points to the Java Runtime Environment (JRE) or Java Development Kit (JDK). If you do not know where your JRE or JDK is located, search for the files JRE or java, respectively. You can test whether your classpath settings are correct. For example, for a JRE, you can enter the following command. For a JDK, replace the `jre` with `java`.

   ```
   jre -cp /usr/lpp/csld/bin/archcls.jar
   com.ibm.esd.commonstore.server.CSVersionCheck
   ```

   If the setting is correct, the response "IBM Content Manager CommonStore Version 8.1.x.x" is displayed.

Installing the CommonStore Server package on Windows

Install the CommonStore Server code by running the setup program in the CommonStore Server subdirectory on the CD-ROM. This starts an InstallShield that guides you through the installation process. To install the CommonStore Server, follow these steps:

1. Run the CommonStore for Lotus Domino InstallShield. You see the Welcome page.
2. Click **Next**. The License Agreement page opens.
3. Select **I accept the terms in the license agreement**.
4. Click **Next**. The Customer Information page opens.
5. Enter your user name and the name of your organization in the appropriate fields. Leave the radio button on **Anyone who uses this computer**.
6. Click **Next**. You reach the Select Type page.
7. Select the **Custom** radio button and click **Next**. You see the Custom Setup page.
8. Unless you want to install other components on the same computer, deselect all components except for *CommonStore Server*.

   You deselect components by clicking the icons next to their names and selecting **This feature will not be available** from the pop-up menu.
9. By default, the selected components are installed in folders under C:\Program Files\IBM\CSLD\server. To change this, click **Change** and select another folder. Click **OK** when finished to return to the Custom Setup page.

Note

When you change the installation folder, you must also change the corresponding entries in the server configuration profile (usually archint.ini).

10. Click **Next**. You reach a page with the heading *Ready to Install the Program*.
11. Click **Install**. The InstallShield wizard starts copying files. You see a progress bar. When the installation is successful, you see a page with the heading *InstallShield Wizard Completed*.
12. Click **Finish**.

Additional installation steps for Tivoli Storage Manager

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

1. Creating client option files
2. Setting environment variables for Tivoli Storage Manager API clients

Creating client option files

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

- ARCHIVE xx
- STORAGETYPE ADSM
- SERVER *servername*

there must exist a separate client option file *servername*.opt containing all Tivoli Storage Manager parameters required for connecting to the specified Tivoli Storage Manager server. It is recommended that you keep all client option files in a separate directory. To create the client option files, you can use the following procedure:

1. Copy the *dsm.opt* file, which comes with your Tivoli Storage Manager product package, to the appropriate directory.
2. Create another copy of the dsm.opt file under another name in the target directory. Specify the new name by replacing the prefix *dsm* with the required server name. You now have a copy of dsm.opt and the required *servername*.opt file.

All client option files must reside in the same directory. This directory must contain a copy of the *dsm.opt* file. Although CommonStore ignores the contents of dsm.opt, dsm.opt must exist, and the environment variable DSMI_CONFIG must point to it. To avoid potential errors, it is recommended that you delete the content of the dsm.opt file, that is, keep it as an empty file. See “Setting environment variables for Tivoli Storage Manager API clients”.

Setting environment variables for Tivoli Storage Manager API clients

Tivoli Storage Manager API clients need the following environment variables to locate certain files:

DSMI_CONFIG
: Fully qualified name (path and file name) of the client option file named dsm.opt

DSMI_DIR
Path to the directory in which the Tivoli Storage Manager message file named dscameng.txt resides

DSMI_LOG
Path to the directory in which the Tivoli Storage Manager error log named dsmerror.log resides

In addition, the location of the Tivoli Storage Manager API DLLs must be included in the PATH environment variable. See the following example:

1. Set DSMI_CONFIG to `C:\Program Files\IBM\CSLD\server\adsm_opt\dsm.opt`
2. Set DSMI_DIR to `C:\Program Files\Tivoli\tsm\api`
3. Set DSMI_LOG to `C:\Program Files\IBM\CSLD\server\adsm_opt`
4. Set PATH to `%path%;C:\Program Files\Tivoli\tsm\api\dll`

In this example, it is assumed that you have created a directory `C:\Program Files\IBM\CSLD\server\adsm_opt` containing all client option files with the extension *.opt. You can define these environment variables on the Environment page of the System Properties sheet. To open this window, follow these steps:

1. Click **Start** → **Settings** → **Control Panel**.
2. Double-click the **System** icon.
3. Click the **Advanced** tab.
4. Click the **Environment Variables** button.

Installed files

During the installation, the InstallShield wizard copies numerous files to your computer. This section discusses the most important of these files.

Programs

This section describes the most important programs installed by CommonStore. Not described are shared libraries used by CommonStore.

Note

The file names listed below are valid for AIX systems. If the CommonStore Server is installed on a Windows system, the file names listed without an extension also consist of the suffix *.exe*.

The CommonStore Server package contains the following executable files:

archadmin
This program allows a (remote) connection to a CommonStore Server in order to view the messages issued by the CommonStore Server. You can establish a connection across machine and platform boundaries.

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

Note

You only need this program if Tivoli Storage Manager is the archive system.

archagentod
: Same function as archagent.exe, but for Content Manager OnDemand. The CommonStore Server starts as many CMOD agents as are defined by the keyword ODAGENTS in the server configuration profile.

Note

You only need this program if Content Manager OnDemand is the archiving system.

archagentvi
: Same function as archagent.exe, but for Content Manager. The CommonStore Server starts as many Content Manager agents as are defined by the keyword VIAGENTS in the CommonStore Server profile.

Note

You only need this program if Content Manager is the archiving system.

archagentvi400
: Same function as archagent.exe, but for Content Manager for iSeries. The CommonStore Server starts as many Content Manager for iSeries agents as are defined by the keyword VIAGENTS in the CommonStore Server profile.

Note

You only need this program if Content Manager for iSeries is the archiving system.

archpro
: This is the continuously-running CommonStore Server main program that controls all other CommonStore components.

archservice
: This program installs and uninstalls the CommonStore Server as a Windows service.

archstop
: This program shuts down the CommonStore Server.

archcls.jar and csmimes.properties
: The archcls.jar file is a Java class library required for accessing the CommonStore Server by using the Hypertext Transfer Protocol (HTTP). Among other things, it contains the Web dispatcher and the Domino dispatcher functionality. The CommonStore Server starts a Java run-time process with as many Domino dispatcher threads as defined in the server configuration profile (usually archint.ini). The keyword for setting the number of threads is DOMINODPS.

The csmimes.properties file is a resource file, which is used by the Web dispatcher. It specifies the mappings of archive content types to MIME types. The Web browser needs this information to display the retrieved content correctly.

Sample configuration profiles for the CommonStore Server

The server configuration profile is also referred to as the initialization file or ini file. If the option *-i* is not used in connection with the **archpro** command, the CommonStore Server looks for the default server configuration profile (usually archint.ini) in the local directory.

The easiest way to create this file is by modifying the appropriate sample profile. You edit the sample profile that is suitable for your archive system, and then save it as archint.ini or whatever name might be required by your system setup. The server configuration profile contains all necessary information for the CommonStore Server and its connections to the archive systems. See "Basic setup" on page 87.

Notes:

1. The CommonStore Server needs only this file for customization (no side information, no environment variables).
2. The following sample configuration profiles are provided as part of the CommonStore package. They are intended for use as templates. In other words, after you have customized them according to your setup, they assume the role of the server configuration profile whenever you specify them using the option -i.

   archint_sample_tsm.ini
   Use this sample profile as a template when your archive system is Tivoli Storage Manager.

   archint_sample_cm.ini
   Use this sample profile as a template when your archive system is Content Manager.

   archint_sample_cmod.ini
   Use this sample profile as a template when your archive system is Content Manager OnDemand.

   archint_sample_cmod400.ini
   Use this sample profile as a template when your archive system is Content Manager OnDemand for iSeries.

   archint_sample_cm400.ini
   Use this sample profile as a template when your archive system is Content Manager for iSeries.
3. You can configure the CommonStore Server to work with different archive types. To do so, follow these steps:
   a. Open the appropriate sample profiles and copy the sections beginning with the ARCHIVE keyword to your server configuration profile (usually archint.ini).
   b. Change the settings in the copied sections so that they work with the archive systems that you want to use.
   c. Specify the number of agents to be employed by using the following keywords in the server configuration profile:
      - VIAGENTS for Content Manager archives

- ADSMAGENTS for Tivoli Storage Manager archives
- ODAGENTS for Content Manager OnDemand archives

If you are not sure about the settings to use, have a look at the other sample profiles or see Appendix A, "Keywords in the server configuration profile" on page 203.

Installing the CommonStore Server as a service

On a Windows machine, you can install the CommonStore Server component as a service. This allows the server to run continuously even if all users have logged off.

Notes:

1. Make the installation of the server as a service the last step in the installation procedure. See also "Hints" on page 83. The service functionality is implemented as a separate program named archservice.exe. This program must reside in the same directory as the other CommonStore Server programs, such as archpro.exe.
2. In addition to the service functionality, archservice.exe also implements the functionality to install and uninstall the CommonStore service. As soon as the service is installed, it appears in the **Services** window that you can open from the Control Panel. You can also start or stop the CommonStore service from this window. If you remove this service, the name of the service disappears from the **Services** list.
3. The program archservice.exe contains none of the CommonStore Server functionality. The server functionality is fully contained in the archpro.exe program and in the other CommonStore Server programs. When the CommonStore service is started, it starts the archpro.exe program, which in turn starts the other programs. When the service is stopped, it stops archpro.exe, which in turn stops the other programs. The CommonStore service checks every few seconds if archpro.exe is still running. When the service detects that this program has stopped, it waits for one minute and then restarts it.

Installing and uninstalling the CommonStore service

To install or uninstall the CommonStore service, follow these steps:

1. Open a Command Prompt window.
2. Switch to the directory in which the file archservice.exe resides.
3. Enter **archservice install** to install the service. To obtain help, you can enter **archservice -h**

When the service is installed, the Windows Service Control Manager runs the archservice.exe program. You cannot run archservice.exe from the command prompt without specifying parameters. This way of starting the service is restricted to internal calls of the Windows Service Control Manager. During the installation, registry keys are created in the Windows registry. All modifications of the registry are displayed in the Command Prompt window. You can view information about the success or failure of installing, removing, starting, or stopping archservice.exe in the Windows Application Event Log by using the Event Viewer tool.

Starting and stopping the CommonStore service

You can start and stop the CommonStore service using the **Services** program, which you can access from the Windows Control Panel. You can also do this from a Windows Command Prompt:

- To start the CommonStore service, enter **archservice start**.
- To stop the CommonStore service, enter **archservice stop**.
- To obtain status information about the CommonStore service, enter **archservice status**.

Multiple installations of the CommonStore service

You can install multiple instances of the CommonStore service on a single machine. Each instance must have a different name. You determine the name by using the command-line option *-n* when you install the CommonStore service.

In addition, each instance must use a separate fixed port. You specify this port in the server configuration profile, using the ARCHWIN_PORT keyword. This means that you must use a separate profile for each instance of the CommonStore service.

Examples:

- `archservice install -i c:\ibm\csld\instance02\archint.ini -n 2`
- `archservice remove -n 2`

The first command installs CommonStore as a service under the name CommonStore_2 . The second command removes the service CommonStore_2.

Hints

Read the following list of hints carefully. It reflects a thoroughly tested setup and might help you if you run into problems during the installation process.

- When you start the CommonStore Server for the first time, run the archpro program from the command line and complete the configuration process with the initial settings. Only install the CommonStore Server as a service if it runs without problems. You have achieved this when you no longer see any screen output. As you might need the error information for future reference, switch on tracing (TRACE=ON) in the server configuration profile during initial testing. The trace file keeps all warnings and error messages, even if you no longer see any screen output.
- When using Content Manager as an archive, you must run the CommonStore service under the account of a Content Manager user. The following procedure reflects the steps in a Windows 2000 system. The labels and controls on the panels differ from those in a Windows NT system. Bear this in mind if you still use Windows NT.
  1. Open the Services window by clicking **Start → Settings → Control Panel → Administrative Tools → Services**.
  2. Select the CommonStore service in the list and click **Action → Properties**. The CommonStore Service Properties window opens.
  3. Click the Log On tab.
  4. Click the **This Account** radio button.
  5. Enter the name of the Content Manager account in the field next to the radio button. If you cannot remember the name, click the **Browse** button to select the account from the Select User window.
  6. Click **OK** to close window.

7. Click **Close** to close the Services window.

This action automatically sets the appropriate environment variables for the service. If you leave the default setting and run the service under the System Account, these variables are probably not set properly.

- To run the CommonStore service under a normal user account, provide the account with the *Log on as a service* right. To do so, follow these steps:
  1. Open the Windows User Manager by clicking **Start → Settings → Control Panel → Administrative Tools → Local Security Policy**.
  2. Expand the Local Polices folder in the Tree pane on the left.
  3. Click the User Rights Assignment folder. A list of user rights becomes visible in the right pane.
  4. Right-click *Log on as a service* in the list and select **Security** from the pop-up menu. A window called Local Security Policy Setting opens.
  5. Click **Add**. The Select Users or Groups window opens.
  6. Select the Content Manager user account from the list and click **Add**. The account name is displayed in the lower pane of the window.
  7. Click **OK**.
  8. Click **OK** again to leave Local Security Policy Setting window.
  9. Close the Local Security Settings window.
- This right is assigned in **User Manager → User rights policy**. Please do not forget to activate the **Show advanced user rights** option.
- Furthermore, the exchange paths defined by the keywords BASEPATH and ARCHPATH are probably not available when you run CommonStore as a service. However, for the archpro program to start, all paths must be available. If the archpro program cannot verify that this is the case, it does not start. Again, the trace file contains all corresponding error messages.

Part 5. The CommonStore Server

Chapter 22. Administration

Basic setup

This section provides an overview of the basic steps you must perform to configure the CommonStore Server. The following steps are described:

1. Setting up the basic server configuration profile
2. Enabling the Hypertext Transfer Protocol (HTTP)
3. Creating the passwords to access the archive server
4. Starting the CommonStore Server using the configured profile

CommonStore Server configuration profile setup

The installation package includes a sample profile for each of the supported archiving systems:

- The archint_sample_tsm.ini profile for Tivoli Storage Manager
- The archint_sample_cm.ini profile for Content Manager
- The archint_sample_cm8.ini profile for Content Manager Version 8
- The archint_sample_cmod.ini profile for Content Manager OnDemand

Depending on the operating system that you use to run the CommonStore Server, the sample profiles are copied to one of the directories shown in Table 5.

Table 5. Where to find the sample profiles

AIX	/usr/lpp/csld
Windows	C:\Program Files\ibm\csld\server\instance01

They are intended for your use as templates. After you have customized them according to your setup, they will assume the role of the CommonStore Server configuration profile whenever you specify them using the option *-i*. For more information, see "Starting the CommonStore Server" on page 97.

The CommonStore Server reads the profile before it executes. Any change in this profile requires that you restart the server for the changes to take effect.

Syntactical rules

The following rules apply to the syntax of the entries in a server configuration profile:

- Every line is analyzed separately.
- Keywords can start in any column of the line.
- There must not be any characters — 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 the file is reached.
- It is strongly recommended that you do not use keywords as the values of keywords. However, you can use keywords as values when you *enclose them in single quotes*, for example: `SERVER 'VI'`.

Performing basic setup steps

For the basic setup, perform the following steps:

1. Copy the appropriate sample profile to a file named archint.ini.
2. Adjust the paths specified in the profile to reflect your environment. This usually involves setting the following keywords:
   - BINPATH
   - LOGPATH
   - TEMPPATH
3. Enter the settings for an archive created in your archive system. At this point, it is enough to specify a test archive in the server configuration profile. For each logical archive in the archint.ini file, the following settings (depending on your archive system) are necessary:

   Tivoli Storage Manager:
   - `STORAGETYPE   ADSM`
   - `SERVER`
   - `MGMT_CLASS`
   - `ADSMNODE`

   > **Note**
   > If your Tivoli Storage Manager server is named ADSM, enclose that name in single quotes, that is,`'ADSM'`.

   Content Manager:
   - `STORAGETYPE   VI`
   - `LIBSERVER`
   - `INDEX_CLASS`
   - `VIUSER`

   Content Manager Version 8:
   - `STORAGETYPE   CM`
   - `LIBSERVER`
   - `ITEM_TYPE`
   - `CMUSER`

   Content Manager for iSeries:
   - `STORAGETYPE VI400`
   - `LIBSERVER`
   - `INDEX_CLASS`
   - `VIUSER`
   - `TRUNCATE_ATTRIBUTE` (optional)

   Content Manager OnDemand:
   - `STORAGETYPE   ONDEMAND`
   - `ODHOST`
   - `APPGROUP`
   - `APPLICATION`
   - `FOLDER`
   - `ODUSER`

4. Set the system type of the server application from which you want to archive documents with CommonStore. Depending on the server application, add or change the SYSTEMTYPE statement in the server configuration profile as shown:

For Exchange 2000:	`SYSTEMTYPE EXCHANGESYSTEM`
For Lotus Domino:	`SYSTEMTYPE DOMINOSYSTEM`
For SAP R/3:	`SYSTEMTYPE SAPSYSTEM`

   If you have a mixed setup, and want to archive from two or all of these server applications, add the appropriate value to the statement. In the sample profile, all applications are enabled so that you probably do not have to change anything:

```
SYSTEMTYPE DOMINOSYSTEM EXCHANGESYSTEM SAPSYSTEM
```

5. Set the number of agents for your archive system. Use at least one agent for your archiving system, that is, set the value to 1 for the archive system employed.

Example:
CommonStore supports a mixed archiving environment. In one server configuration profile, you can specify agents for different archiving systems.

```
ADSMAGENTS              1
VIAGENTS                2
ODAGENTS                1
```

Configuring the CommonStore Server for the Hypertext Transfer Protocol

For communication by the Hypertext Transfer Protocol (HTTP), you must specify a valid Web port in the server configuration profile.

Passwords

When you start the CommonStore Server for the first time, you are prompted for passwords.

Start the CommonStore Server by entering the following command at the command prompt:

```
archpro -f serverpasswd
```

You are prompted for the passwords for each logical archive defined in the server configuration profile, the archint.ini file.

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

Getting started

Start the CommonStore server by issuing one of the following commands. Bear in mind that the archpro program always needs to know which server instance to start.

archpro
: When using the archpro command without additional parameters, enter it from the instance directory of the server instance that you want to start. Using the default setup with only one CommonStore Server, this is the instance01 directory.

archpro -i <*ini file*>
: Choosing this option, you can start the CommonStore Server from any directory. It also allows you to specify a server configuration profile other than archint.ini. Note that in addition to the server configuration profile (<*ini file*>), you must specify the relative or absolute path to the profile. See the following example:

```
archpro -i c:\Program Files\IBM\CSLD\instance02\archint2.ini
```

See "Starting the CommonStore Server" on page 97 for more information.

Creating multiple server instances

It is possible to run more than one instance of the CommonStore Server. In other words, several independent sets of CommonStore 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 CommonStore Server, it is necessary to maintain distinct server configuration profiles, one for each instance of the CommonStore Server. These profiles must reside in separate instance directories.

All profiles 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.

If you want to use a particular instance, change to the instance directory first, and enter all commands from there. Alternatively, include the option -i in all command invocations to specify the profile to be used (refer toChapter 23, "Working with the CommonStore Server" on page 97). For unassisted operation, it is recommended that you install corresponding CommonStore services on Windows (refer to "Installing multiple instances of the CommonStore service" on page 91) as appropriate.

The following steps are necessary for installing multiple instances:

1. Creating instance directories
2. Separating the server configuration profiles
3. Creating additional Windows services (optional)

Creating instance directories

To create multiple instances of the CommonStore Server on the same machine, you must place each instance in its own instance directory. For example, to create a second instance, create a directory named instance02 at the same level as the instance01 directory. Copy the content of the instance01 directory to the new directory. In the instance02 directory, change the settings in the server configuration profile (archint.ini) as needed. In the server configuration profile of each instance, set the INSTANCEPATH keyword to the instance directory. To run multiple instances as a Windows service, refer to "Installing multiple instances of the CommonStore service".

Separating the server configuration profiles

It is necessary that you place each server configuration profile in a distinct directory in the file system. Additionally, these profiles must contain different values for the following keywords:

- INSTANCEPATH. This keyword specifies the directory in which the profile itself resides and in which instance-dependent files are stored. In particular, make sure that the following keywords, which are connected with the INSTANCEPATH keyword, have different values in each server configuration profile:
  - The name of the server configuration profile itself (archint.ini)
  - TRACEFILE (if TRACE is not switched to OFF)
  - CONFIG_FILE (archint.cfg)
  - LOGPATH (if LOG is not switched to OFF)
  - QUEUEPATH
  - Nodelock file (containing the license passwords for the instance)
- The TCP/IP ports used for communication to remote CommonStore modules, that is:
  - WEBPORT (if the WEBDPS parameter is present and set to a non-zero value)
  - ARCHPRO_PORT (the port over which an instance accepts connections)

Installing multiple instances of the CommonStore service

You can install multiple instances of the CommonStore service on a single machine. Each instance of the service must have a different name. You define the names by using the `-n` option when entering the installation command from the command prompt.

Each instance must also use a separate fixed port. You set the port numbers using the ARCHPRO_PORT keyword in the server configuration profile. You must specify a separate configuration profile for each instance of the CommonStore service.

Examples:

- `archservice install -i c:\Program Files\IBM\CSLD\server\instance02\archint.ini -n 2`
- `archservice remove -n 2`

The first command installs CommonStore as a service under the name CommonStore_2. The second command removes the instance CommonStore_2.

License setup

This section describes the new license model of IBM Content Manager CommonStore 8.1 and how to setup licensing for both Try & Buy usage and productive usage.

License types and certificate files

CommonStore distinguishes between different license types, as shown in Table 6.

Table 6. 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	• **CSLD8.lic** for archiving from a Lotus Domino server • **CSX8.lic** for archiving from an Exchange 2000 server • **CSSAP8.lic** for archiving from an SAP R/3 system

CommonStore profile keywords

The following keywords and subkeywords concern the license setup:

- SYSTEMTYPE
- DOMINOSYSTEM
- EXCHANGESYSTEM
- SAPSYSTEM

The keyword SYSTEMTYPE can have the values DOMINOSYSTEM, EXCHANGESYSTEM, SAPSYSTEM, or a combination of two or all of these.

Unless you use the Try & Buy license, you must enroll a production license for each specified system type.

The keyword INSTANCEPATH specifies the path to the directory in which instance-specific files are stored. This is particularly important with regard to nodelock files, which contain the license passwords. Since there is one nodelock file for each instance of the CommonStore Server, this file cannot be stored in the bin directory when running more than one instance of the CommonStore Server. Instead, each instance must have its own nodelock file in the directory specified by the INSTANCEPATH keyword.

Enrolling productive licenses

To run the CommonStore Server with a productive license, you must register the license before you start the server. To enroll a productive license for an instance, call the archpro program with the *-f license* parameter from the instance directory:

```
> archpro -f license
>
>
>       ****************************************************************
>       * IBM Content Manager CommonStore - Server 8.1.0.0             *
>       * (c) Copyright IBM Corporation, 1997-2002. All rights reserved.*
>       * Build  92, Compiled at Mar 4 2002.                           *
>       ****************************************************************
>
> CSS0213I: Please enter the name of the certificate file:
```

Enter the path to the certificate file containing the productive license. For Lotus Domino, enter the path to the CSLD8.lic file. This file resides in the directory named license on the product CD-ROM.

If you enter the path correctly, the license password is extracted from the file and registered in the nodelock file, which is located in the directory specified by the INSTANCEPATH keyword. You can now start archpro for productive use.

Notes:

- You must enroll a production license for each instance of the CommonStore Server.
- If you want to run the same instance of the CommonStore Server for different applications, you must specify the values of the SYSTEMTYPE keyword accordingly in the server configuration profile (default archint.ini). For example, to use the same instance with SAP R/3 and Lotus Domino, specify the system type with the following values:

  ```
  SYSTEMTYPE SAPSYSTEM DOMINOSYSTEM
  ```

  When you have saved the profile, you must enroll the license for each application.

Enrolling the Try & Buy license

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

```
> archpro
>
>          ******************************************************************
>          * IBM Content Manager CommonStore - Server 8.1.0.0               *
>          * (c) Copyright IBM Corporation, 1997-2002. All rights reserved.*
>          * Build  92, Compiled at Mar 4 2002.                             *
>          ******************************************************************
>
> CSS0926E: Archpro could not find the certificate file
> CSS0910I: Trying to get a LUM Production License for IBM Content Manager
>          CommonStore
> CSS0916E: Could not get a LUM Production License for IBM Content Manager
>          CommonStore
> CSS0912I: Trying to get a LUM Try and Buy License for IBM Content Manager
>          CommonStore
> CSS0933I:
>          ******************************************************************
>          *                                                                *
>          * IBM Content Manager CommonStore for Exchange Server            *
>          * 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 fully 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 is saved in the nodelock file residing in the directory specified by the INSTANCEPATH keyword in your server configuration profile. It expires after 90 days.

Note:
Yon can enroll the Try & Buy license only once. If you try it again, archpro shows the following message:

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

Running more than one instance with the Try & Buy license

You can enroll the Try & Buy license only once. During the enrollment, a nodelock file is created in the INSTANCEPATH directory of the instance that you currently use. The license in this nodelock file expires after 90 days. If you want to run multiple instances using the Try & Buy license, copy this nodelock file to the remaining INSTANCEPATH directories, that is, to the INSTANCEPATH directories specified in the server configuration profiles of the other instances you defined.

Enabling the browser-viewing function

To display content, your Web browser reads the MIME type information that was assigned to a file type, and starts the appropriate plug-in for displaying this MIME type. You must prepare your browser for displaying archived content by mapping the content types of archived attachments to the appropriate MIME types. You do this in a file called *csmimes.properties*. MIME types are assigned to content types by use of the equals character. You first type the content type, then the equals character, and then the MIME type. There must be exactly one mapping per line. See the following example:

```
TIFF6=image/tiff
PDF=application/pdf
```

The csmimes.properties file must reside in the INSTANCEPATH directory, that is, the directory that the INSTANCEPATH keyword points to in the server configuration profile (usually archint.ini). See Appendix A, "Keywords in the server configuration profile" on page 203 for more information about keywords and their use.

A sample version of the csmimes.properties file is copied to the instance01 directory at installation time. If you run more than one instances of the CommonStore server, you must have a copy of this file in each instance directory.

If CommonStore cannot find a copy of csmimes.properties in the INSTANCEPATH directory, it checks the directories specified in the BINPATH statement. If it can neither find a copy of csmimes.properties in any of these directories, it reads the required information from the compressed sample version, which is included in the *archcls.jar* Java archive. This file is also copied at the time you install the CommonStore server. The sample version contains the following mappings:

```
FAX=image/tiff
TIFF6=image/tiff
PDF=application/pdf
ALF=application/x-alf
OTF=application/x-otf
WAV=audio/wav
HPG=application/vnd.hp-HPGL
MP3=audio/x-mpeg
MPG=video/mpeg
MPEG=video/mpeg
JPG=image/jpeg
GIF=image/gif
TXT=text/plain
HTM=text/html
XML=text/xml
```

In most cases, you must edit and customize the sample file because it probably does not cover all your needs. In a normal setup with only one instance of the CommonStore Server, edit the copy in the instance01 directory.

Important

- To apply the changes that you made in any copy of csmimes.properties, you must restart the corresponding instances of the CommonStore Server (archpro).
- If you use the sample file as a basis, check if the content types match the content types that you defined in your archive. For example, if an .mp3 file has MPEG3 as its content type, the mapping `MP3=audio/x-mpeg`, as included in the sample file, does not work. You would have to change it to `MPEG3=audio/x-mpeg`.

Chapter 23. Working with the CommonStore Server

This section explains how to start and stop the CommonStore Server in various run time environments, including operation as a Windows service. It also tells you how to turn off the initial archive availability check. This section uses syntax diagrams to illustrate the syntax of commands that you enter at the command prompt. If you are not familiar with reading syntax diagrams, read the brief explanation in Appendix H, "Reading syntax diagrams" on page 261.

Starting the CommonStore Server

The CommonStore Server consists of several components. These components are all controlled by `archpro`, the continuously-running CommonStore main program. To start the entire CommonStore Server, it is sufficient to enter `archpro` at the command prompt. The archpro program then reads the server configuration profile (either the archint.ini file 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 prompt is as follows:

archpro

```
►►──archpro──┬──────────────────────────────────────────────┬──────────────►◄
             ├─-f  serverpasswd──┬─────────────────────────┬─┤
             │                   └─srv──┬────────────────┬─┘ │
             │                          └─node──┬──────┬─┘   │
             │                                  └passwd┘     │
             ├─ -i──ini file────────────────────────────────┤
             ├─ -n──name────────────────────────────────────┤
             └─ -h──────────────────────────────────────────┘
```

-f serverpasswd *[srv [node [passwd]]]*
: Use this parameter to specify the passwords for Tivoli Storage Manager, Content Manager, and Content Manager OnDemand. You only have to do this once, when you set up CommonStore.

-h
: Specify this parameter to obtain online help for the archpro command. This parameter is exclusive. If you specify it, you cannot use any other parameter.

-i *ini file*
: If archpro is started without any parameters, the server configuration profile must reside in the current directory and must be named *archint.ini*. If you want archpro to use a profile located in a directory other than the current one or a profile that is not named archint.ini, you can specify the parameter -i followed by path and file name of the profile you want to use.

-n *name*
: The parameter *-n* followed by a name specifies the CommonStore instance that you want to use. This is necessary when the CommonStore Server is already installed as a Windows service (see "Installing the CommonStore Server as a service" on page 82). The option is ignored if CommonStore is not installed as a service.

Examples:

- `archpro`
- `archpro -i /usr/lpp/csld/bin/archint.ini2` (AIX)
- `archpro -i C:\Program Files\IBM\CSLD\server\instance02\archint.ini2 -n 2` (Windows)

As soon as CommonStore is running, the main component (archpro) checks the other child processes every few seconds. If the archpro program detects that a process is no longer active, it automatically restarts the corresponding component. The archives are not checked for availability before a restart.

Starting CommonStore as a service

To run the CommonStore Server continuously, even if all users have logged off, you must start CommonStore as a service. You must install the service before you can use it. See “Installing the CommonStore Server as a service” on page 82 for more information. You can start the CommonStore service in the following ways:

- Enter **archservice start** in a Command Prompt window.
- Use the Services program that comes with Windows 2000. To do so, follow these steps:
  1. Open the Services window by clicking **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**.
  2. Select the CommonStore service in the list and click the **Start** button.
  3. Click **Close** to exit the Services program.

When the service is running, it starts the archpro program. It then checks regularly whether archpro is still active, and restarts it if necessary.

Turning archive availability checks off

By default, the archpro program verifies the connections to all configured archive servers when you start the CommonStore Server. It also checks whether certain internal attributes exist in the archives. CommonStore for Lotus Domino does not use these attributes. Therefore, turn off this check by changing the setting of the CHECK_ARCHIVE_SERVER keyword in the server configuration profile as shown:

```
CHECK_ARCHIVE_SERVER=OFF
```

If the check remains enabled, you receive an error message and the archpro program does not start.

Stopping the CommonStore Server

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

The syntax for this command is as follows:

archstop

```
►►─archstop─┬─┬──────────────┬──┬─────┬─┬──────────────────────────────────►◄
            │ ├─ -p─port─────┤  └─now─┘ │
            │ └─ -i─ini file─┘          │
            └─ -h───────────────────────┘
```

-h Specify this parameter to obtain online help for the **archstop** command. This parameter is exclusive. If you specify it, you cannot use any other parameter.

-p *port*
Specifies the port, where *port* is the fixed port number as defined in the server configuration profile by the ARCHPRO_PORT keyword.

-i *ini file*
Instead of the port, you can specify the path and the file name of the server configuration profile. In this case, the port number is taken from the profile.

now
Causes the server to stop immediately. If this parameter is left out, CommonStore first finishes the active jobs before it terminates.

Examples:

- `archstop`
- `archstop -p 5510`
- `archstop -p 5510 now`
- `archstop -i C:\Program Files\IBM\CSLD\instance01\archint.ini`

You can stop the CommonStore service from the Windows 2000 Services application or by issuing the **archservice stop** command. The archstop command cannot be used to stop the service. The archstop command only stops the archpro program; however, the service soon detects that archpro is down and restarts it automatically.

Stopping the CommonStore service

When you started the CommonStore Server as a Windows service, you must stop this service to terminate the archpro program. If you just end the archpro program, the service detects that it is no longer running, and, consequently, restarts it.You can stop the CommonStore service in the following ways:

- Enter **archservice stop** at a Windows Command Prompt.
- Use the Services application that comes with Windows 2000. To do so, follow these steps:
  1. Open the Services window by clicking **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**.
  2. Select the CommonStore service in the list and click the **Stop** button.
  3. Click **Close** to exit the Services application.

This stops the CommonStore service, and, as a result, terminates the archpro program.

Chapter 24. Files created at run time

CommonStore for Lotus Domino creates a number of files at run time. The most important ones are discussed in this section.

Trace files

The CommonStore Server can produce different trace files to provide you and the support team with information about error cases. You can configure tracing in the CommonStore Server profile. The following trace files are available:

archint_startup.trace
: This file contains only error information on starting and stopping the CommonStore Server.

archservice.trace
: This file contains only error information on starting and stopping the CommonStore service.

archint.trace
: This file is written by the CommonStore Server if specified in the server configuration profile. You can also change the name of this file in the server configuration profile. The file contains all CommonStore Server trace information, including information about starting, stopping, file names, and errors.

CommonStore Server log file

The log file produced by the CommonStore Server gives you detailed information about the most recent operations or events. You can use the data in the server log to display information about performance bottlenecks or errors in self-developed applications. The server log file contains one line for each operation. These lines contain an error code. If you encounter problems, you can look up the error codes in the server log file. Unless your errors go back to a misconfigured setup, you rarely need to switch on the additional trace facility.

The CommonStore Server log file is basically a table. The first block of columns is the same under all conditions, whereas the second block varies with respect to the type of the operation. The structure of the first block of columns is reflected in the example in Table 7.

Table 7. The common block of columns in the server log file

1	2	3	4	5	6	7	8
Time stamp	**Return code**	**Job number**	**Operation**	**Archive ID**	**CS server exec. time**	**Agent exec. time**	**Agent PID**
18:47:32	-50	17	Archive	A1	0.456	0.123	A123F
14:02:21	0	16	Append	A1	0.217	0.035	B14C
...	...	...	...	...	...	...	...
09:42:04	0	1	Att Search	A1	0.158	0.020	34FC

The following list briefly explains the meaning of the data in each column:

Time stamp
Points of time when the CommonStore Server received the jobs for processing.

Return code
The validation codes that the CommonStore Server returned for each processed job. A return code of 0 indicates a successful operation. All other codes indicate an error.

Job number
The numbers assigned to the jobs that were processed by the CommonStore Server.

Operation
Indicates the type of an operation or the stages in the process of completing it. See Table 8 for reference.

Table 8. Operation types as indicated in the server log file

Value	Meaning
Archive	Archiving
Retrieve	Retrieval
Part-Retrieve	Partial retrieval
Delete	Deletion
Query	Searching the archive for documents that match a query pattern created with the search pattern template of Lotus Domino
Update	Replacement of an archived document with a newer version

Archive ID
The logical archive IDs of the archives addressed in the operations, as specified in the server configuration profile.

CS server exec. time
The times that the CommonStore Server needed to process the jobs. The numbers reflect the times in seconds.

Agent exec. time
The times that the agent needed to process the jobs. The numbers reflect the times in seconds.

Agent PID
The identification numbers (process IDs) that an agent assigns to the jobs that it processes. These are hexadecimal values.

Variable columns for operation type *Archive*

If the operation type of a job is *Archive*, the second block of the CommonStore Server log file shows the columns of the example in Table 9.

Table 9. Columns in the server log for operation type Archive

9	10	11
DocID	**Content type**	**Full source file name**
2000021421141232#405A.0908	FAX	D:\cstore\test\notice.txt
...	...	...
1000911493141728#405A.0908	TXT	D:\cstore\test\invoice.txt

Variable columns for operation type *Retrieve*

If the operation type of a job is *Retrieve*, the second block of the CommonStore Server log file shows the columns of the example in Table 10.

Table 10. Columns in the server log for operation type Retrieve

9	**10**	**12**
DocID	**ComponentID**	**Full target file name**
2000021421141232#405A.0908	data	D:\cstore\test\notice.txt
...	...	...
1000911493141728#405A.0908	data	D:\cstore\test\invoice.txt

Variable columns for operation type *Part-Retrieve*

If the operation type of a job is *Part-Retrieve*, the second block of the CommonStore Server log file shows the columns of the example in Table 11.

Table 11. Columns in the server log for operation type Part-Retrieve

9	**10**	**11**	**12**
DocID	**ComponentID**	**Offset**	**Length**
2000021421141232#405A.0908	data	21373.0	208.0
...	...	...	...
1000911493141728#405A.0908	data	14577.0	217.0

If an operation was successful, the value of the **Length** field reflects the actual length of the retrieved item. Otherwise, the **Length** field shows the length value of the request.

Variable columns for operation type *Delete*

If the operation type of a job is *Delete*, the second block of the CommonStore Server log file shows the columns of the example in Table 12.

Table 12. Columns in the server log for operation type Delete

9	**10**
DocID	**ComponentID**
2000021421141232#405A.0908	data
...	...
1000911493141728#405A.0908	data

Variable columns for operation type *Query*

If the operation type of a job is *Query*, the number of columns in the second block of the CommonStore Server log file varies according to the number of arguments that you specified for the query. See Table 13 on page 104 for an example.

Table 13. Columns in the server log for operation type Query

9	10	11	12	13	14	15	16
Number of hits	**Search pattern template**	**Attribute name**	**Operator**	**Attribute value**	**Attribute name**	**Operator**	**Attribute value**
5	p1 and p2	Last name	==	Jones	First name	!=	Tom
...	...	...	...	...	...	...	...
16	p1	Last name	Rogers				

A query request consists of a set of basic search terms, which are combined by the search pattern template. Each of these search terms occupies three columns because it consists of an attribute name, an operator, and an attribute value. Query requests are recorded completely in the CommonStore Server log file. The number of hits is also shown in this log file, but the individual hits are not.

Variable columns for operation type *Update*

If the operation type of a job is *Update*, the second block of the CommonStore Server log file shows the columns of the example in Table 14.

Table 14. Columns in the server log for operation type Update

9	10	11	12	13
DocID	**Action**	**Target workbasket**	**From folder**	**To folder**
2000021421141232#405A.0908	attributes +workbasket +folder	WORKFLOW_WB	Folder1	Folder2
...	...	...	...	...
1000911493141728#405A.0908	attributes +workbasket +folder	STACK_1	New	Old

The values of the **Action** field are converted to strings and written to the CommonStore Server log file. The **Target workbasket**,**From folder**, and **To folder** fields are left empty when you do not specify a value for these in the request message.

Variable columns for operation type *GetAttrSpec*

There are no additional variable columns for this operation type.

Part 6. Configuring the archives

Chapter 25. Configuring Tivoli Storage Manager

Tivoli Storage Manager (formerly: ADSM) provides automated, centrally-scheduled, policy-managed backup, archive, and space management services for file servers and workstations in a multivendor environment. It is capable of storing data on a wide variety of media, including hard disks, magnetic tapes, and optical disks. The information on the physical location of individual data blocks is kept in an internal database. This database enables Tivoli Storage Manager (TSM) to locate the data without having to access the physical media. It only accesses the storage devices if data is stored or retrieved. The TSM database also contains the TSM server setup information. In storing documents, CommonStore makes use of TSM's archiving facilities. However, its Hierarchical Space Management (HSM) capabilities are not exploited.

To prepare Tivoli Storage Manager for use with CommonStore, you must perform these steps:

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

Before you begin, make sure that the Tivoli Storage Manager server is already installed and properly configured.

Creating Tivoli Storage Manager management classes and nodes

Connect to the TSM server as administrator by entering, for example, the following command at a Windows Command Prompt:

```
dsmadmc -se=<servername>
```

Then enter the following commands[1]:

1. `register admin` *msxadmin <password>*
2. `grant authority` *msxadmin* `class=system`
3. `def stgpool` *archpool TAPE_DEV* `maxscr=`*10*
4. `def stgpool` *dummy* `DISK`
5. `def domain` *msx*
6. `def policyset` *msx msx_policy*
7. `def mgmtclass` *msx msx_policy msx_mgmt*
8. `def copygroup` *msx msx_policy msx_mgmt* `type=archive destination=`*archpool* `ser=static retver=`*9999*
9. `def copygroup` *msx msx_policy msx_mgmt* `type=backup destination=`*dummy*
10. `assign defmgmtclass` *msx msx_policy msx_mgmt*
11. `validate policyset` *msx msx_policy*
12. `activate policyset` *msx msx_policy*
13. `register node` *cstore <password>* `domain=`*msx* `archdelete=yes`

Explanation:

1. Enter the character strings in `monospace` verbatim; replace the character strings in *italics* so that they reflect your environment settings.

Steps 1 and 2 on page 107 create an additional Tivoli Storage Manager administrator with all rights (`class=system`).

Step 3 on page 107 creates a primary storage pool named *archpool* in device class TAPE_DEV with a capacity of ten scratch tapes. The device class defines the available storage devices (disk drives, tape drives, optical disk drives). It is assumed that the device class TAPE_DEV has already been defined.

Step 4 on page 107 creates a dummy storage pool in device class DISK. DISK is a predefined Tivoli Storage Manager device class. It is not used to store any data. It merely satisfies a requirement for the definition of the backup copy group in step 9 on page 107.

Steps 5 to 7 on page 107 define a policy domain named *msx*, a policy set named *msx_policy* in this domain, and a management class named *msx_mgmt* in this policy set.

Step 8 on page 107 and step 9 on page 107 create the archive copy group and the backup copy group in the *msx_mgmt* management class. The archive copy group defines the storage pool in which the data is stored physically (in this example, it is stored in *archpool*). It also defines how long Tivoli Storage Manager retains the data (in this example: for 9999 days). The backup copy group is not used by CommonStore. However, the application programming interface of Tivoli Storage Manager requires that the default management class contains a backup copy group. Hence, you must define a dummy backup copy group.

Step 10 on page 107 declares *msx_mgmt* as the default management class.

Steps 11 on page 107 and 12 on page 107 validate and activate the previous definitions.

Step 13 on page 107 creates a Tivoli Storage Manager node named *cstore* in the policy domain *msx* and allows the client program (CommonStore) to delete archived data.

Using Tivoli Storage Manager options

CommonStore supports the following Tivoli Storage Manager options:
- PASSWORDACCESS PROMPT
- PASSWORDACCESS GENERATE

Both options are specified on the client side rather than on the Tivoli Storage Manager server. See the necessary settings for each option below.

Tivoli Storage Manager option settings

Table 15 on page 109 shows the settings for the options PASSWORDACCESS PROMPT and PASSWORDACCESS GENERATE.

Table 15. Settings for the Tivoli Storage Manager options PASSWORDACCESS PROMPT and PASSWORDACCESS GENERATE

	PASSWORDACCESS PROMPT	**PASSWORDACCESS GENERATE**
dsm.sys or<*my_srv*>.opt	SERVERNAME <*my_srv*> PASSWORDACCESS PROMPT	SERVERNAME <*my_srv*> PASSWORDACCESS GENERATE NODENAME <*my_node*>
archint.ini	ARCHIVE <*xx*> STORAGETYPE ADSM SERVER <*my_srv*> MGMT_CLASS <*my_mgmt*> ADSMNODE <*my_node*>	ARCHIVE <*xx*> STORAGETYPE ADSM SERVER <*my_srv*> MGMT_CLASS <*my_mgmt*>

The node is specified in the <*my_srv*>.opt file or in the server configuration profile, but never at both locations (see the table above).

The use of PASSWORDACCESS PROMPT presupposes that you specified a password for the Tivoli Storage Manager node that the option relates to. You must specify the same password when you install CommonStore:

```
archpro -f serverpasswd
```

This password is used for all connections. When it expires, update it on the Tivoli Storage Manager server and (if a new password is used) on the CommonStore Server.

The use of PASSWORDACCESS GENERATE presupposes that you manually set an initial password for the node that the option relates to. You must specify this initial password when you connect to the Tivoli Storage Manager node for the first time. Tivoli Storage Manager changes the initial password to an automatically generated one after the first access. Later, Tivoli Storage Manager updates the generated password automatically. The generated password is stored in a safe place on the client machine and on the Tivoli Storage Manager server. All subsequent connections are established by using the generated password.

Connect to the Tivoli Storage Manager node for the first time by entering the following commands in a Command Prompt window:

```
dsmc -se=<my_srv>             <-- login on ADSM server
dsmc> q mgm                   <-- query management classes
```

After entering the command for querying the management classes, Tivoli Storage Manager prompts you for the initial password.

Recommendation

PASSWORDACCESS PROMPT is easy to set up. This setting is recommended for initial testing. PASSWORDACCESS GENERATE is a little more difficult to set up, but once this step is done, you no longer need to concern yourself with passwords and password expiration. The latter is therefore the preferable solution for everyday use of CommonStore.

Tivoli Storage Manager – troubleshooting

Problem: The PASSWORDACCESS GENERATE option does not work.

Solution: Make sure that the node name is not specified in the server configuration profile, but rather in <*my_srv*>.opt

Important

The PASSWORDACCESS GENERATE option has been verified to work with the Tivoli Storage Manager application programming interface (API) version 3.1.3. It does not work with API Version 3.1.0.

Chapter 26. Configuring Content Manager (VisualInfo)

With Content Manager, IBM provides a multi-platform client/server solution for capturing, archiving and administering documents and data, covering the entire spectrum of corporate operations. Content Manager is a multi-purpose document management system that supports all kinds of information regardless of type and origin. Documents created with other applications can even be stored in common folders in order to provide archive users with convenient access to important business data.

Central or distributed archives:

Content Manager supports the concept of distributed information management. You can implement multiple object servers within a common enterprise-wide archiving solution, allowing documents and data to be stored directly at the location of their users. This minimizes network utilization and facilitates fast access to all kinds of information. Users can access all distributed archives directly from their desktop workstation, regardless of server location and system platform.

The solution for your environment:

In order to meet individual customer requirements for the configuration of archive solutions, Content Manager supports several different operating systems:

- Windows 2000, Windows NT
- AIX
- MVS/ESA™

This unique scalability allows to install the main archive on a powerful UNIX or mainframe computer, whereas smaller archives for decentralized organization structures can be implemented on low-cost PC servers. You can add servers or jukeboxes to extend the Content Manager archiving solution at any time.

IBM's well-known System Managed Storage technology (SMS) is another advantage of Content Manager. Based on a customizable setup, SMS automates the transmission of data from one archive to another and supports the time-controlled migration of documents between different storage media.

Usability:

With the graphical user interface of Content Manager, you can search and retrieve all important documents from the archiving system, whether they are scanned originals (such as letters or invoices), faxes, text files, spreadsheets, graphic files, or audio-visual recordings. By using Content Manager, you can store all these types of information in collective files and thus deal with them in a convenient manner. Furthermore, Content Manager lets you automate recurrent jobs using the built-in workflow component.

To prepare Content Manager for use with CommonStore, you must perform these steps:

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

3. Create a Content Manager workbasket.

Important

You cannot use Content Manager archives and Content Manager for iSeries archives with the same instance of the CommonStore Server.

Creating Content Manager user accounts

CommonStore needs its own log-in user. In Content Manager, you can create a user account by following these steps:

1. Start the System Administration Client.
2. To log in, select a user ID with administrative rights.
3. Select and open the **Users** folder.
4. Right-click somewhere in the folder and select **New** from the pop-up menu.
5. Enter the required information. Make sure that the new user has administration privileges.

Creating Content Manager index classes

In Content Manager, documents of the same type are grouped in index classes. Documents of the same index class have the same set of descriptive search attributes. CommonStore maps document properties to index attributes. For every document type (form) that you want to archive, you must create an index class. For every document property that you want to search for in the archive, you must create an attribute.

To create an index class, you must first define the necessary key fields. To do this, follow these instructions:

1. Log on to the Content Manager System Administration Client and open the **Fileroom** folder.
2. Open the **Key Fields** folder.
3. Right-click somewhere in the folder and select **New** from the pop-up menu. Repeat this step until you have defined the necessary attributes.
4. Open the **Index Classes** folder.
5. Right-click in the folder and select **New** from the pop-up menu. Fill in the requested fields and select the key fields to define the index class.

Multiple archives can — but do not need to — share the same index classes.

For a given document property type, you can determine the Content Manager attribute type from Table 16.

Table 16. Content Manager attribute types

Property type (form field)	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

For every index class to archive documents to, you must define a logical archive in the server configuration profile (usually archint.ini) to make it known to CommonStore. You can then use the logical archive ID in a form-to-archive mapping as a valid target.

CommonStore logs on to the archive with the CommonStore archive ID specified in the server configuration profile. Provide this ID with full access rights to the index class. Create a separate user ID just for CommonStore. Whether other archive user IDs must be granted access to your index classes depends on your requirements. If the archive is only accessed by CommonStore, restrict access to the index class to the CommonStore archive ID and the archive administrator.

Creating Content Manager workbaskets

To create a Content Manager workbasket, follow these instructions:

1. Log on to the Content Manager System Administration Client and open the **Work Management** folder.
2. Open **Workbaskets** folder.
3. Right-click somewhere in the folder and select **Create workbasket** from the pop-up menu.
4. Provide the requested information, but do *not* select the **Overload** function or the **Remove items after indexing** functions.

Preparing Content Manager for folder archiving

To archive and restore complete folder structures, the CommonStore archive administrator must define a Content Manager index class to hold internal CommonStore information. Using the security features of the archive, the administrator must ensure that users cannot access the index class by browsing through the folders. Document mappings for the index class must not exist to ensure that users cannot access the internal information.

The index class must contain the following attributes:

CSLDOrigUser
Variable character string, extended alphanumeric. The attribute must be able to hold the longest possible user name in canonical format; the recommendation is 256 characters.

CSLDOrigDB
Variable character string, extended alphanumeric. The attribute holds the replica ID of the Lotus Notes database that contains the original of the archived folder. Size this attribute to 17 characters.

CSLDFolderName
Variable character string, extended alphanumeric. The attribute holds the name (or hierarchical name structure) of the folder or subfolder. Size the attribute to 256 characters or more to archive deeper folder structures.

CSLDFolderAlias
Variable character string, extended alphanumeric. The attribute holds possible aliases of the archived folder. Size the attribute to at least 100 characters. However, it can be shorter if you did not define aliases for the mail client application.

You can choose an arbitrary name for the index class. Then, create a logical archive ID for the index class in the server configuration profile (usually archint.ini).

Finally, enter the name of the archive ID in the **Folder Archive ID** field of the profile document used by your archiving or retrieval task.

Important

In Tivoli Storage Manager, you can only archive folders without preserving the folder structure.

Content Manager – troubleshooting

If you encounter problems with CommonStore and Content Manager, try to track down and solve the problem by using the following procedure:

1. Stop the CommonStore Server.
2. Specify TRACE ON DEBUG in the server configuration profile.
3. Switch on folder manager tracing by entering the following command in a Command Prompt window:
   ```
   FRNFMTRACE=/p
   ```
4. Manually restart the CommonStore Server by using the archpro command in the same window.
5. Check the resulting trace file (usually archint.trace) and the folder manager protocol of Content Manager, the fmtrace.txt file.
6. Check the Content Manager error log on the library server to which the problematic logical archive refers.
7. See the book *IBM EDMSuite ImagePlus VisualInfo: Messages and Codes* for detailed error descriptions or the *IBM EDMSuite ImagePlus VisualInfo: System Administration Guide* for further trace information.
8. When in doubt, double-check the Content Manager key fields, the index classes, and workflow definitions.

Chapter 27. Configuring Content Manager Version 8

The steps for configuring Content Manager Version 8 are basically the same as for previous versions of Content Manager. However, there are a few differences regarding the setup and the naming of objects on the System Administration panel. These subjects are discussed in this chapter.

With Content Manager Version 8, CommonStore currently does not offer the same range of features as with earlier versions of Content Manager. Thus, you cannot perform the following actions:

- Archive folder structures
- Retrieve folders and subfolders
- Search in the archive
- Archive to workbaskets
-

Preparing CommonStore for use with Content Manager Version 8

CommonStore connects to Content Manager Version 8 through a local connector, which is part of the Content Manager Version 8 API. The API is included in the Enterprise Information Portal (EIP), which is delivered with Content Manager Version 8. Before you can use the CommonStore, install the following software on all computers running an instance of the CommonStore Server:

- Enterprise Information Portal
- DB2 Client

Setting the classpath environment variable:

CommonStore needs access to the .zip and .jar files of the EIP local connector. Therefore, you must include the path to these files by setting the classpath environment variable accordingly. Include the paths to the following files in the classpath statement:

- cmbcm81.jar
- cmbicm81.jar
- cmbicmc81.jar
- db2java.zip
- xerces.jar
- cmblog4j81.jar
- log4j.jar
- cmbsdk81.jar

Example:

```
classpath=C:\CMBROOT\cmbcm81.jar;C:\CMBROOT\cmbicm81.jar;
C:\CMBROOT\cmbicmc81.jar;C:\SQLLIB\java\db2java.zip;C:\CMBROOT\xerces.jar;
C:\CMBROOT\cmblog4j81.jar;C:\CMBROOT\log4j.jar;C:\CMBROOT\dmbsdk81.jar
```

In this example, it is assumed that you have installed the EIP in the C:\CMBROOT directory and the DB2 Client in the C:\SQLLIB directory. You can change the setting of the CLASSPATH environment variable on the Environment page of the System Properties sheet. To open this window in Windows 2000, follow these steps:

1. Click **Start → Settings → Control Panel**.
2. Double-click the System icon. The System Properties window opens.
3. Click the Advanced tab.
4. Click **Environment Variables**.
5. In the lower box labelled System variables, select the **classpath** environment variables and click **Edit**. If the classpath environment variable is unset, click **New**.
6. Append the paths to the .jar and .zip files to the current setting, or, if the classpath variable is unset, enter the paths in **Variable Value** field.
7. Click **OK** until the System Properties window closes.

Creating user accounts in Content Manager Version 8

To create new users in Content Manager Version 8, follow these steps:

1. Start the Content Manager System Administration program and log on.
2. Expand the **Authentication** item in the tree-view on the left.
3. Right-click the item **Users** in the expanded view.
4. Select **New** from the context menu. A wizard starts that guides you through the remaining steps.

Structuring Content Manager Version 8 archives

In Content Manager Version 8, documents of the same type are grouped in item types. Documents in an item type have the same set of descriptive search attributes. CommonStore maps document properties to attributes. For each document property that you want to search for in the archive, you must create an attribute. For every document type or form to be represented in the archive, you must create an item type. Hence, to make CommonStore work with Content Manager Version 8, you must perform the following tasks:

1. Creating attributes
2. Creating item types

Creating attributes in Content Manager Version 8

The procedure of creating attributes in Content Manager Version 8 is different to that of older Content Manager versions. Follow these steps:

1. Log on to the Content Manager System Administration Client.
2. Expand the **Data Modelling** item in the tree-view on the left.
3. Right-click the item **Attributes** in the expanded view.
4. Select **New** from the context menu. A wizard starts that guides you through the remaining steps.

When creating attributes for your document properties, select a suitable Content Manager attribute type according to the Lotus Notes property type. The types you can choose are shown in Table 17 on page 117.

Table 17. The proper Content Manager attribute types for various Lotus Notes property types

Lotus Notes property type (form field)	Content Manager attribute type
Text	Character Variable character CLOB BLOB
Number	Short integer Long integer Decimal Double
Date only	Date
Time only	Time
Date and Time	Time stamp

Important

Running CommonStore with Content Manager Version 8, you can only use first-level attributes.

Creating item types

In version 8 of Content Manager, the term *index class* is replaced with the term *item type*. Follow these steps to create item types:

1. Start the Content Manager System Administration Client and log on.
2. Right-click the item **Item Types** in the tree-view on the left.
3. Select **New** from the context menu. A tabbed notebook opens with the *Definition* page in front.
4. On the Definition page, enter a name for the item type in the **Name** field.
5. From the **Item type classification** drop-down list, select **Document**.
6. Click the Document Management tab.
7. On the Document Management page, click **Add**. A window with the title Define Document Management Relations opens.
8. From the **Part type** drop-down list, select **ICMBASE** and click **OK**. The selected part type is listed in the box on the Document Management page.

   ICMBASE is a required part type for basic parts, holding document content, such as attachments, images, and so on.
9. Click the Attributes tab. The available attributes are listed in the box on the left.
10. Select the attributes that you want to make searchable in the archive and click **Add**. The selected attributes are displayed in the *Selected attributes and components* box on the right.

    To remove attributes from this box, select one or more attributes and click **Remove**. For more information about attributes, see “Creating attributes in Content Manager Version 8” on page 116.
11. Click **OK** to complete the item type.

To make the item types known to CommonStore, define a logical archive for each item type in the server configuration profile (usually archint.ini) by using the ITEM_TYPE keyword. You can then use the logical archive ID in a form-to-archive mapping. Multiple archives can – but need not – share the same item type. None of the attributes defined by the ITEM_TYPE keyword is set to required. Besides, the attributes are all single-value fields.

CommonStore logs on to the archive with the CommonStore archive user ID specified in the server configuration profile. Provide this ID with full access rights to the item type. Create a separate user ID just for CommonStore. Whether other archive user IDs must be granted access to your item types depends on your requirements. If the archive is only accessed by CommonStore, restrict access to the CommonStore user ID and the archive administrator.

Content Manager Version 8 troubleshooting

If you encounter errors or unexpected behavior in connection with CommonStore and Content Manager Version8, follow this procedure:

1. Start the Content Manager System Administration program and log on.
2. Expand the **LibServerName** item in the tree-view on the left.
3. Expand **Library Server Parameters**.
4. Right-click the item **Configuration** in the expanded view.
5. Select **Explore** from the context menu. A window with the title Library Server Configuration opens.
6. Click the Log and Trace tab.
7. Select all check boxes under Trace level.
8. In the **Trace file name** field, enter the path and the name of your trace file, for example `C:\ICMSERV.LOG`.
9. Run the problematic CommonStore process again.
10. Submit the trace file along with other information to the CommonStore support team.

Chapter 28. Configuring Content Manager for iSeries

This section explains how to set up Content Manager for iSeries archives so that they work with CommonStore for Lotus Domino.

Important

You cannot use Content Manager for iSeries archives and Content Manager archives with the same instance of the CommonStore Server.

Creating index-class definitions

Before you can archive documents, you must perform a number of administrative steps to prepare your Content Manager for iSeries archives.

In Content Manager for iSeries, documents of the same type are grouped in index classes. Documents of the same index class have the same set of descriptive search attributes. CommonStore for Lotus Domino maps Lotus Notes document fields (items) to index attributes. Therefore, you must create an index class for every Notes document type (form).

Furthermore, for every Notes field that you want to be able to search in the archive, you must create an attribute. Every Notes field type is saved as a string value in Content Manager for iSeries.

Important

- Content Manager for iSeries restricts index class names and attribute names to a maximum length of 8 characters. However, CommonStore for Lotus Domino does not use these names, but the descriptions of the index classes and attributes.

  To enable a Content Manager for iSeries archive for archiving with CommonStore for Lotus Domino, each index class supposed to contain archived documents must have an attribute called **OrigFilename**. This attribute is used to store and retrieve the original file names of documents. The attribute must be 40 characters in length and of the type *Filter*.
- Bear the limits of the Content Manager for iSeries archive system in mind when you create attributes:
  - The maximum number of attributes per index class is eight.
  - The maximum length of the attribute description is 20 characters.
  - The maximum length of the attribute value is 40 characters.

If you want to use the *Restore to original database* and *Restore by original user* features, your index classes must additionally contain the following attributes:

CSLDOrigUser

This attribute must be able to hold the longest Lotus Notes user ID that you have in canonical format. The recommendation is 40 characters. User IDs longer than 40 characters cannot be stored in Content Manager for iSeries archives.

CSLDOrigDB
: This attribute holds the replica IDs of the Lotus Notes databases that you archive from. Size it to the minimum of 17 characters.

For every index class you want to archive Notes documents in, you must define a logical archive in the server configuration profile (usually archint.ini) to make it known to CommonStore for Lotus Domino. You can then use the ID of the logical archive in a document mapping. This way, it becomes a valid target for document archiving.

CommonStore for Lotus Domino logs on to the archive with the logical archive ID specified in the server configuration profile. Therefore, this ID must have full rights with regard to the corresponding index class. Create a separate user ID just for CommonStore.

Grant other users access to the index class as required. If the archive is only accessed by CommonStore for Lotus Domino, it is sufficient to grant access to the logical archive ID and to the archive administrator.

Preparing Content Manager for iSeries for folder archiving

To archive and restore complete Lotus Notes folder structures, you must define an index class to hold internal CommonStore information. Use the security features of the archive to make sure that users cannot browse the folders of such an index class. Unfortunately, you cannot create a document mapping to prevent users from accessing the internal information.

This index class must contain the following attributes:

CSLDOrigUser
: Holds the Lotus Notes user names of the users attached to the CommonStore system. The attribute must be able to hold the longest possible Lotus Notes user name in canonical format. The recommendation is 40 characters. You cannot store user names that are longer than 40 characters in Content Manager for iSeries.

CSLDOrigDB
: Holds the replica ID of the Lotus Notes database that contains the original of the archived folder. Size this attribute to the minimum of 17 characters.

CSLDFolderName
: Holds the name or the entire name structure of a Lotus Notes folder or subfolder. Size this attribute to 40 characters. Note that you cannot archive a folder structure whose string representation exceeds 40 characters in length.

CSLDFolderAlias
: Holds aliases of the archived Lotus Notes folder. Size this attribute to 40 characters, too. You can use fewer characters if you know that aliases are not specified in the application. You cannot archive aliases longer than 40 characters.

You can choose an arbitrary name for the index class. You must then create a logical archive ID for the index class in the server configuration profile (usually archint.ini). Finally, you must enter the name of the archive ID in the **Folder Archive ID** field of both, the profile document of your archive task and the profile document of your retrieval task.

Content Manager for iSeries troubleshooting

If you encounter problems with CommonStore and Content Manager for iSeries, follow these steps:

1. Stop the CommonStore Server.
2. Supply the parameter `TRACE ON DEBUG` in the server configuration profile (usually archint.ini).
3. Restart the CommonStore Server.

Chapter 29. Using Content Manager views (subsets)

CommonStore for Lotus Domino supports Content Manager index classes with multiple views (subsets). These are constructs by which the archive administrator can hide attributes and documents of an index class from certain users.

An index class is always connected with at least one view. When you create an index class, it is associated with the standard view. The standard view is a view on all attributes and documents of an index class. In Content Manager, you can create further views, which allows you to limit the scope of what is visible and accessible by the users. Hence the view determines what a user can access, and not the index class itself. The fact that CommonStore supports this feature means that you can extend access restriction by additional views or subsets to CommonStore users.

CommonStore only allows one view on an index class. There is no way of accessing different views of the same index class. The view that is taken depends on the view ID, which is an internal, 10-digit identifier. Any view has such an identifier. CommonStore accesses the view with the lowest ID number. This is normally the standard view.

To restrict CommonStore user access to an additional view or subset, follow these steps:

1. Make sure that the CommonStore users can access only one view. Normally, you have only one CommonStore user per CommonStore Server.
2. Remove the users' access rights for the standard view so that they cannot access it anymore.

This causes the archive agent to try the view with next higher ID, which is the ID of the first subset. Cancelling access to the standard view is the only way of directing user access to another view.

Chapter 30. Configuring Content Manager OnDemand

As IBM's strategic solution for electronic or optical storage of computer output, Content Manager OnDemand has a broad, world-wide installation base. This advanced system makes it possible to store, retrieve, distribute, print, and fax both SAP and non-SAP documents in just seconds — right from a user's workstation.

Tivoli Storage Manager, Content Manager, and Content Manager OnDemand all use a client/server architecture. There are the following kinds of servers:

Library servers
The library server contains indexes and control information. It runs under Windows, AIX, HP-UX, and Sun Solaris. In addition, it supports DB2®. The Windows NT version of the CMOD library server supports Microsoft SQL Server instead of DB2.

Object servers
Object servers contain the archived data and can optionally use Tivoli Storage Manager as a storage manager. Clients run under OS/2, Windows NT, and CICS/ESA®. Clients are available for Netscape Navigator or Microsoft Internet Explorer.

Using the comfortable user interface of Content Manager OnDemand, you can perform simple or advanced queries to find and retrieve documents for processing in a variety of ways. You can view entire documents, portions of documents, or simultaneously view multiple documents — even if the documents are in different formats. You can also annotate documents and copy sections to the clipboard for insertion into a spreadsheet or word processing application. Alternatively, documents can be retrieved using SAP's ArchiveLink viewer or the CommonStore client in conjunction with an arbitrary generic document viewer.

Combined with its robust architecture and suitability for storage of high volumes of data, these features make Content Manager OnDemand a powerful solution for storing and retrieving all kinds of documents. Additionally, Content Manager OnDemand can be used to maintain inactive application data:

Apart from its capabilities as an archive platform for CommonStore, the Content Manager OnDemand server can receive reports and documents from a production system, such as an MVS™ mainframe computer. The system then extracts index fields, such as customer name and account number, from the data and stores them in a relational database to provide fast information.

To prepare Content Manager OnDemand for use with CommonStore, you must perform these steps:
1. Create a CMOD user account.
2. Create a CMOD application group.
3. Create a CMOD application.
4. Create a CMOD folder.

Refer to *IBM Content Manager OnDemand for Multiplatforms: Administrator's Guide* for concepts and detailed information on how to perform the operations requested in this section. The duration and complexity of the configuration process strongly depends on your requirements.

Creating Content Manager OnDemand user accounts

It is recommended that you create a CMOD user account for each logical archive or content repository. The account is used for all processing transactions between the CMOD archive and theLotus Domino content repository. Of course, you can also use the same user account for several or all repositories. Grant the users of a specific content repository appropriate access to all CMOD application groups and folders that you created for the repository.

Creating Content Manager OnDemand application groups

Important

You must set the Expiration Type to 'Segment' because this is no longer a default. Otherwise, CommonStore does not work with CMOD. To set the Expiration Type accordingly, click **Application Group → Storage Management** in CMOD.

In Content Manager OnDemand, documents of the same type are grouped in application groups. Documents in the same application group have the same set of descriptive search attributes. CommonStore maps document properties to index attributes. For every document type (form) that you want to archive, you must create an application group. For every document property that you want to search for in the archive, you must create an attribute.

Table 18 shows the mapping of document field types to OnDemand attribute types:

Table 18. OnDemand attribute types

Property type or form field	OnDemand attribute type	Format
Text	String	
Number	Integer, Small Integer or Decimal	
Date only	Date	%Y-%m-%d
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 application group must contain the following attributes:

CSLDOrigUser
: Type variable character. Maximum length is the length of the longest mail client user name in canonical format in your domain. It is used to store the mail client user name.

CSLDOrigDB
: Type character. Length: 17 characters. It is used to store the replica ID of the database the document was archived from.

In addition, you must define the attributes in Table 19 for every application group storing CommonStore documents:

Table 19. OnDemand attribute names

Attribute Name	Length	Description

Table 19. OnDemand attribute names (continued)

WORKBASKET	128	Name of the virtual workbasket a document has been archived to
FOLDER	60	Used to store the folder name for folder archiving
CONTENT_TYPE	80	The content type as defined in the content-type mapping table. 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 the type *Variable String* and attribute case *Mixed.*

For every index class or application group to archive documents to, you must define a logical archive in the server configuration profile (usually archint.ini)to make it known to CommonStore. You can then use the logical archive ID in a form-to-archive mapping as a valid target to archive documents to.

CommonStore logs in to the archive with the CommonStore archive ID specified in the server configuration profile. Provide this ID with full access rights to the application group. Create a separate user ID just for CommonStore. Whether other archive user IDs must be granted access to the application groups depends on your requirements. If the archive is only accessed by CommonStore, restrict access to the application group to the CommonStore archive ID and the archive administrator.

Creating Content Manager OnDemand applications

It is recommended that you specify a CMOD application for each logical archive or content repository. The application defines the data format for this repository in the CMOD archive. Multiple repositories can, but need not, share the same application.

Note:
A specific logical archive or content repository cannot refer to several applications; thus, you must configure a distinct repository for each data format (application) to be archived in Content Manager OnDemand via CommonStore.

You must assign every application you create to one of the application groups that you have defined before. See "Creating Content Manager OnDemand application groups" on page 126 for more information.

With regard to the document type you want to store in a CMOD archive, look up the appropriate data type, file extension, and compression method in Table 20 on page 128. Assign those parameters in your application definition.

Table 20. Data types, extensions, and compression methods to assign to the supported document types

Document type	Technical document class	Data type	Extension	Compression
Inbound Documents	FAX	TIFF	N/A	disabled
Outbound Documents (OTF)	OTF	User-defined	OTF	LZW/OD77
Outbound Documents (PDF)	PDF	PDF	N/A	LZW/OD77
Print Lists	ALF	User-defined	ALF	LZW/OD77
Archive Data	REO	None	N/A	LZW/OD77
Binary Data	BIN	None	N/A	LZW/OD77

Create additional applications for each document type that you want to archive. For each application that you define, specify *%y%m%d %H%M%S* as the format to be used by the load preprocessor for the DATE_TIME_C and the DATE_TIME_M fields.

Creating Content Manager OnDemand folders

It is recommended that you create a CMOD folder for each logical archive or content repository. A folder is necessary for accessing the archived documents and the meta-data archived in this repository. Multiple repositories can, but do need not share the same folder.

You must map the fields defined in the folder to the attributes of the application group. The names in the folder need not have the same names as the ones in the application group.

In each folder, include at least the search fields listed in Table 21.

Table 21. Search fields to be included in CMOD archive folders

Name	Field type	Database field name	Operators defined
Logical archive/content repository	String	CONT_REP	**Equal** (default), Not Equal, In, Not In
Document identifier	String	DOC_ID	Equal, **Like** (default)
Object identifier/bar code	String	OBJECT_ID	Equal, **Like** (default)
Component identifier	String	COMP_ID	**Equal** (default)
Document protection	String	DOC_PROT	Equal, **Like** (default)
Content length (bytes)	Integer	CONTENT_LENGTH	**Equal** (default), Between, Less Than, L/T or Equal, Greater Than, G/T or Equal
Technical document class/content type	String	CONTENT_TYPE	**Equal** (default), Like
Character set	String	CHARSET	**Equal** (default), Like
Application version	String	VERSION	**Equal** (default), Between, Less Than, L/T or Equal, Greater Than, G/T or Equal
Created	Date/Time	DATE_TIME_C	Equal, **Between** (default), Less Than, L/T or Equal, Greater Than, G/T or Equal
Last modified	Date/Time	DATE_TIME_M	Equal, **Between** (default), Less Than, L/T or Equal, Greater Than, G/T or Equal

It is useful to specify *data* as the default value for the *Component Identifier* search field.

Specify *%m/%d/%Y %H:%M:%S* as the display and default formats for both the *Creation* and the *Last Modified* search fields.

Adapting the server configuration profile

To use CommonStore with Content Manager OnDemand, you must install a CMOD library server on the CommonStore server machine. As a consequence, you must adapt the server configuration profile so that it contains all necessary CMOD-related parameters. You find most of the keywords whose values you must change to reflect your CMOD setup in the archive section of the server configuration profile. The only exception is the ODAGENTS keyword, which controls the number of parallel connections to the CMOD archive.

Provide at least one archive entry per document type that you intend to archive in Content Manager OnDemand. CMOD archive entries are marked by the header STORAGETYPE ONDEMAND and must contain the following keywords:

- ODHOST
- APPGROUP
- APPLICATION
- FOLDER
- ODUSER

You use the ODHOST keyword to specify the name of the CMOD library server. The keywords APPGROUP, APPLICATION, FOLDER, and ODUSER are used likewise to refer to the application group, the application, the folder and the user account of the CMOD archive that you want to enable. See Chapter 30, "Configuring Content Manager OnDemand" on page 125 for more information. Here is a sample archive definition:

```
ARCHIVE  A1
  STORAGETYPE                      ONDEMAND
  ODHOST                           asterix
  APPGROUP                         'Inbound SAP R/3 Documents'
  APPLICATION                      'Inbound SAP R/3 Documents'
  FOLDER                           'SAP R/3 Documents'
  LOGICAL_SYSTEM                   T90CLNT090
  ODUSER                           COMMONSTORE
```

Preparing Content Manager OnDemand for folder archiving

To archive and restore complete folder structures, the CommonStore archive administrator must define a Content Manager OnDemand application group to hold internal CommonStore information. Using the security features of the archive, the administrator must ensure that users cannot access the application group by browsing through the folders. Document mappings for the application group must not exist to ensure that users cannot access the internal information.

The application group must define the following attributes:

CSLDOrigUser
Variable character string, extended alphanumeric. The attribute must be able to hold the longest possible user name in canonical format; the recommendation is 256 characters.

CSLDOrigDB
Variable character string, extended alphanumeric. The attribute holds the replica ID of the Lotus Notes database that contains the original of the archived folder. Size this attribute to 17 characters.

CSLDFolderName
Variable character string, extended alphanumeric. The attribute holds the name (or hierarchical name structure) of the folder or subfolder. Size the attribute to 256 characters or more to archive deeper folder structures.

CSLDFolderAlias
Variable character string, extended alphanumeric. The attribute holds possible aliases of the archived folder. Size the attribute to at least 100 characters. However, it can be shorter if you did not define aliases for the mail client application.

You can choose an arbitrary name for the application group. Then, create a logical archive ID for the application group in the server configuration profile (usually archint.ini). Finally, enter the name of the archive ID in the **Folder Archive ID** field of the profile document used by your archiving or retrieval task.

Important

In Tivoli Storage Manager, you can only archive folders without preserving the folder structure.

Changing the Content Manager OnDemand server port

You can change the port that is used on the Content Manager OnDemand (CMOD) server. This is useful if you run into conflicts when using the default port or if you want to run several instances of the Content Manager OnDemand system on the same computer. In the latter case, you must specify a different port number for each instance and also a different IP address.

You can set the port number by using the ODHOST keyword in the server configuration profile (usually archint.ini). If Content Manager OnDemand cannot find an ODHOST entry in the server configuration profile, it uses the default port with the instance name as the IP address.

If you have several CMOD library servers in a multiple-instance environment or a single library server using a non-default port, you must perform a few configuration steps to enable CommonStore access to these servers. See the appropriate section below.

Accessing CMOD library servers from AIX systems

If CommonStore is installed on an AIX system, you can add entries for various CMOD library servers or a library server with a non-default port to the ars.ini file. Edit the file to create a CMOD instance for each library server, including the port number and the IP address. This enables CommonStore to access these CMOD library servers.

Accessing CMOD library servers from Windows systems

If CommonStore is installed on a Windows system, you can use the server configurator program delivered with Content Manager OnDemand to add CMOD instances for remote AIX, HP-UX, iSeries, Sun Solaris, or z/OS library servers. You can also use it to add CMOD instances for another Windows system.

Content Manager OnDemand – troubleshooting

If you encounter problems with CommonStore and Content Manager OnDemand, try to track down and solve the problem by using the following procedure:

1. Stop the CommonStore Server.
2. Specify TRACE ON DEBUG in the server configuration profile.
3. Enable the message logging facility for all problematic operations by changing the corresponding application group definitions accordingly. Use the *Content Manager OnDemand Administrator* for that purpose.
4. Manually restart the CommonStore Server by using the archpro command from a Command Prompt window.
5. Check the resulting trace file (usually archint.trace) and the CMOD error protocol.

6. Check the CMOD system log on the library server to which the problematic logical archive refers.
7. When in doubt, double-check the CMOD application group and the application definitions.

Part 7. CSLD administration

Chapter 31. Migrating to Version 8

To make your existing CSLD-enabled applications work with CommonStore for Lotus Domino Version 8, you must perform a number of migration steps. Some of these steps are mandatory, others are optional.

Performing mandatory migration steps

To use your existing applications with CommonStore for Lotus Domino Version 8, you must perform the following steps:

- Updating the job database template
- Updating the configuration database template

Updating the job database template

The way in which the CSLD Task scans your databases for jobs has changed. Instead of scanning the All Documents view, the CSLD Task in Version 8 scans two specialized views for jobs. Therefore, you must add these new views to the job database template. To do so, perform one of the following tasks:

- Replace the design of the job database
- Copy the new views to the design of your existing database. This is recommended if your current job database is customized.

Updating the configuration database template

The design of the configuration database was completely reworked. New fields were added to the existing forms, and additional forms were introduced to cater for the new automation features. You must therefore replace the design of your existing configuration database.

Performing optional migration steps

Some of the migration steps are not required. However, you must perform these steps if you want to use some of the new functions in CommonStore for Lotus Domino. To be able to use these functions, you must update the following elements:

- The CreateCSNJobs script library
- The CSLD query form
- The Web functions of CSLD

Updating the CreateCSNJobs script library

The CreateCSNJobs script library was extended by new functions for the classes CSNArchiveJob, CSNRetrieveJob, and CSNDeleteJob. These classes partly reflect the implementation of new features in CommonStore for Lotus Domino.

To use these features, you must replace your existing version of CreateCSNJobs with the new one. Note that you lose the customizations of your existing script library when you do this. You must redo these customizations manually.

Updating the CSLD query form

When you have updated the CreateCSNJobs script library, you must also update the CSLD query form. This is because the CreateCSNJobs script library contains the createCSNQueryJob function, which is called by the CSLD query form when you run the setupDB tool. In version 8 of CommonStore for Lotus Domino, the createCSNQueryJob function was provided with a new interface.

Updating CSLD Web functions

In CommonStore for Lotus Domino Version 8, a Web function and a Web agent were changed to eliminate a problem source in connection with the passing of the HTTP_REFERRER cgi-variable. The names of the job database and the job database server are usually included in the value of this variable. However, some browsers pass an empty HTTP_REFERRER variable to CSLD. For this reason, the hyperlinks (URLs) in hit lists returned by CSLD now include the names of the job database and the server hosting the job database.

To obtain the new hit lists, you must replace the CSNWebFunctions script library and the WebRetrieveSingleDoc agent with their updated versions. You can also modify a custom Web agent by changing the format of the hyperlinks in the source code as follows:

```
http://<DominoServerCN>/<dbName>/WebRetrieveSingleDoc?OpenAgent&archID
=xxx&jobDB=yyy&jobSrv=zzz
```

Chapter 32. Basic setup in Lotus Notes

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.

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. 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.

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.

Including the CSLD user in the access control lists of application databases

Add the CSLD user with *Editor* access to the access control lists (ACLs) of all Lotus Notes databases that make use of CSLD archiving functionality. In doing so, you allow CSLD Tasks to work with the documents in these databases. If you use the CSLD *setupDB* tool to create default forms, the CSLD user must have *Manager* rights in the database to which the setupDB tool is applied.

Chapter 33. Customizing the configuration database

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

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 (for retrieval tasks for example), 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 8, “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.

Created query results as single hit list or multiple result documents
This field is visible only when **Retrieval** is selected. When **single hit list** is selected, the result of a query will be a single hit-list 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.

Task will process jobs from

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 9, “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 e-mail 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, that is, jobs from all databases in the given subdirectories will be processed. Suppose you have a number of subdirectories below the notes data directory, for example, 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, for example, ″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 breakdowns, 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.

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`.

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.

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.

Notify the following CSLD administrators
: 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.

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.

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.

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.

Export directory
This field is used to specify the directory for temporary files. CSLD tasks and the CommonStore 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 TCP/IP port
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 CommonStore configuration file `archint.ini`. Can be any valid, non-reserved TCP/IP port number.

CommonStore host name
The IP name of the CommonStore Server.

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 hit lists. 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**).

Folder archive ID
The ID of a logical archive as defined in the server configuration profile. When you store entire Lotus Notes folder structures in an archive, you only need to specify a logical archive ID. The proper document-type mapping is automatically applied to the documents in the folders. See "Preparing Content Manager for folder archiving" on page 113 for more information.

Trace file directory
Trace files are written to the specified directory.

Log file directory
Log files are written to the specified directory.

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 special field**

  Allows writing the archiving state to a custom field. The state field can be a *Text* item or a *Numbers* item. When archiving succeeds, CSLD writes the value entered in the field *Success value* to the item entered in the field *Status field name*. When archiving fails, CSLD writes the string entered in the field *Error value* to the item entered in field *Status field name*. For example, you can configure CSLD so that it writes the strings *Success* and *Error* to the field *CSLDArchivingState* of the archived document.

  Note: In order to use document rearchiving (see Chapter 12, "Archiving options" on page 45 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.
- **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.

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.

Important
CSLD profiles are read 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:

Define mapping for document form or document field value
Click the radio button to select the appropriate type of mapping.

Notes form name
All documents with this form will be mapped to the archive ID given in the CommonStore 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**.

Notes field name ... when value is ...
Documents containing the specified value in the field defined as the *Notes field name* are given the *CommonStore archive ID* that you specify in the next field.

CommonStore archive ID
Name of a logical archive ID as defined in the server configuration profile (usually archint.ini)

Notes fields to display in hit lists
When users perform queries, they retrieve hit lists. In order to distinguish the different entries of a hit list, 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, for example, **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 hit-list 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 hit list. This does not apply to queries returning hit lists as single result documents, as these documents contain all mapped attributes rather than the representative attributes.

Form used to display returned documents with

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).

Script agents to run custom logic

Enables you to list additional agent programs to be started by CSLD. You employ agents to perform custom tasks. CSLD is able to start agents at the following stages:

- Before an archiving operation
- After an archiving operation
- After a retrieval operation
- After an updating operation
- After deleting documents in the archive. In this case, CSLD can only start an agent if it finds the archive IDs of the deleted documents in the Lotus Notes database.

Specify the names of the agents that you want to use in the following fields, according to the category that they belong to:

Pre-archiving agent

Name of a pre-archiving agent (see Chapter 14, “Agents for processing archived documents” on page 55) that is invoked before a document is archived

Post-archiving agent

Name of a post-archiving agent (see Chapter 14, “Agents for processing archived documents” on page 55) that is invoked after a document has been archived

Post-retrieval agent

Name of a post-retrieval agent (see Chapter 14, “Agents for processing archived documents” on page 55) that is invoked after a document has been retrieved

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 CommonStore 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 logical archive ID as defined in the server configuration profile (usually archint.ini). Documents fulfilling the special mapping criteria are archived in this logical archive.

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 is 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 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 appears 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 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 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 fields or RTF fields. When a multi-valued field is used, CSLD interprets only the first value.
- The CommonStore archive ID defined in special mappings always overwrites the CommonStore 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 in more complex situations. Imagine that you want to store all classical audio CDs of Debussy and Ravel in an 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 field-value mappings

Instead of making the target archive of a document dependent on the underlying form (form mapping), you can alternatively make it dependent on the value of a certain text field. This type of mapping is called field-value mapping. It is very useful under the following circumstances:

- You want to archive documents in different archives, according to their properties (fields).
- The documents are based on the same form, but they contain fields that differentiate them from another. These fields are defined (computed) by subforms. The value of a certain form field determines the subform that is used.
- The fields defined in the subforms contain the essential information so that you want these fields to become archive attributes.

Do not confuse field-value mappings with special mappings as described in "Defining special mappings" on page 146. Special mappings also determine the target archive on the basis of a field value, but they do not allow you to have a different set of attributes in each archive. Field-value mappings are not subject to this limitation.

Example:

Suppose you define a form for a Lotus Notes application. This basic form contains a number of fields, such as *CreationDate*, *DocVersion*, *ModificationHistory*, and so on. In addition to these basic fields, there is a field called *DocType*. The value of this field determines the subform that is used in connection with the basic form. The subform defines additional fields, which describe the documents more accurately.

If the target archive was determined by the form, you could only map the fields of the basic form to archive attributes. Using special mappings, you can define different destinations, according to the value of the *DocType* field. However, the documents would still have the same attributes, no matter what archive they are in. This is because the fields containing the important information are defined by the subforms, which you cannot include in a field-to-attribute mapping when you define a form mapping. As a result, attribute-based queries in the archive would be limited to attributes with non-essential information. With field-value mappings, you can overcome this limitation.

To define a field-value mapping, you must select the **Document field value** radio button when you create a mapping document. You can then enter the field name and the value in the **Notes field name** and **when value is** fields. See "Defining document mappings" on page 144 for more information.

Important

In contrast to special mappings, document-field mappings are limited to text fields.

Defining content-type mappings

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 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.

Important

- Content-type mappings are required for Content Manager archives only. However, under certain circumstances, they might prove useful for Content Manager OnDemand archives, too. For more information about content-type mapping, see "Content types and original file names" on page 21.
- 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.
- 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.

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 "Content types and original file names" on page 21.

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.

Using the automation features

The following sections explain how to set up the automation features of CommonStore for Lotus Domino.

Defining crawler tasks

Before you can define and use policies, you must create create a profile document for a CSLD Task, which contains a number of settings for the CSLD crawler. You enter these settings in a Lotus Notes document derived from the CSC Task form. To create such a document in the configuration database, follow these steps:

1. Change to the **All Automatic Archiving Tasks** view.
2. Click the **New Task** button. The Automatic Archiving Tasks dialog opens.
3. In the Automatic Archiving Tasks dialog, provide the appropriate settings. The dialog offers the following fields and controls:

   Task name

   The name of the crawler task that you want to create. The task name is passed to the crawler program csc.exe by means of the -t parameter. Do not use blanks in this task name.

Working database set
The name of a list of databases. The working database set is a Lotus Notes document containing the names of databases that you want the crawler program to work on. See "Defining working database sets" on page 152 for more information.

Job database
The name of the job database supposed to process the jobs created by the crawler program.

Server The name of the server on which the job database resides.

Scheduling
This group of controls allows you to specify settings for the activation of the crawler task. See the following list:

Weekly
By selecting **Weekly**, you configure the task to run on a certain day every week. To complete this setting, you must specify the day in the week and the time when you want the crawler to start.

Daily By selecting **Daily**, you configure the task to run every day. To complete this setting, you must specify the time when you want the crawler to start.

Hourly
By selecting **Hourly**, you configure the task to run at hourly intervals. To complete this setting, you must specify an interval period in hours.

Limit run to
Allows you to limit the runtime of the crawler program. The programs stops after the specified number of hours. Documents remain unprocessed if the crawler program cannot finish its work on the specified set of databases during this time.

Shutdown port
In this field, enter the number of the TCP/IP port that the crawler task uses to receive shutdown request.

Send error notification to administrator
Check this box if you want the task to notify the CSLD administrator in case of a severe error.

Enable tracing
If you check this box, you can enter the name (including the full path) and the maximum size of a trace file in the adjacent fields. The crawler task then writes trace information for debugging purposes to this file.

4. Save and exit the profile document.

Defining policies

Apart from a few other parameters defined therein, policies are groupings of selection criteria for policy-driven processes. By using policies, the crawler program selects the documents to be archived or deleted according to the schedule that you set up in the crawler task. The crawler program is able to select Lotus Notes documents for archiving on the basis of their age and their size. For deletion, the crawler program selects documents on the basis of selection formulas, which you also define in a policy.

Polices are Lotus Notes documents derived from the CSCPolicy form. To create a policy in the configuration database, follow these steps:

1. Change to the **All Automatic Archiving Tasks** view.
2. Click the **New Policy** button. The policy form opens.
3. Provide the appropriate settings. The form offers the following fields and controls:

   Policy name
   The name of the policy that you want to create. Make sure that this name is unique.

   Policy type
   The CSLD crawler task distinguishes between archiving and deletion policies. Select the appropriate policy type. For more information, see "Policies" on page 27.

   Select by age
   When you check this box to let the crawler program select documents by age, you enable further controls by which you can refine your selection criteria.

   Documents can be selected by age according to a relative date or an absolute date. When you specify a relative date, you define a limit in days. For example, you might set a relative date of 5 days, which causes the crawler program to select documents for archiving that are five or more days old.

   An absolute date, on the other hand, is similar to a deadline. For example, if you specify the 15th of March, 2002, the crawler program selects all documents for archiving that were created or last modified before or on that day.

   In addition, you can set the reference date, which is the date that serves as the reference point for the relative or absolute date. The reference date can be the *creation date* or the date of the last modification (*last modified*).

   Select by document size
   When you check this box to let the crawler select documents by size, you can enter a limit with respect to the size of the documents or the database.

   When you specify a document size-limit, you enter a number of kilobytes (KB). For example, if you enter `50`, the crawler program selects all documents for archiving whose size is 50 KB or more.

   When you specify a database size-limit, you enter a number of megabytes (MB). For example, if you enter `100`, the crawler program selects all documents in a database for archiving if the total size of that database exceeds 100 MB.

   Selection formula
   This field allows you to enter a selection formula in Lotus Notes formula language for deletion policies. Based on this formula, the crawler program selects documents in the working database set for deletion.
4. Save and exit the policy.

Defining working database sets

By defining a working database set, you create a Lotus Notes document that contains the name of a database or data directory to apply policy-driven processes to. If you specify a data directory, you assign a crawler task and at least one policy to all the databases in this directory. The crawler program only works on the databases that are defined in the working database set.

A working database set is derived from the *CSC Database* form. To create a working database set from the configuration database, follow these steps:

1. Change to the **All Automatic Databases** view.
2. Click the **New Database** button. A Lotus Notes form opens.
3. Enter the appropriate settings. The form offers the following fields:

   Name The name of the working database set

   Policy/Policies
   The name or names of the policies that you want to assign to the database or data directory in the working database set

   Database or data directory on server
   The name of the database or data directory that you want to assign policies to. Add the full path of the database or database directory on the Lotus Domino server to your specification. Enter the server name in abbreviated format.

   Exclude these databases
   Allows you to list databases to be excluded from policy-driven processing if you specified a data directory in the **Database or data directory on server** field
4. Save and exit the working database set.

Chapter 34. Command-line administration

Starting and stopping CSLD Tasks

CSLD archiving and retrieval tasks are both started by launching the csld.exe program. You must start the csld.exe program 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 139 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. This parameter is only considered if the CSLD Task runs on a Windows system. If it runs on an AIX system, the parameter is ignored.

-f serverpasswd
Specify this parameter to provide a password for the currently active Notes user ID. 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 reads it the next time it starts, instead of bringing up a password prompt. However, on an AIX system, this only works if you use Lotus Notes R5 or above.

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 Lotus Notes run time in an unpredictable state.

Starting CSLD Tasks without typing in Notes passwords

In CommonStore for Lotus Domino 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 8, 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 gap as the passwords in the file are protected by a sophisticated encryption algorithm.

To use this feature, add the following line to your Lotus Notes initialization file:

```
EXTMGR_ADDINS=CSLDExtPwd.dll
```

Important

- It is recommended that you use this setting in an additional .ini file and pass the name of this file to the csld.exe program by specifying the appropriate command-line parameter.

  The reason for this is that Extension Manager add-ins are common to all Lotus Notes client programs. If you add the EXTMGR_ADDINS parameter to the notes.ini file, the CSLD dynamic link library (.dll file) for the Extension Manager might also suppress the password prompt of other applications in the Lotus Notes client workspace.
- On AIX, you can only use this parameter in conjunction with Lotus Notes R5 or higher.

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.

Logging and tracing

CSLD Tasks log all errors and events like startup or shutdown in the *profilename*.log file, where *profilename* is the name of the task profile. The error logs include date and time, and are identical to the error messages written to the jobs that were carried out while the error occurred. Check the size of this file from time to time. 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 the *profilename*.trace file.

The trace and log files are placed in the directories that you specified for the CSLD Task profile in the configuration database. However, CSLD does part of the tracing work before it reads the configuration data. Hence it does not know the trace directory at this point in time. Therefore, CSLD places this part of the trace information in the directory that you specified by setting the INSTANCEPATH environment variable.

To make CSLD write both parts of the trace information to the same directory, let the INSTANCEPATH variable and you task profile point to the same location or leave the trace directory field in the configuration database empty.

Chapter 35. 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 is sent to the CSLD administrator.

Chapter 36. 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 rasterized
HANDLE hDB:	C handle to Notes DB containing above note
unsigned long ulRasterFormat:	Constant defining the rasterizing 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 that determines what to rasterize. CSLDRaster.h defines symbols for allowed values. Possible values are: RASTER_NOTE_APPEND_ATTACHMENT - rasterize both, note and attachments, then append the attachments behind the note. RASTER_NOTE_EMBED_ATTACHMENT - rasterize both, note and attachments, embed attachments at the position they were attached at. RASTER_NOTE_ATTACHMENTS_ONLY - rasterize only the attachments of the note, but not the note itself.
unsigned long addtlFlags:	Additional set of Ored together flags determining how to rasterize 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 space characters 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 rasterizing. 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 rasterizing 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:	info@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 37. 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 (for example, `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 38. 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 10, "Security" on page 33). This way, no other user can copy information from another job and thus access documents illegally.

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 Table 22 on page 166.

Table 22. 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 rasterizing 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″.
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.

Table 22. General fields (continued)

Field name	Data type	Usage
jobSigner	Text	Jobs are signed with the electronic signature of the requester. When CSLD modifies the job documents, for example by changing the job state or by logging errors, the signature is changed to that of the CSLD user. The jobSigner field is used to store the signature of the original requester. CSLD uses this field internally. The job requester does not need to provide a value.
reqStart	Date	CSLD uses this field to store timestamps that mark the beginning of a job process. CSLD uses this field internally. The job requester does not need to provide a value.
reqStop	Date	CSLD uses this field to store timestamps that mark the end of a job process. CSLD uses this field internally. The job requester does not need to provide a value.
reqDuration	Number	CSLD uses this field to store the duration times of job processes in seconds and fractions of seconds. CSLD uses this field internally. The job requester does not need to provide a value.

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, that is, in a bytestream format that can, when retrieved, be restored to that exact Notes document
- The archiving in Microsoft Rich Text Format, that is, rasterizing the given Notes document(s) to an RTF image that can be restored as a file attachment of the type RTF
- The archiving in ASCII format, that is, 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 determine whether the following actions are taken:
- Deletion of the original document from the Notes database after it has been successfully archived (**deleteOriginal**),
- Removal of archived attachments from their originating documents (**deleteAttachments**)

- Creation of a document stub from an archived document (**createDocumentStub**)
- Check of the archive integrity in connection with re-archiving requests (**checkArchIntegrity**)
- Deletion of the job document itself after successful job completion (**deleteJob**)

Archiving job fields

If you set the **requestType** field to CSN_ARCHIVE_RTF_FORMAT, CSN_ARCHIVE_ASCII_FORMAT or CSN_ARCHIVE_TIFF_FORMAT, in addition to the general fields, you must also define the fields in Table 23.

Table 23. 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 rasterized. CSLD defines a set of constants, each of which stands for a particular rasterizing option. Available values are: 1. CSN_RASTER_NOTE_EMBED_ATTACHMENTS (rasterize the document and its attachments. Attachments will be rasterized at the position they were attached.) 2. CSN_RASTER_NOTE_APPEND_ATTACHMENTS (rasterize the document and its attachments. Attachments will be rasterized at the end of the document.) 3. CSN_RASTER_NOTE_ATTACHMENTS_ONLY (only the document's attachments will be rasterized. All attachments will be rasterized 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″.

Table 23. Required fields for archiving requests (continued)

Field name	Data type	Usage
createDocumentStub	Text	Optional field. Determines if CSLD creates a so-called stub in the Lotus Notes database after the content was archived successfully. The stub can be seen as the shell of the former document. If you set the field value to *yes*, CSLD creates a stub for each archived document, containing only the first RichText item of the original document. In this RichText item, it inserts a replacement text that you can specify in the configuration database. The default value of this field is *no*, which means that CSLD does not create stubs. When archiving encrypted documents, this parameter is ignored if the CSLD user ID lacks the proper decryption key.
checkArchIntegrity	Text	Optional field. Determines how CSLD handles re-archiving requests. Possible values are *yes* and *no*. The default value is *no*. For more information, see "Controlling document rearchiving" on page 46.
workbasketName	Text	If this field is given and non-empty, the document will be archived to the workbasket with the given name (Content Manager and Content Manager OnDemand 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, that is, 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 you set the **requestType** field to CSN_UPDATE_INDEX, in addition to the general fields, you must also define the fields in Table 24 on page 170.

Table 24. 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 to 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, you can also state the UNID of a document containing the link to the archived document. 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 you set the **requestType** field to CSN_DELETE, in addition to the general fields, you must also define the fields in Table 25.

Table 25. 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. Enter the name in abbreviated format, otherwise CSLD does not process the job.
deletionIDs	Text, multi-valued	This field contains the archive document IDs of all the documents to be deleted
docUNID	Text	Contains the UNIDs of Lotus Notes documents that refer to the archived documents you want to delete (**CSNDArchiveID**). If set, the CSLD process deleting the archived document specified in the job also deletes the reference to this document from the Lotus Notes document pointing to it. This parameter is optional.
deleteSourceDoc	Text	Optional field determining if the Lotus Notes document containing the reference to the archived document is also deleted during the deletion process. Possible values are *yes* and *no*. The default value is *no*. CSLD considers this field only if the **docUNID** field contains values.

CSNDSpecificJob uses subform **DeleteByID** to display deletion requests. The additional fields defined by this subform are described in Table 26.

Table 26. 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 (that is, taken from the CSNDArchiveID field of an archived Notes document or from a hit list), 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 hit list or as 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 hit list will return the folder content in a second hit list (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, for example, 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 Lotus Notes.

In addition to the usual locations for single document retrieval, you can specify a Lotus Notes document in a Lotus Notes database as the target. This causes CSLD to place the retrieved content as a file attachment in an RTF field of this document or as a stand-alone attachment at the bottom of the document. The descriptive information for this archive document will be skipped.

Furthermore, you can specify the native archive server, archive container, and the document archive ID (ID provided by the archive system) of an archived document in a retrieval request. This way, you can retrieve documents without CSLD document archive IDs, for example documents that were archived by an application other than CSLD.

Single document retrieval job fields

If you set the **requestType** field to CSN_REQUEST_BY_ID, in addition to the general fields, you must also define the fields in Table 27.

Table 27. 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, that is, without their original UNID. If not specified or set to 0, CSLD will restore natively archived documents exactly as they were before archiving, that is, 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, for example, an archive folder is parent to all documents residing in it.
CSNHandlePlaceholders	Text	Optional field whose value determines if CSLD removes the placeholders of archived attachments when the attachments are retrieved. CSLD differentiates between the values *zero* and *non-zero* (any other value). The default value is zero (0). When set to zero, CSLD does not remove the placeholders. Instead, it hides them so that they are invisible for the time that the attachments stay in the document. When you remove the attachments again, the placeholders reappear. If you set the field to a value other than zero, the placeholders are removed when CSLD retrieves the attachments.

Table 27. Required fields for request type CSN_REQUEST_BY_ID (continued)

Field name	Data type	Usage
nativeArchiveServer	Text	Optional field. It includes the names of archive servers from which to retrieve archived content. If you enter any value in this field, you must also specify values in the **nativeArchiveContainer** and **nativeArchiveDocID** fields. **Note:** If your backend archive is Content Manager, enter the names of the library servers.
nativeArchiveContainer	Text	Optional field. It includes the names of archive containers from which to retrieve archived content. If you enter any value in this field, you must also specify values in the **nativeArchiveServer** and **nativeArchiveDocID** fields. **Note:** If your backend archive is Content Manager, enter the names of index classes or item types.
nativeArchiveDocID	Text	Optional field. It includes the archive document IDs of the documents that you want to retrieve (IDs provided by the archive system). If you enter any value in this field, you must also specify values in the **nativeArchiveServer** and **nativeArchiveContainer** fields. **Note:** If your backend archive is Content Manager, enter the item IDs of the documents.

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 von 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, for example, `"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,″ that is, 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, you must define the fields in Table 28 for a query job.

Table 28. 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, that is, without their original UNID. If not specified or set to 0, CSLD will restore natively archived documents exactly as they were before archiving, that is, 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 hit list 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 hit-list document.

CSNQueryForm

CSLD provides a default form named **CSNQueryForm** that can be used to create query requests for a particular document type (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.

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 173). 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 17), 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, that is, 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 173), those documents will be returned without their content. Note also that the maximum number of hits that a single hit list can display is 250.

The administrator setting up a CSLD system has two choices as to how these results will be displayed. From the profile (see Chapter 33, "Customizing the configuration database" on page 139 for details), the administrator can select either a single hit-list 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 hit-list document

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

Hit-list documents contain the following information:

- The requester and request timestamp of the query request that resulted in this hit list
- The document type (that is, the form name) of all the hits contained in the document
- The hits themselves contained in an RTF item
- 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 hit lists 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 hit-list 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 hit-list 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 hit-list 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, that is, the number of archive document IDs contained in this hit list

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 hit list. This hit list 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 hit-list form uses the content of this field to retrieve all documents in the hit list.

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 hit list 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 (that is, 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 165) 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, for example, ″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 you set the **requestType** field to CSN_RESTORE_FOLDER, in addition to the general fields, you must also define the fields in Table 29.

Table 29. Fields to define when setting the **requestType** *field to CSN_RESTORE_FOLDER*

Field name	Data type	Usage
folderArchID	Text	Document archive ID that was given to the archived folder when it was archived by CSLD. Specify this field or **folderName**.
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 by a retrieve job of the request type CSN_LIST_WORKBASKET. In addition to the general fields, you must set the fields in Table 30.

Table 30. Fields to define when you want to list documents in a workbasket

Field name	Data type	Usage
targetDB	Text	Name of the database to write the hit list (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 hit list (or multiple result documents) will be added to the folder with the given name. Otherwise the hit-list 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.

Table 30. Fields to define when you want to list documents in a workbasket (continued)

parentDocUNID	Text	Optional. The hit-list 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 39. 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 38, "Creating job documents" on page 165 for details.

The setupDB tool

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:

- A query form to enter the parametric search that is passed to the underlying archive
- A form to display archived content with

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

The *setupDB* tool 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 hit-list forms are independent of a certain Notes form, setupDB does not create a default for a hit list. However, CSLD also provides a default hit-list 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. The setupDB tool 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. The setupDB tool creates 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 to display search results in, and so on.

Chapter 40. 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 2 on page 186 shows the hierarchy of the CSLD script classes. The arrows indicate inheritance. Lines between the boxes indicate the use of a class by another class.

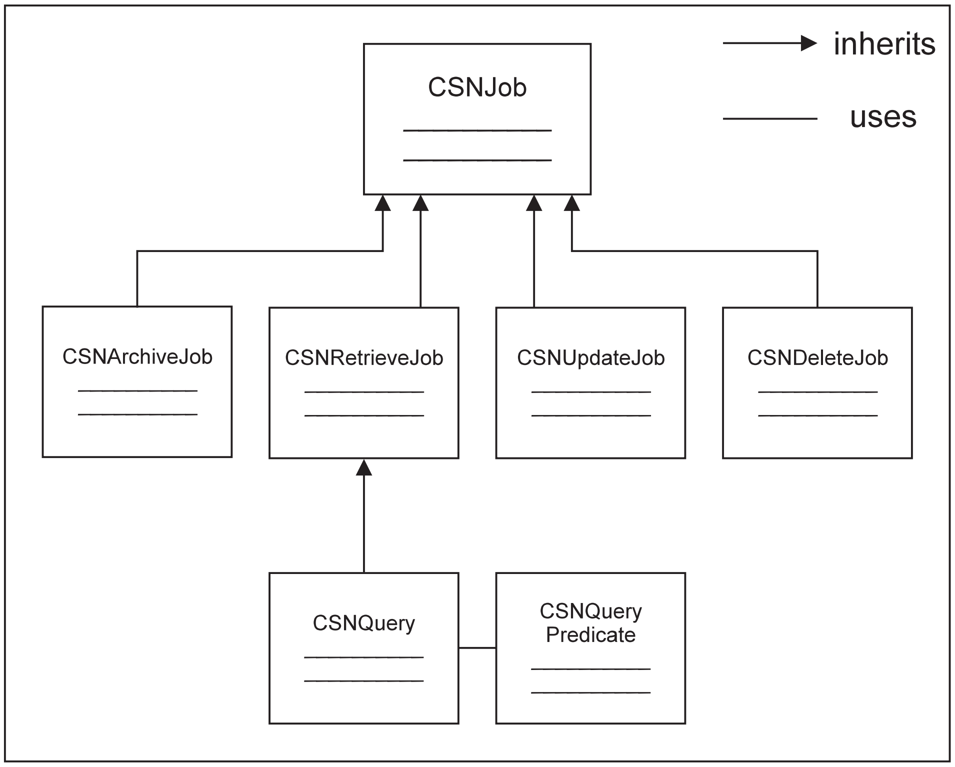

Figure 2. Class hierarchy of the CSLD script class

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, that is, 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.

Archiving options for rasterizing

There are symbolic values defined for the available rasterizing 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_ATTACHMENTS% = 101
Const CSN_RASTER_NOTE_ATTACHMENTS_ONLY% = 102
```

CSN_RASTER_NOTE_EMBED_ATTACHMENTS 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_SUCCESS = 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 NotesSession’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 rasterized. 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 Property Set CreateDocumentStub As Variant`

`Public Property Get IsCreateDocumentStub As Variant`
Used to set/get the flag that tells the system whether or not to transform a document into a document stub. When a stub is created, all attachments and RichText items are deleted from the original Lotus Notes document after it has been successfully archived. You can use the stub as a handle to retrieve the archived document.

`Public Property Set CheckArchiveIntegrity As Variant`

`Public Property Get IsCheckArchiveIntegrity As Variant`
Used to set/get the flag that determines how the system handles rearchiving requests. If the value is `TRUE`, the system allows the rearchiving of documents with the same request type only once, and checks the validity of rearchiving requests. For more information, see “Controlling document rearchiving” on page 46.

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, that is, 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 193.

`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 193.

`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, for example, 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 (for example, "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 (that is, without their UNID), while setting it to FALSE will restore them including their UNID.

`Public Property Set NativeArchiveServer As String`

`Public Property Get NativeArchiveServer As String`
Used to set/get the name of the archive server from which to retrieve an archived document. You need this property if you want to retrieve documents that do not have a CSLD document archive ID.

When you set this property, you must also set `Public Property Set NativeArchiveContainer As String` and `Public Property Set NativeArchiveDocID As String`.

If your archive system Content Manager, use the name of the library server as the value of `Public Property Set NativeArchiveServer As String`.

`Public Property Set NativeArchiveContainer As String`

`Public Property Get NativeArchiveContainer As String`
Used to set/get the name of the archive container from which to retrieve

an archived document. You need this property if you want to retrieve documents that do not have a CSLD document archive ID.

When you set this property, you must also set `Public Property Set NativeArchiveServer As String` and `Public Property Set NativeArchiveDocID As String`.

If your archive system Content Manager, use the name of the appropriate index class or item type as the value of `Public Property Set NativeArchiveContainer As String`.

`Public Property Set NativeArchiveDocID As String`

`Public Property Get NativeArchiveDocID As String`
Used to set/get a document's ID in the archive (ID provided by the archive system). You need this property if you want to retrieve documents that do not have a CSLD document archive ID.

When you set this property, you must also set `Public Property Set NativeArchiveServer As String` and `Public Property Set NativeContainer As String`.

If your archive system Content Manager, use the item ID as the value of `Public Property Set NativeArchiveDocID As String`.

`Public Property Set RemovePlaceholders As Variant`

`Public Property Get IsRemovePlaceholders As Variant`
Used to set/get the flag that tells the system whether or not to remove the placeholders in Lotus Notes documents when their content has been successfully retrieved from an archive. If you set this property to `TRUE`, the placeholders are removed. If you set it to `FALSE`, the placeholders are just hidden from view for the time that the attachments remain in the documents. The default value is `FALSE`.

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 (that is, 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 Property Set DocUNID As Variant`
Used to set the UNIDs of documents containing references to other documents that you want to delete from the archive. After the documents were deleted, the references are removed from the documents with the specified UNIDs.

`Public Property Set DeleteSourceDoc As Variant`
: Used to set the flag that tells the system whether or not to additionally delete Lotus Notes documents if these documents contain references to content that you want to delete from the archive. This property is not considered unless you specify UNIDs for `Public Property Set DocUNID As Variant`.

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 hit-list 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 197).

`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 von 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. Keywords in the server configuration profile

The keywords allow you to adapt server configuration profiles to your needs. Each keyword is explained in detail; the following sections describe the function of each keyword, the context in which to use it, and the parameters or values you can specify. The keywords are in alphabetical order.

General remarks

To avoid unnecessary errors, read the following general remarks carefully *before* you start modifying a server configuration profile.

Important::

1. Do not use the names of keywords as values. It is especially important that you do *not* use VI as an archive ID by setting the ARCHIVE keyword to this value.
2. The # character is the comment symbol. When a line in the server configuration profile starts with a #, CommonStore does not process it.
3. Some keywords are now obsolete. Therefore, they are no longer documented. The CommonStore Server still accepts these keywords, but issues a warning if it detects one of them in a server configuration profile. The following keywords are obsolete:
   - ARCHAGENT
   - ARCHAGENTOD
   - ARCHAGENTVI
   - ARCHREG
   - ARCHWIN
   - DISPATCHER

   Use the BINPATH keyword to replace them. BINPATH specifies the directory in which the binary files of the CommonStore Server reside. For more information, see the entry for BINPATH on page 205.

ACCESS_CTRL YES|NO
: Additional keyword for the ARCHIVE statement. Causes CommonStore to call the CSExit.DLL file to replace the registered Exchange 2000 user with another Content Manager user. The CSExit.DLL file must be provided by the companies themselves.

  DEFAULT
  : NO

  EXAMPLE:
  ```
  ACCESS_CTRL YES
  ```

ADSMAGENTS *number*
: For use with Tivoli Storage Manager only. Using this TSM-specific keyword, you can specify the total number of parallel Tivoli Storage Manager client sessions (name: *archagent*) that the CommonStore Server establishes. For a direct archiving or retrieval 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. For performance reasons, it is recommended that you use as many agents as there are tape drives available. The default value is 0.

EXAMPLE:

```
ADSMAGENTS 3
```

ADSMNODE *nodename*
For use with Tivoli Storage Manager only. Using this TSM-specific keyword, you can specify the node name for the Tivoli Storage Manager log-in procedure. Do not add this keyword to the ARCHIVE statement if you use the PASSWORD GENERATE option of Tivoli Storage Manager.

APPGROUP *group*
For use with Content Manager OnDemand only. Adding this CMOD-specific keyword to the ARCHIVE statement, you can specify the name of the Content Manager OnDemand application group. Enclose application group names containing spaces in single quotation marks.

EXAMPLE:

```
APPGROUP 'CSX Mail Demo'
```

APPLICATION *app*
For use with Content Manager OnDemand only. Adding this CMOD-specific keyword to the ARCHIVE statement, you can specify the name of the CMOD application. Enclose application group names containing spaces in single quotation marks.

EXAMPLE:

```
APPLICATION 'CSX Mail Demo'
```

ARCHIVE *archive_ID*
The value *archive_ID* specifies the logical archive ID, for example A1. The archive ID must be unique. CommonStore uses it to identify the requested archive. All keywords required to access this archive are combined in the so-called ARCHIVE statement. The STORAGETYPE keyword 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. See the following example:

```
ARCHIVE A2
  STORAGETYPE VI
  LIBSERVER LIBSERV2
  INDEX_CLASS TestClass
  VIUSER FRNUSER
```

Important

- You must not use VI as an archive ID. In general, keywords of the server configuration profile must not be used as names.
- It is recommended that you save these settings and do not change them after you have completed the set-up; this is because any retrieval operation depends on them.
- The specification `ARCHIVE DEFAULT` is no longer valid. If you cannot access an archive, check if this statement is in the server configuration profile (usually archini.ini). If the answer is yes, replace `DEFAULT` with the logical archive ID.

ARCHIVETYPE GENERIC | SAP
This is an additional keyword for the ARCHIVE statement. If you set it to SAP, CommonStore verifies if certain special attributes exist, which are required for SAP archives.

ARCHPRO_PORT *port*

Using this keyword, you can specify the TCP/IP registration port used by all CommonStore clients and all CommonStore DLLs for connecting to the CommonStore Server. ARCHPRO_PORT replaces ARCH*WIN*_PORT.

This port must correspond to the settings in the client configuration profile (name: CSClient.ini) on Windows PCs. This fixed port is also used by the archstop and the archbc programs.

EXAMPLE:

```
ARCHPRO_PORT 5500
```

Notes

The port number you specify must be greater than or equal to 5000.

This is the fixed registration port of all Windows PCs. If more than one CommonStore server is installed on a single machine, each instance requires a different setting of the ARCHPRO_PORT keyword.

ARCHWINS *number*

Using this keyword, you can specify the total number of parallel sessions (name: *archwin*) which the CommonStore Server establishes for the CommonStore Client and the CommonStore DLL. The default is **1**.

EXAMPLE:

```
ARCHWINS 3
```

Note

If you do not use CommonStore Client or the CommonStore DLL, specify `0`.

ARCHWIN_PORT *portnumber*

This keyword is obsolete. Use ARCHPRO_PORT instead. ARCHPRO_PORT has exactly the same effect and is used in the same manner. For more information, see ARCHPRO_PORT.

BINPATH *path*

Specifies the complete path to the binary files of the CommonStore Server.

EXAMPLE:

```
BINPATH C:\Program Files\ibm\csld\bin
```

Attention

Do not rename any binary files. In addition, make sure that they reside in a single directory.

CHECK_ARCHIVE_SERVER ON|OFF

Specifies whether the CommonStore Server starts if 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, CommonStore refuses to start. When set to OFF, CommonStore displays a warning if an archive is not available; nevertheless, it continues to start up. The default value is ON.

CMAGENTS

For use with Content Manager Version 8 only. The keyword allows you to specify the number of parallel Content Manager sessions. This is the number of agents that CommonStore starts simultaneously to access Content Manager Version 8. The default value is 0, which means that no agent is configured. To access a Content Manager Version 8 archive, you must specify at least one agent.

EXAMPLE:

```
CMAGENTS 3
```

CMUSER

For use with Content Manager Version 8 only. The keyword allows you to specify a user ID to log on to Content Manager Version 8. This way, CommonStore logs on to Content Manager Version 8 automatically, that is, at the time when you start the CommonStore Server.

COMMENT

This is an additional keyword for the ARCHIVE statement. Using this general 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 CommonStore Server to store all variable parameters such as passwords, user names, and the current version number. The value filename specifies the full path and the name of the file. This keyword is required.

EXAMPLE:

```
      CONFIG_FILE c:\ibm\csld\archint.cfg
```

Important

The configuration file is encrypted.

END

Using this keyword, you can specify the end of the parameter definitions. When END is encountered, the CommonStore Server stops searching the configuration profile for keywords.

ERRORLOG_FILE *filename*

Specifies a directory and a file in which all errors occurring during a CommonStore operation are 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, and so on).
3. Return code, extended return code if present, reason code if present.
4. Error description obtained from the application programming interface or generated by CommonStore.

The error log file grows without size limitation.

DEFAULT
Value of INSTANCEPATH + "csserror.log"

EXAMPLE
```
ERRORLOG_FILE   C:\Program
Files\IBM\CSLD\server\instance01\log\csserror.log
```

FOLDER *folder*
Using this CMOD-specific keyword, you can specify the name of the Content Manager OnDemand folder. The term *folder* must be the name of a folder which references the CMOD application group specified using the keyword APPGROUP. Folder names containing spaces must be enclosed in single quotation marks.

EXAMPLE:
```
FOLDER 'SAP R/3 Documents'
```

Note
This keyword refers to the corresponding **ARCHIVE** statement and is used only for Content Manager OnDemand archives.

FORMAT *format*
Using this Content Manager-specific keyword, you can specify the Content Manager data format to be used for archived documents. The default is **TIFF6**.

EXAMPLE:
```
FORMAT SAPALF
```

Note
This keyword refers to the corresponding **ARCHIVE** statement and is used only for VisualInfo and Content Manager archives.

INDEX_CLASS *index_class_name*
For use with VisualInfo or Content Manager only. Specifies the index class that the CommonStore server uses to archive documents content. This index class must be defined on the corresponding Content Manager or VisualInfo library server.

EXAMPLE:
```
INDEX_CLASS TestClass1
```

INDEX_CLASS_COMP *index_class_name*
Using this Content Manager-specific keyword, you can specify the Content Manager index class which the CommonStore Server uses to archive the single components of a document. This index class represents the final index class where the components of an archived document are stored. If this keyword is omitted, a default index class named <INDEXCLASS>comp will be used. This index class must be defined on the corresponding Content Manager library server.

Note
This keyword refers to the corresponding ARCHIVE statement and is used only for Content Manager archives.

INSTANCEPATH *path*

Specifies the directory in which the instance-related files (profile, configuration file, ...) are located. Create a sub-directory for each CommonStore server instance that you use. Set the INSTANCEPATH keyword for each instance, that is, for each instance specify a path that points to the related subdirectory. All instance-related files are maintained in these directories.

DEFAULT

Value of the BINPATH keyword.

EXAMPLE

```
INSTANCEPATH   c:\csldinst\inst1
```

ITEM_TYPE

For use with Content Manager Version 8 only. Specifies an item type (formerly: index class) that the CommonStore Server archives to. You must also define this item type on the Content Manager library server.

EXAMPLE:

```
ITEM_TYPE TestType1
```

LOG ON|OFF

If you set this keyword to ON, CommonStore creates a log file containing information about all archived and retrieved data for each day.

The log files are generated in the following format:

```
aiyyyymmdd.log
```

where

- `yyyy` = the year,
- `mm` = the month, and
- `dd` = the day.

LOGPATH *path*

The value path defines the complete path to the log file. The log files' file names are generated automatically.

EXAMPLE:

```
LOGPATH C:\Program Files\ibm\csld\bin
```

MAILSRV *host[:port]*

Using this keyword, you can specify the host and port of an SMTP e-mail server to which the e-mails created in the CommonStore browser view are sent. By default, no e-mail server is predefined.

EXAMPLE:

```
MAILSRV mailserv:47110
```

MGMT_CLASS *management_class*

For use with Tivoli Storage Manager only. You must specify this keyword. It defines the TSM management class that the CommonStore Server uses to archive documents. The parameter string can consist of up to 16 characters.

Attention:
Keep in mind that Tivoli Storage Manager automatically deletes all files as soon as the expiration period is reached. Therefore, check the Tivoli Storage Manager expiration date (specified in the management class in the Tivoli Storage Manager archive storage pools).

MULTIPART ON|OFF
Applies to Content Manager and VisualInfo archives only. Specifies if documents are stored on the Content Manager server in multiple parts or not. It is recommended that you store the documents in one part because documents in multiple parts can cause problems if they are displayed in the Content Manager viewer. Setting the value of MULTIPART to OFF or NO stores any content in one part. Setting the value of MULTIPART to ON or YES stores documents in multiple parts on the Content Manager server.

DEFAULT
OFF

EXAMPLE
```
MULTIPART ON
```

ODAGENTS *number*
For use with Content Manager OnDemand only. Using this keyword, you can specify the maximum number of parallel CMOD sessions (name: *archagentod*). The default value is 0.

EXAMPLE:
```
ODAGENTS 1
```

ODHOST *hostname*
For use with Content Manager OnDemand only. Using this keyword, you can specify the host name or IP address of the CMOD library server.

EXAMPLE:
```
ODHOST asterix
```

ODUSER *username*
Using this CMOD-specific keyword, you can grant a CMOD user the right to view, add, and delete documents contained in the application group that you specified by using the APPGROUP keyword. At the same time, this user gains access to the folder specified by the FOLDER keyword.

EXAMPLE:
```
ODUSER admin
```

PROTECTION *prot_flags*|**OFF**
This is an additional keyword for the ARCHIVE statement. Using this keyword, you can specify the default protection for this archive ID.

The value of *prot_flags* is a combination of the letters r (read), c (create), u (update), and d (delete). The default is **rcud**. When set to OFF, the archive is not protected, that is, all operations are allowed. Typically, OFF is used in connection with CommonStore 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 CommonStore Server produces some additional information. The output is written to an output unit called stdout, which is normally the console. The default value is OFF.

Note
Set the keyword to ON for tracing purposes, for example, when you set up the CommonStore Server or track down errors.

SERVER *server_name*
For use with Tivoli Storage Manager only. Using this keyword, you specify the name of the Tivoli Storage Manager library server. CommonStore establishes a connection to this server with the subsequent definitions. The dsm.sys file contains all necessary communication parameters for Tivoli Storage Manager.

EXAMPLE:

```
SERVER ADSMSERV01
```

SERVICE_TRACEFILE *filename*
Specifies an additional trace file to record the startup and shutdown of the CommonStore service. This trace file is useful only for analyzing problems with the CommonStore service. If the keyword is not specified, a service trace file is not written.

STARTUP_TRACEFILE *filename*
Specifies the full file name of the startup trace file. When a non-empty file name is specified, all CommonStore 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 the keyword is not specified, a startup trace file is not written.

EXAMPLE:

```
STARTUP_TRACEFILE C:\Program Files\ibm\csld\startup.trace
```

Note
The startup trace file is re-written at each start of the CommonStore Server.

STORAGETYPE ADSM|VI|VI400|ONDEMAND
Using this keyword, you specify the archive system to which the logical archive is attached. CommonStore needs this information to select the proper archiving agent. You must specify the storage type for each logical archive that you define. Specify one of the following values:

- `ADSM` for Tivoli Storage Manager archives
- `VI` for Content Manager archives
- `VI400` for Content Manager for iSeries archives
- `ONDEMAND` for Content Manager OnDemand archives

EXAMPLE:

```
STORAGETYPE ADSM
```

Note

This keyword refers to the corresponding ARCHIVE statement and is used for all archives. Any further archiving parameters that you specify apply to the archive with this setting.

SYSTEMTYPE DOMINOSYSTEM|EXCHANGESYSTEM|SAPSYSTEM
Specifies whether CommonStore is used for Lotus Domino, Exchange 2000, or SAP. These settings affect only the LUM licensing part.

EXAMPLE:

```
SYSTEMTYPE SAPSYSTEM
```

TEMPPATH *path*
Specifies the directory in which the CommonStore Server writes temporary files needed for processing.

If this setting is missing in your server configuration profile, CommonStore checks the environment variable TMPDIR. If this variable is not set either, the temporary files are written to the system's temporary directory.

EXAMPLE:

```
TEMPPATH c:\temp
```

TRACE ON|OFF
If set to ON, the CommonStore Server writes trace information to the trace file.

If you specify one or more of the following parameters instead of ON or OFF, you can force the CommonStore Server to write only specific trace information:

- FILEIO (for information on the input and output file activities)
- ARCHPRO (for information coming from the archpro program)
- AGENTS (for information coming from the Content Manager agents)

The value ON includes all of these parameters.

Examples

- `TRACE ON`
- `TRACE ARCHPRO`

Note

This parameter should be used only for the purpose of detecting problems. The default value is **OFF**. Do not delete the trace file while the CommonStore Server is running as this affects further writing to this file.

TRACEFILE *filename*
Specifies the trace file for the CommonStore Server. All trace information stored is stored in this file. The value filename specifies the path and the name of this file. CommonStore uses this setting only if tracing has been activated. The default value is archint.trace.

EXAMPLE:

```
TRACEFILE C:\Program Files\ibm\csld\server\instance01\archint.trace
```

Attention

Do *not* delete a trace file while the CommonStore Server is running. Stop the CommonStore Server first by using the archstop command.

TRACEMAX *number*
Specifies the maximum size of the CommonStore Server trace file in KB.

EXAMPLE:

```
    TRACEMAX 500
```

TRUNCATE_ATTRIBUTE ON|OFF
If you switch truncation on, attribute values that are longer than the maximum length defined in the Content Manager for iSeries archive are cut off so that the values fit in the space reserved for them. If you switch truncation off, documents with attribute values longer than the specified maximum cannot be archived. To switch truncation on, you add the following line to the server configuration profile:

```
TRUNCATE_ATTRIBUTE ON
```

VIAGENTS *number*
For use with Content Manager and VisualInfo only. Using this keyword, you specify the number of client sessions (name: *archagentvi*) that the CommonStore server starts in parallel. The default value is 0.

EXAMPLE:

```
    VIAGENTS 3
```

VIUSER *username*
For use with Content Manager and VisualInfo only. Using this keyword, you can specify the user name for the log-in procedure.

WEBDPS *number*
Specifies the number of parallel sessions for CommonStore Web access. The default value is 0.

EXAMPLE:

```
    WEBDPS 5
```

Note

By default, Web access to the CommonStore archive is not enabled. To enable Web access, specify the WEBDPS keyword.

WEBPORT *port*
Using this 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 in the *~archive* section 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 CommonStore Server is running, the default TCP/IP port 8085 can be used.

WEBROOT *path*

Specifies the directory in which the HTTP interface of the CommonStore Server expects to find the files necessary for Web operations, such as browser viewing.

EXAMPLE:

```
WEBROOT /home/csadm1/webroot
```

Appendix B. CommonStore Server commands

This command reference serves as a quick look-up guide that allows more experienced users to control CommonStore from a Windows Command Prompt. To illustrate the command syntax, syntax diagrams are employed. If you are not familiar with reading syntax diagrams, read the brief explanation in Appendix H, "Reading syntax diagrams" on page 261.

archadmin

Purpose

This program allows a connection to a CommonStore Server to be opened in order to view the messages issued by the CommonStore Server. It is possible to open the connection across machine and platform boundaries.

Format

archadmin

```
►►──archadmin──┬──┬──────────────────┬── -m──machine── -p──port number──┬──────────────►◄
               │  └── -i──ini file──┘                                   │
               └── -h───────────────────────────────────────────────────┘
```

Parameters

-m *machine*
: Specifies the machine name or IP address of the computer on which the archpro program is running.

-p *port number*
: Specifies the fixed port used by the archpro program.

-i *ini file*
: Specifies the path and the file name of the server configuration profile used by the archpro program. Using this parameter, you can specify a file on the local machine only.

-h Displays help information on how to use the archadmin command.

Comments

You connect a computer to another computer running the archpro program by specifying the machine name and the port number (parameters -m and -p). If you do not specify a machine name, CommonStore assumes that the machine to connect to is the one named local_host. The fixed port number can also be read from the server configuration profile. If you neither specify *-p*, nor *-i*, the required information is taken from the standard server configuration profile, the archint.ini file. Only port numbers above `5000` are accepted. Connections between different operating systems, for example Windows and AIX, are supported.

Examples

`archadmin`
: Connects to the archpro program by reading the port number from the archint.ini file

`archadmin -p 5510`
: Connects to the archpro program by using the fixed port 5510

`archadmin -m obelix -p 5510`
: Connects to the archpro program running on a computer named obelix; the connection is established by using the fixed port 5510

`archadmin -m 9.164.10.20 -p 5510`
: Connects to the archpro program running on a computer whose IP address is 9.164.10.20; the connection is established by using the fixed port 5510

`archadmin -i C:\Program Files\IBM\csld\server\archint2.ini`
: Connects to the archpro program running on the local machine. The port number is read from the specified server configuration profile (archint2.ini).

Note:
If the CommonStore Server is installed on a machine other than the one from which you want to run the archadmin command, you must copy the message catalog file CSSrvMsg.dll to the directory containing the archadmin.exe file.

archpro

Purpose

The archpro program is the continuously running CommonStore main program which controls all of the other CommonStore components.

Format

archpro

```
►►──archpro──────────────────────────────────────────────────────────────────►

►──┬──┬──────────────┬──┬──────────────────────────────────────┬──┬──────────┬──┬──────►◄
   │  └─ -i──ini file─┘  └─-f  serverpasswd──┬─────────────────────┬┘  └─ -n──name─┘  │
   │                                          └─srv──┬──────────────┬┘                  │
   │                                                 └─node──┬──────┬┘                  │
   │                                                         └─passwd─┘                 │
   ├─ -f license──────────────────────────────────────────────────────────────────────┤
   └─ -h──────────────────────────────────────────────────────────────────────────────┘
```

Parameters

-i *ini file*
: Specifies the path and the file name of the server configuration profile that you want to use, where *ini file* is the file name including path information.

-f serverpasswd *[srv [node [passwd]]]*
: Use this parameter to specify the passwords for Tivoli Storage Manager, Content Manager, and Content Manager OnDemand. You only have to do this once, when you set up CommonStore.

-f license
Use this parameter to enroll a production license. You are then prompted to specify the license file.

-n *name*
Specifies the instance name of the server instance that you want to use.

-h Displays help information on how to use the archpro command.

Comments

Specify the name of the server configuration profile (ini file) by using the *-i* parameter if the CommonStore software is distributed among several subdirectories of the root directory. Specify the *-i* parameter before any other parameter.

Examples

`archpro`
Starts the CommonStore Server with the default server configuration profile

`archpro -i C:\Program Files\IBM\csld\server\archint2.ini`
Starts the CommonStore Server with the specified server configuration profile

`archpro -f serverpasswd`
Causes CommonStore to prompt you for all archive passwords

`archpro -f serverpasswd SRV`
Causes CommonStore to prompt you for the passwords of the archive and of all nodes or users with access to this archive on the server named SRV

`archpro -f serverpasswd SRV USR`
Causes CommonStore to prompt you for the passwords of the archive and of the node or user named USR on the server named SRV

`archpro -f serverpasswd SRV USR PWD`
Specifies the password PWD for the node or user USR with access to the archive on the server named SRV. Using the parameter in this way omits the prompting for the password.

`archpro –f license`
Causes CommonStore to prompt you for a license file of a productive license in order to enroll it

archservice

Purpose

This program contains the whole service functionality of CommonStore for Exchange Server.

Format

archservice

```
►►──archservice──┬─┬─install─┬──┬─────────────┬──┬───────────┬─┬──────────►◄
                 │ ├─start───┤  └─-i──ini file─┘  └─-n──name──┘ │
                 │ ├─stop────┤                                  │
                 │ ├─remove──┤                                  │
                 │ └─status──┘                                  │
                 └─-h───────────────────────────────────────────┘
```

Parameters

-h Displays help information on how to use the archservice command

-i *ini file*
: Specifies the path and file name of the server configuration profile that you want to use, where *ini file* stands for the full path and the file name

install Installs the CommonStore service

-n *name*
: Specifies the name for an instance of the CommonStore service

remove
: Removes the CommonStore service

start Starts the CommonStore service

status Displays the current status of the CommonStore service

stop Stops the CommonStore service

Comments

You must specify the name of the server configuration profile (ini file) by using the *-i* parameter if the CommonStore software is distributed among several direct subdirectories of the root directory.

The instance name permits multiple installations of CommonStore service on a single machine.

The corresponding service instance is labeled `CommonStore_<name>`.

Examples

`archservice install`
: Installs CommonStore as a Windows 2000 service

`archservice install -i C:\Program Files\IBM\csld\server\archint2.ini -n 2`
: Installs an instance of the CommonStore service using the server configuration profile archint2.ini, located in the C:\Program Files\IBM\csld directory. The new instance is named CommonStore_2 (the prefix CommonStore_ plus the value that you specify).

`archservice remove -n 2`
: Removes the instance CommonStore_2 of the CommonStore service

`archservice start -n 2`
: Starts the instance CommonStore_2 of the CommonStore service

`archservice stop -n 2`
: Stops the instance CommonStore_2 of the CommonStore service

`archservice status -n 2`
: Displays the status of the CommonStore service instance CommonStore_2

Note:
Do not start the archservice program from the command line *without* parameters. This way of running the archservice program is restricted to internal calls.

archstop

Purpose

This program completely stops the CommonStore Server by means of a regular shutdown.

Format

archstop

```
►►──archstop──┬─┬───────────────┬── -p──port number──┬─────┬─┬──────────────────►◄
              │ └─ -i──ini file─┘                    └─now─┘ │
              └─ -h──────────────────────────────────────────┘
```

Parameters

-p *port*
Stops the archpro instance that uses the specified port

-i *ini file*
Stops the archpro instance using the port listed in the specified server configuration profile

now
Stops the specified archpro instance immediately, without waiting for active jobs to complete

-h Displays help information on how to use the archstop command

Comments

The port number is essential in establishing a connection between a computer and the archpro program. The fixed port number can also be read from the server configuration profile. If you neither specify -p, nor -i, the required information is taken from the standard server configuration profile, the archint.ini file. Only port numbers above 5000 are accepted.

Examples

`archstop`
Stops archpro using the port number listed in the standard server configuration profile

`archstop -p 5510`
Stops the archpro instance using port 5510 after all jobs have been completed

`archstop -p 5510 now`
Stops the archpro instance using port 5510 immediately

`archstop -i C:\Program Files\IBM\csld\server\archint2.ini`
: Stops the archpro instance using the port listed in the specified server configuration file

Appendix C. Sample profiles

The following sections describe the sample profiles for Tivoli Storage Manager, Content Manager, Content Manager for iSeries, and Content Manager OnDemand as delivered on the product CD-ROM. Although these profiles were tested in a particular environment, it is unlikely that you can simply copy a profile as it is. Most probably, you must make adjustments for a profile to work with your setup.

If, for example, you decide to use archives of more than one type, you can adapt the sample profile file for Content Manager by adding Tivoli Storage Manager archives as described in the sample profile for Tivoli Storage Manager. Bear in mind, however, that some archive systems do not work in a mixed environment.

CommonStore Server on Windows using Tivoli Storage Manager

```
##########################################################################
#                                                                        #
#                 Sample configuration profile for                       #
#                                                                        #
#        IBM Content Manager CommonStore for Lotus Domino 8.1            #
#                                                                        #
#              on Windows using Tivoli Storage Manager                   #
#                                                                        #
#      (c) Copyright IBM Corporation 2000, 2002. All rights reserved.    #
#                                                                        #
##########################################################################

#------------------------------------------------------------------------#
# 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:\Program Files\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:\Program Files\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
```

CommonStore Server on AIX using Tivoli Storage Manager

```
#########################################################################
#                                                                       #
#                 Sample configuration profile for                      #
#                                                                       #
#        IBM Content Manager CommonStore for Lotus Domino 8.1           #
#                                                                       #
#                on UNIX using Tivoli Storage Manager                   #
#                                                                       #
#      (c) Copyright IBM Corporation 2000, 2002. All rights reserved.   #
#                                                                       #
#########################################################################

#-------------------------------------------------------------------------#
# 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
```

CommonStore Server on Windows using Content Manager

```
#########################################################################
#                                                                       #
#                  Sample configuration profile for                     #
#                                                                       #
#        IBM Content Manager CommonStore for Lotus Domino 8.1           #
#                                                                       #
#                  on Windows using Content Manager                   #
#                                                                     #
#    (c) Copyright IBM Corporation 2000, 2002. All rights reserved.     #
#                                                                     #
#######################################################################

#---------------------------------------------------------------------#
# 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:\Program Files\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:\Program Files\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
```

CommonStore Server on AIX using Content Manager

```
#########################################################################
#                                                                       #
#                 Sample configuration profile for                      #
#                                                                       #
#        IBM Content Manager CommonStore for Lotus Domino 8.1           #
#                                                                       #
#                  on UNIX using Content Manager                        #
#                                                                       #
#     (c) Copyright IBM Corporation 2000, 2002. All rights reserved.    #
#                                                                       #
#########################################################################

#-----------------------------------------------------------------------#
# 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
```

CommonStore Server on Windows using Content Manager Version 8

```
##########################################################################
#                                                                        #
#                 Sample configuration profile for                       #
#                                                                        #
#        IBM Content Manager CommonStore for Lotus Domino 8.1            #
#                                                                        #
#              on Windows using Content Manager Version 8                #
#                                                                        #
#     (c) Copyright IBM Corporation 2000, 2002. All rights reserved.     #
#                                                                        #
##########################################################################

#------------------------------------------------------------------------#
# 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:\Program Files\ibm\csld\server\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:\Program Files\ibm\csld\server\instance01

#-----------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
#-----------------------------------------------------------------------#
REPORT          ON

#-----------------------------------------------------------------------#
# TRACE     ON OFF RFC_OFF ...
# Switches trace levels
#
# Examples for usage:
# TRACE         ON
# TRACE         ON RFC_OFF
#-----------------------------------------------------------------------#
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\


#-----------------------------------------------------------------------#
# Archive ID definitions
#
# We just define one archive ID required for the CSLD_MAILDEMO item type.
# The archive is Content Manager Version 8 (keyword CM).
# The name of the item type is CSLD_MAILDEMO.
# The name of the library server is LIBSVI.
# Log into the archive with user ID CSLD.
#-----------------------------------------------------------------------#
ARCHIVE A1
    STORAGETYPE     CM
    ITEM_TYPE       CSLD_MAILDEMO
    LIBSERVER       LIBSVI
    CMUSER          CSLD

#-----------------------------------------------------------------
# Number of parallel instances of the CommonStore child programs
#-----------------------------------------------------------------
ADSMAGENTS          0
CMAGENTS            2
VIAGENTS            0
ODAGENTS            0
CMAGENTS            0
DOMINODPS           5
WEBDPS              2

#-----------------------------------------------------------------------#
# TCP/IP port numbers used by CommonStore.
# ARCHPRO_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
ARCHPRO_PORT        8012

#-----------------------------------------------------------------------#
# LUM
#-----------------------------------------------------------------------#
SYSTEMTYPE DOMINOSYSTEM

#-----------------------------------------------------------------
# End of profile
#-----------------------------------------------------------------
END
```

CommonStore Server on Windows using Content Manager for iSeries

This appendix contains a printout of the sample server configuration profile, which is part of your product package. The sample profile contains the attributes and definitions necessary to run CommonStore for Lotus Domino with Content Manager for iSeries. You can adapt the sample profile for your needs and then replace the existing profile with the modified sample profile.

```
#########################################################################
#                                                                       #
#                 Sample configuration profile for                      #
#                                                                       #
#        IBM Content Manager CommonStore for Lotus Domino 8.1           #
#                                                                       #
#             on Windows using Content Manager for iSeries              #
#                                                                       #
#    (c) Copyright IBM Corporation 2000, 2002. All rights reserved.     #
#                                                                       #
#########################################################################

#-----------------------------------------------------------------------#
# TIP:
#
# If a directory or a file name in a path in this ini file includes a blank,
# you must enclose the path specification in single quotes (').
#
# Example:
#
# BINPATH  'C:\Program Files\csld'
#
#-----------------------------------------------------------------------#

#-----------------------------------------------------------------------#
# Path (directory) of the CommonStore binaries
#-----------------------------------------------------------------------#
BINPATH  'C:\Program Files\ibm\csld\bin'

#-----------------------------------------------------------------------#
# Path (directory) of the CommonStore instance.
# Must point to the installation directory of the appropriate server
# instance.
# All instance-related files reside in this directory, e.g.
# trace files, log files, configuration files, and so on.
#-----------------------------------------------------------------------#
INSTANCEPATH            C:\Program Files\ibm\csld\bin\instance01
```

```
#--------------------------------------------------------------------------#
# REPORT        ON|OFF
# Switches reporting to STDOUT on or off
#--------------------------------------------------------------------------#
REPORT ON

#--------------------------------------------------------------------------#
# TRACE     DISP FILEIO ARCHPRO AGENTS DEBUG ADSM RFC ON OFF
# Switches tracing on or off and defines the 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
#
# This sample contains just one archive ID definition for the
# CSLD_MAILDEMO indexclass.
# The type of the archive is VI.
# The name of the index class is CSLD_MAILDEMO.
# The name of the library server is LIBSVI.
# To log on to this archive, use the user ID CSLD.
#--------------------------------------------------------------------------#
ARCHIVE A1
STORAGETYPE VI400
INDEX_CLASS CSLD_MAILDEMO
LIBSERVER LIBSVI
VIUSER CSLD
TRUNCATE_ATTRIBUTE  ON

#------------------------------------------------------------------
# Number of parallel instances of  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 the archstop program
# DOMINOPORT is the port that the CommonStore Server turns to for
# requests from CSLD Tasks. Must be identical to the port number in the
# profile document of the configuration database.
# WEBPORT is the port that the CommonStore Server turns to for
# HTTP requests. It must be identical to the port number in the profile
# document of the configuration database.
#-----------------------------------------------------------------------#
DOMINOPORT 47111
WEBPORT 8085
ARCHWIN_PORT 8012

#-----------------------------------------------------------------------#
# LUM
#-----------------------------------------------------------------------#
SYSTEMTYPE DOMINOSYSTEM

#-------------------------------------------------------------------
# End of profile
#-------------------------------------------------------------------
END
```

CommonStore Server on Windows using Content Manager OnDemand

```
#########################################################################
#                                                                       #
#                Sample configuration profile for                       #
#                                                                       #
#       IBM Content Manager CommonStore for Lotus Domino 8.1            #
#                                                                       #
#            on Windows using Content Manager OnDemand                  #
#                                                                       #
#    (c) Copyright IBM Corporation 2000, 2002. All rights reserved.     #
#                                                                       #
#########################################################################

#-----------------------------------------------------------------------#
# 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:\Program Files\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:\Program Files\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, application group 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
```

CommonStore Server on AIX using Content Manager OnDemand

```
#########################################################################
#                                                                       #
#                Sample configuration profile for                       #
#                                                                       #
#       IBM Content Manager CommonStore for Lotus Domino 8.1            #
#                                                                       #
#              on UNIX using Content Manager OnDemand                   #
#                                                                       #
#      (c) Copyright IBM Corporation 2000, 2002. All rights reserved.   #
#                                                                       #
#########################################################################

#-----------------------------------------------------------------------#
# 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, application group 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. Troubleshooting

This section provides solutions to known problems. Please refer to this section before you call IBM technical support. The most common problems are covered here. Applying a solution described in this book is usually less time-consuming than using external help. For problems with the supported archive systems, see the corresponding sections in this book:

- “Tivoli Storage Manager – troubleshooting” on page 110
- “Content Manager – troubleshooting” on page 114
- “Content Manager OnDemand – troubleshooting” on page 131

Problems with the CSLD Task

See this section if you encounter problems that are probably related to the task component.

Problem: Instead of error messages, jobs contain “No message found for...”

Solution: The error message file csnmsg.msg cannot be found because the environment variable CSNBASE is not set correctly to the CommonStore binary directory, or because the csnmsg.msg file has been deleted.

Problem: I cannot shutdown a task instance for some reason. After I stopped an instance using the NT Task Manager, my mail client shows strange behavior.

Solution: The internal thread and session handling of most supported mail clients is highly volatile. Log out and in again, or reboot.

Problem: After starting a task, it displays the CommonStore user name, then “hangs” (does not display a log-in prompt).

Solution: You have two copies of the nnotes.dll file, one of which was installed with your mail client software. Remove one of them.

Problem: When starting up a CommonStore task, it stops with the error message “The ID file is locked by another process. Try again later.”

Solution: You started another CommonStore 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 CommonStore Task does no harm.

Problem: A job contains the error message “The archive server returned error code `<errorcode>`.”

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

Problem: After processing, a job document contains the Content Manager error "Archiving to CommonStore failed. The archive server returned error code 6056."

Solution: Most probably a Content Manager setup problem. Error 6056 indicates a generic Content Manager error. To find out the exact error code, perform the following steps:

1. Look up the error code, the extension error code, and the reason code in the CommonStore csserror.log file. The extension error code contains the "real" error number.
2. Look up the extension error code in the Content Manager documentation for a description of this error code.

   Extension Code = 7389
   When an index class is created, you must wait for a while until you can use it because Content Manager must create a dynamic link library (.dll file) for the index class.

   Extension Code = 7937
   A field in the document that you want to archive is exceeds 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 receive this error message.

Problem: "Export filter is unable to export document to `<format>` file."

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

Problem: I have set the mail client password using the **csld -f serverpasswd** command, but the task still prompts for the mail client ID.

Solution: Check whether the initialization file of your mail client 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 an initialization file containing a particular ID.

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

Solution: When the task starts, it displays the current ID. The ID is always stored in your default initialization file. If you use the -i parameter to specify a different .ini file, the task uses the ID in that file. When you switch to a different ID in Lotus Notes, the ID is stored in the default initialization file.

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

Solution: CommonStore for Lotus Domino is a Lotus Notes application, and therefore requires that the dynamic link libraries of Lotus Notes are installed. The notes.dll library belongs to the Lotus Notes client application, and resides in the Lotus Notes installation directory. Add this directory to the PATH environment variable.

Problem: 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 e-mail I receive an error message saying that the user ID does not exist.

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

Problem: A CSLD Task instance ignores the jobs that I created.

Solution: First, make sure that the value in the database server field **sourceSrv** submitted with the job is identical to the server name given in the task profile (case is ignored). Then, check if both are specified in abbreviated format, for example, BOEDOM1/IBM_IDE. Also, check the **requestType** field of the job. See Chapter 38, “Creating job documents” on page 165 for a description of job parameters. Also check if the CommonStore user ID (the ID the CommonStore task is running under) is assigned the role *CSLDUsers*. See “Creating the job database” on page 137 for details.

Problem: The archive server returned error code 1.

Solution: The archive does *not* have an index class with the name you specified. Check the spelling of your index class name in the server configuration profile (usually archint.ini). Index class names are case-sensitive.

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

Solution: The archpro program has nothing to do with your mail client. It is an independent application that archives the files it receives from a task instance. A task is a job that the CommonStore Server component processes.

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

Solution: You probably forgot to add an entry to the csmimes.properties file in order to map the content type to a MIME type that the browser can understand. Another explanation is that you used the default mapping (file extension .unk on Windows) to map all files to the same content type instead of assigning every file extension its own content type.

Problems with the CommonStore Server

See this section if you encounter problems that are probably related to the CommonStore Server component.

Problem: Archiving takes too long. Where is the bottleneck?

Solution: See the most common reasons:

- The archive server is overloaded. Content Manager and Tivoli Storage Manager can import only a limited number of documents simultaneously.

- The number of archiving agents is too low. If you create archiving jobs faster than the current number of agents can import documents, the documents are written to a queue. Increase the number of agents by setting the values of the appropriate keyword accordingly in the server configuration profile (usually the archint.ini file):

  ADSMAGENTS
  For ADSM and Tivoli Storage Manager

  VIAGENTS
  For Content Manager

  Do not specify more than ten agents because this slows down the system due to swapping.
- The CommonStore Server has too little memory.
- Tracing is enabled.

Problem: The CommonStore Server does not start.

Solution: The CommonStore Server checks at startup whether all settings are correct and whether it was possible to establish a connection to the archive servers. Furthermore, all paths and files specified in the server configuration profile (usually 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, enable tracing by setting the keyword as follows:
   ```
   TRACE = ON
   ```
   Look for error messages in the CommonStore Server trace files archint.trace and startup.trace.
2. Determine which child process has problems with the connection. The archpro.exe program normally displays a corresponding error message.

   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 agent at a time.

Problem: How can I tell that a CommonStore child component is working (is ready)?

Solution: The connection is working when the archpro program displays the following messages:

- archpro.exe is informed that xxx has started (from the agents)
- archpro.exe is informed that xxx is ready to obtain order (from the agents)

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 agents, this 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 correctly.

Problem: The Windows commands **net start** and **net stop** do not start or stop the CommonStore service.

Solution: Use **archservice start** and **archservice stop** instead.

Reporting errors to the support team

To report an error to the CommonStore support team, open a problem management record (PMR). Make sure that you include the information listed in this section. When asked for files, include the versions that were created at the time when the error occurred.

- Description of your environment. Try to be as precise as possible when specifying the number of Lotus Domino servers, the hardware configuration, the client configuration, the archive systems and their version numbers, any installed fixpacks, and other relevant software.
- Scenario description. What do you want to do? What exactly does not work?
- Error messages (literally)
- Is the problem reproducible? If yes, tell us how.
- Error log file
- Server configuration profile (usually archint.ini)
- Server trace file (usually archint.trace). Note that the archint.trace file is truncated when it has reached its maximum file size. So make sure that it contains trace information of the time when the problem occurred. If not, try to reproduce the problem and stop the system.
- Screen shots of your configuration dialog boxes. This way, we can verify the values that you entered.
- If CommonStore does not retrieve a document, can you retrieve it using the Content Manager client or the Content Manager OnDemand client?

Important

The CommonStore support team expects the administrator to have read the documentation. Most problems occur in connection with incorrectly configured archives. In such cases, we ask you to consult Tivoli Storage Manager, Content Manager, or Content Manager OnDemand support.

Appendix E. CommonStore Server return codes

Refer to the following list to look up the meaning of a particular return code. The codes are listed in numerical order. Therefore, take down the number in parantheses when you receive a return code message.

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 because the startup check of the corresponding archive failed.

CS_RC_VERSION_ERROR (-6)
CommonStore does not start because it detected a child process that was created by the wrong version of a program file. Replace the corresponding executable file with the correct version.

CS_RC_CHILD_TERMINATED (-10)
CommonStore has detected that a child process has terminated unexpectedly. The child process is restarted automatically.

CS_RC_SHUTDOWN (-99)
CommonStore was shut down by a shutdown request from the archstop program.

CS_RC_SHUTDOWN_NOW (-100)
CommonStore was shut down by an immediate shutdown request from the archstop program.

CS_RC_NOMEM (-110)
A CommonStore process is running out out of memory.

CS_RC_NOTFOUND (-115)
The requested data was not found. See the CommonStore trace file for further details.

CS_RC_ERRDELETE (-116)
The archived document or component cannot be deleted. This error code can also occur when data is appended to a component or a component is updated.

CS_RC_NOTSUPPORTED (-118)
CommonStore is unable to carry out requested operation. Such operations can be, for example, index transfer requests sent to an agent of Tivoli Storage Manager or invalid actions for an update operation.

CS_RC_FILENOTFOUND (-200)
CommonStore cannot find a file or document.

CS_RC_UNKNOWNDOC (-201)
The requested document cannot be found in the archive.

CS_RC_QUERYNOTFOUND (-202)
The document cannot be found in the archive. The query was unsuccessful.

CS_RC_ACCESSDENIED (-203)
You do not have the proper access rights to process the archived document in this way.

CS_RC_DOCEXISTS (-204)
The document cannot be archived because the same document already exists in the archive.

CS_RC_ERRCERT (-205)
CommonStore failed when it tried to administer a certificate.

CS_RC_COMPNOTFOUND (-206)
The component you want to append data to cannot be found in the archive. It probably does not exist.

CS_RC_CONTREP_NOTFOUND (-207)
The content repository (archive ID) you specified cannot be found in the server configuration profile (usually archint.ini).

CS_RC_INVALIDOFFSET (-208)
The offset specified for the part retrieval goes beyond the end of document.

CS_RC_FREESEARCH_NOTFOUND (-210)
The specified pattern cannot be found in a free search request.

CS_RC_ATTRSEARCH_NOTFOUND (-211)
The specified pattern cannot be 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)
The document cannot be modified in the archive because it was archived with CommonStore version 1.

CS_RC_LOGSYS_NOTFOUND (-215)
The DESTINATION statement in the server configuration profile does not contain the specified logical system or it does not contain any logical system at all.

CS_RC_NO_ATTR_ARCHIVED (-216)
The plain document data was archived successfully, but all attributes provided in the attribute list were dropped because the archive cannot store them. This message is typically issued by the TSM agent when processing an archiving request with additional attributes created by CommonStore.

CS_RC_NOCOPYGROUP (-217)
CommonStore cannot use the specified TSM management class due to a problem with the copy group.

CS_RC_TRANSFORM_FAILED (-219)
The invocation of the TRANSFORM command failed.

CS_RC_NO_AGENT (-220)
The CommonStore Server has received a request for an agent that is not configured in the server configuration profile (usually archint.ini). This request has been cancelled immediately.

CS_RC_QUEUE_ERROR (-221)
: The CommonStore Server cannot queue asynchronous jobs on disks. This job has been cancelled immediately. The CommonStore Server shuts down because a normal (safe) operation is not possible any more. Check the CommonStore Server queue directory before restarting the CommonStore Server.

CS_RC_SAP_ATTR_NOTALLOWED (-240)
: An archiving operation failed because the attribute list in the archiving request contains one of the reserved SAP attributes. CommonStore creates these attributes automatically. They must not 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)
: An error occurred when CommonStore tried to open a file or request its status.

CS_RC_FILEREAD_ERROR (-251)
: An error occurred when CommonStore tried to read data from a file.

CS_RC_FILEWRITE_ERROR (-252)
: An error occurred when CommonStore tried to write data to a file.

CS_RC_ADSM_ERROR (-260)
: An error has occurred in the TSM agent, but the related API call did not fail.

CS_RC_VI_ERROR (-261)
: An error has occurred in the Content Manager agent, but the related API call did not fail.

CS_RC_OD_ERROR (-262)
: An error has occurred in the OnDemand agent. Check the csserror.log file for a description of the error.

CS_RC_ATTR_NOT_FOUND (-263)
: The value of an attribute is too long to be stored in a Content Manager for iSeries archive.

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 search attribute pattern is incorrect.

CS_DO_ARCHIVELIST_EMPTY (-1004)
A search operation failed because the list of archive IDs is empty.

CS_DO_FOLDER_ISEMPTY (-1006)
The folder operation cannot be completed because the folder is empty.

CS_VI_RETRIEVE_ERROR (-1112)
The retrieval operation failed although the API functions did not return an error. The error occurred during 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 Content Manager API are inconsistent. The are numbers missing in the sequence.

CS_VI_INDEXCLASS_NOTFOUND (-1118)
The index class cannot be found on the specified Content Manager server.

CS_VI_CONTENTCLASS_NOTFOUND (-1120)
The content class cannot be found on the specified Content Manager server.

CS_VI_NO_DATA_RETURNED (-1121)
An API function does not return the requested data. However, the API function itself did not fail.

CS_BC_RFC_ABORT_BY_USER (-7095)
A user has stopped Archbc processing (no error).

CS_BC_RFC_ERROR_TABLE (-7096)
The creation of the bar-code entry table failed. CommonStore was unable to send the bar codes to SAP.

CS_BC_RFC_ERROR_INSERT (-7097)
The bar codes could not be sent to SAP because the remote function call ARCHIV_BARCODE_INSERT_RFC resp. BAPI_BARCODE_SENDLIST failed.

CS_BC_RFC_ERROR_CONNECTION (-7098)
CommonStore could not establish a connection with SAP. The remote function calls ARCHIV_BARCODE_INSERT_RFC or BAPI_BARCODE_SENDLIST could not be carried out. No bar codes were sent to SAP.

CS_BC_RFC_NO_DATA_TO_SEND (-7099)
The bar-code table is empty. Thus, bar codes cannot be sent to SAP.

CS_RC_BC_ARCHIVE_FAILED (-7100)
The request to archive and send bar codes failed because the archiving operation failed. The bar code was not sent to SAP.

CS_VI_ITEM_IN_WRONG_INDEXCLASS (-7111
An item cannot be re-indexed because it neither resides in the scan index class nor in the target component index class.

CS_VI_ARCHIVE_HAS_NO_SCAN_IC (-7125)
The operation failed because a scan index class for the specified content repository (archive ID) is not defined.

CS_VI_ARCHIVE_HAS_NO_SCAN_WB (-7127)
: The operation failed because a scan workbasket for the specified content repository (archive ID) is not defined.

CS_VI_ARCHIVE_HAS_NO_ERROR_WB (-7128)
: The operation failed because an error workbasket for the specified content repository (archive ID) is not defined.

CS_RC_CWI_R3_CONNECTION_ERROR (-7201)
: The request to create a work item failed because CommonStore could not connect to SAP.

CS_RC_CWI_R3_FAILED (-7202)
: SAP returned an error as it received the request to create a work item.

CS_RC_CWI_ARCHIVE_FAILED (-7203)
: A work item could not be created because the archiving operation failed. The request to create a work item was not sent to SAP.

CS_RC_SOCKET_PROBLEM (-10000)
: A problem with the socket communication in CommonStore occurred. See the CommonStore trace file for further details.

CS_RC_NO_HANDLER (-10001)
: A CommonStore process received a request for which a message handler was not installed.

CS_RC_INTERNAL_ERROR (-10002)
: An internal error occurred in CommonStore or in the socket communication. See the CommonStore trace file for further details.

CS_RC_WRONG_TYPE (-10003)
: The parser detects a wrong data type when it receives a message over a socket.

Appendix F. 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 Lotus Notes Release 4.6, and added a number of design elements to it. For example, we added Lotus Script 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 Lotus Script using the CSLD Lotus Script 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 (for example, ″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 OnDemand for folder archiving” on page 130 or “Preparing Content Manager for folder archiving” on page 113.

OnDemand

- Create an application group (for example, ″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 19 on page 126.
- For the Notes folder archiving feature, create the Notes folder application group as described in “Preparing Content Manager OnDemand for folder archiving” on page 130. 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 6, “Archiving and retrieval tasks” on page 25 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 hit list*, 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 e-mail/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 hit lists.
- 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 (for example, 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 CommonStore 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 program.
- 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, for example, 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, that is, 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 rasterized format (that is, 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 the following method: 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 a rasterized 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 (that is, 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′*″, that is, 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 hit list with all the documents in the workbasket. The hit list (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 14, “Agents for processing archived 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 14, “Agents for processing archived 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 (for example, 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, for example, 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 the following basic 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.

Hit list
: When a query returns more hits than the number of documents directly built or no hits at all, a hit list 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.

Appendix G. 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 CommonStore on one server?
Answer: Yes, you can. CommonStore can be started in multiple instances on the same system. You only have to take care of distinct values of some ini parameters (for example, *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. CommonStore'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 CommonStore 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.

Appendix H. Reading syntax diagrams

This appendix explains how to read the syntax diagrams as used in Chapter 23, “Working with the CommonStore Server” on page 97 and in Appendix B, “CommonStore Server commands” on page 215.

To use a diagram, follow a path from left to right, top to bottom, adding elements as you go. In these diagrams, all spaces and other characters are significant.

Each diagram begins with a double right arrowhead and ends with a right and left arrowhead pair. Lines beginning with single right arrowheads are continuation lines.

```
►►──keyword──variable_value──────────────────────────────────────►◄
```

Keywords are all in lowercase, but can be entered in uppercase or in lowercase. Variable values that you provide are shown in *italics* and are usually in lowercase. Where values are shown in uppercase, they should be entered as they appear.

In a choice of items, the default item is always shown above the main line:

```
               ┌─default_value─┐
►►──keyword────┼─other_value───┼─────────────────────────────────►◄
               └─other_value───┘
```

Optional syntax elements are shown below the main line:

```
►►──keyword──┬───────┬───────────────────────────────────────────►◄
             └─value─┘
```

In some cases, when an item has additional items associated with it, an additional syntax diagram is shown that represents the full syntax of that item. For example, in the following syntax diagram, additional information that can or must be specified for ITEM1 appears in the “ITEM1 Variables” syntax diagram.

```
►►──keyword──keyword_name──┤ ITEM1 ├──ITEM2──────────────────────►◄
```

ITEM1 Variables:

```
├───┬─variable1─┬───────────────────────────────────────────────┤
    ├─variable2─┤
    └─variable3─┘
```

Sample syntax diagram

The following is a sample syntax diagram. It shows the expressions that you can form with the hello command.

Hello Command

```
►►──hello──┬──────────┬──┬──────────────┬─────────────────────────────►◄
           └─┤ Name ├─┘  └─┤ Greeting ├─┘
```

Name

```
├── name_of_person──────────────────────────────────────────────────────┤
```

Greeting

```
├──, how are you?───────────────────────────────────────────────────────┤
```

Valid versions of the hello command are:

```
hello
hello name
hello, how are you?
hello name, how are you?
```

Note that the space before the *name_of _person* value is significant and that, if you leave out a value for *name_of _person*, you still code the comma before **how are you?**.

Appendix I. Notices

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

Index

Readers' Comments — We'd Like to Hear from You

IBM Content Manager
CommonStore for Lotus Domino
Administrator's and Programmer's Guide
Version 8.1

Publication No. SH12-6742-00

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
SH12-6742-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development Dept. 0446
Schönaicher Strasse 220
D-71032 Böblingen
Federal Republic of Germany

Fold and Tape | **Please do not staple** | Fold and Tape

SH12-6742-00

Cut or Fold
Along Line

Program Number: 5724-B86

SH12-6742-00