STOREADM-19.1

Message Store Administration Guide

Table of Contents

1 Software version

This guide is published in support of Isode M-Store X.400 R19.1. It may also be pertinent to later releases. Please consult the release notes for further details.

2 Readership

This guide is intended for administrators who need to set up or manage Message Stores.

3 Related publications

Related topics are discussed in the volumes of the Isode documentation set listed below.

Volume	Title
SWADM-19.1	*M-Switch Administration Guide*
VAUADM-19.1	*M-Vault Administration Guide*
http://docs.isode.com	*M-Switch Advanced Administration Guide*

4 Typographical conventions

The text of this manual uses different typefaces to identify different types of objects, such as file names and input to the system. The typeface conventions are shown in the table below.

Object	Example
File and directory names	*isoentities*
Program and macro names	mkpasswd
Input to the system	`cd newdir`
Cross references	see Section 5, “File system place holders”
Additional information to note, or a warning that the system could be damaged by certain actions.	Notes are additional information; cautions are warnings.

5 File system place holders

Where directory names are given in the text, they are often place holders for the names of actual directories where particular files are stored. The actual directory names used depend on how the software is built and installed. All of these directories can be changed by configuration.

Certain configuration files are searched for first in *(ETCDIR)* and then *(SHAREDIR)*, so local copies can override shared information.

The actual directories vary, depending on whether the platform is Windows or UNIX.

Name	Place holder for the directory used to store...	Windows (default)	UNIX
(ETCDIR)	System-specific configuration files.	*C:\Isode\etc*	*/etc/isode*
(SHAREDIR)	Configuration files that may be shared between systems.	*C:\Program Files\Isode\share*	*/opt/isode/share*
(BINDIR)	Programs run by users.	*C:\Program Files\Isode\bin*	*/opt/isode/bin*
(SBINDIR)	Programs run by the system administrators.	*C:\Program Files\Isode\bin*	*/opt/isode/sbin*
(EXECDIR)	Programs run by other programs; for example, M-Switch channel programs.	*C:\Program Files\Isode\bin*	*/opt/isode/libexec*
(LIBDIR)	Libraries.	*C:\Program Files\Isode\bin*	*/opt/isode/lib*
(DATADIR)	Storing local data.	*C:\Isode*	*/var/isode*
(LOGDIR)	Log files.	*C:\Isode\log*	*/var/isode/log*
(CONFPDUSPOOLDIR)	Large PDUs on disk.	*C:\Isode\tmp*	*/var/isode/tmp*
(QUEDIR)	The M-Switch queue.	*C:\Isode\switch*	*/var/isode/switch*
(DSADIR)	The Directory Server's configuration.	*C:\Isode\d3-db*	*/var/isode/d3-db*

6 Support queries and bug reporting

A number of email addresses are available for contacting Isode. Please use the address relevant to the content of your message.

- For all account-related inquiries and issues: customer-service@isode.com. If customers are unsure of which list to use then they should send to this list. The list is monitored daily, and all messages will be responded to.
- For all licensing related issues: license@isode.com.
- For all technical inquiries and problem reports, including documentation issues from customers with support contracts: support@isode.com. Customers should include relevant contact details in initial calls to speed processing. Messages which are continuations of an existing call should include the call ID in the subject line. Customers without support contracts should not use this address.

- For all sales inquiries and similar communication: sales@isode.com.

Bug reports on software releases are welcomed. These may be sent by any means, but electronic mail to the support address listed above is preferred. Please send proposed fixes with the reports if possible. Any reports will be acknowledged, but further action is not guaranteed. Any changes resulting from bug reports may be included in future releases.

Isode sends release announcements and other information to the Isode News email list, which can be subscribed to from the address: http://www.isode.com/company/news-signup.php [http://www.isode.com/company/contact.php]

7 Export controls

Many Isode products use TLS (Transport Layer Security) to encrypt data in transit. This means that these products are subject to UK Export Controls.

For some countries (at the time of shipping this release, these comprise all EU countries, United States of America, Canada, Australia, New Zealand, Switzerland, Norway, Japan), these Export Controls can be handled by administrative process as part of evaluation or purchase. For other countries, a special Export License is required. This can be applied for only in context of a purchase order for those Isode products.

You must ensure that you comply with these Export Controls where applicable, i.e. if you are licensing or re-selling Isode products.

The TLS feature of Isode products is enabled by a TLS Product Activation feature. This feature may be turned off, and Isode products without this TLS feature are not export controlled. This can be helpful to support evaluation of Isode products in countries that need a special export license.

Isode products are used to administer sensitive data and so Isode strongly recommends that all operational deployments of Isode products use the export-controlled TLS feature.

All Isode Software is subject to a license agreement and your attention is also called to the export terms of your Isode license.

Chapter 1 Introduction

This chapter provides a basic overview of a Message Store, outlining its role within a message handling system.

1.1 Role of a Message Store

The primary role of a Message Store is to accept the delivery of messages from a Message Transfer Agent (MTA) on behalf of a user, and to retain them for subsequent retrieval by the user's User Agent (UA).

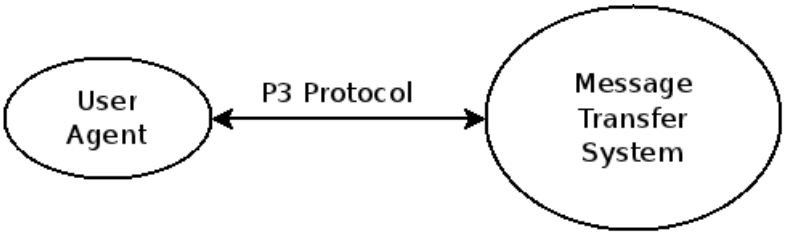
A secondary function is to provide indirect message-submission and message-administration services to the UA. This enables the Message Store to provide additional functionality to that available through direct submission to an MTA, such as forwarding messages residing in the Message Store.

A Message Store (MS) is, therefore:

- a provider of messaging services for User Agents (UAs), and
- a user of messaging services provided by Message Transfer Agents (MTAs).

In a messaging system which does not use a Message Store, outbound messages go directly and immediately from the User Agent (UA) to the Message Transfer Agent (MTA) and inbound messages go directly and immediately from the MTA to the UA, as shown in Figure 1.1, "Message Transfer without a Message Store".

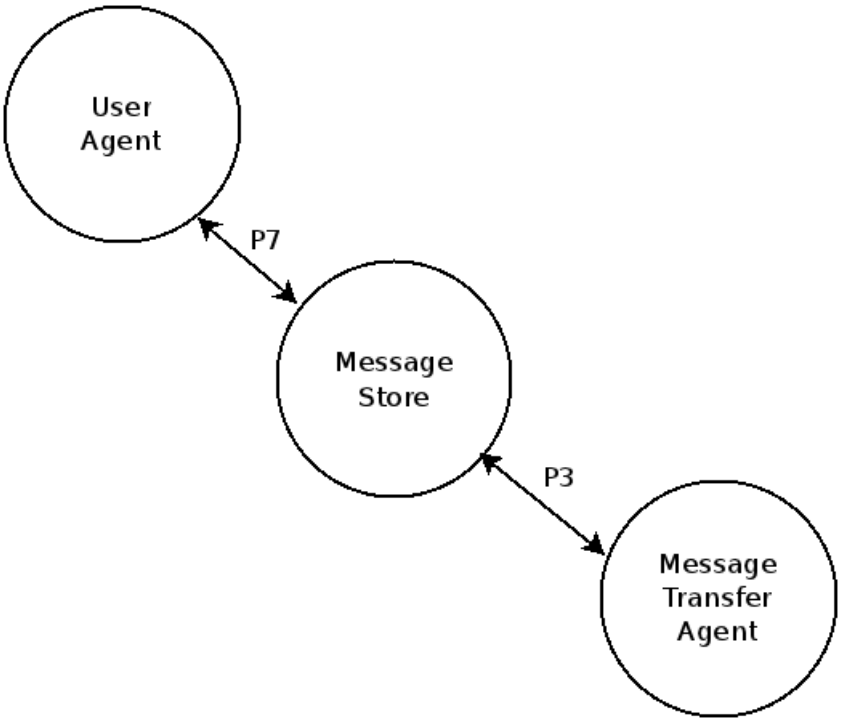
Figure 1.1. Message Transfer without a Message Store



In the basic CCITT model, both inbound and outbound messages are transferred using the P3 protocol. However, the actual protocol used is dependent on the message handling system that is in place and whether or not the MTA and the UA are co-resident.

When a Message Store is in use, messages between a UA and the MTA do not have to be handled immediately; they can be stored for later collection or transmission. The basic relationships for a Message Handling System which includes a Message Store are shown in Figure 1.2, "Message Transmission and Retrieval using a Message Store".

Figure 1.2. Message Transmission and Retrieval using a Message Store



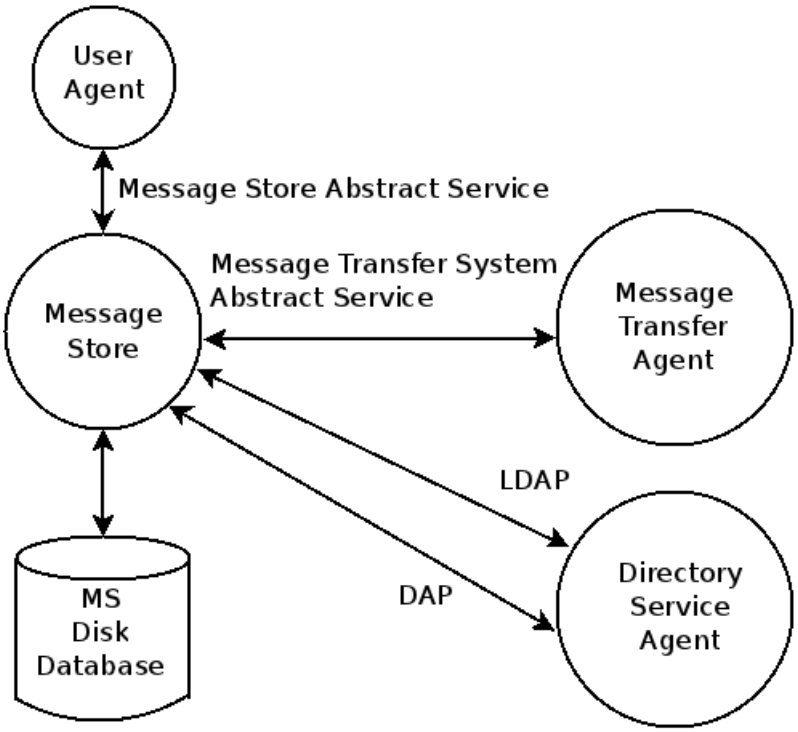
In the CCITT model for implementation of a Message Store, only one protocol is used for communication between the UA and the Message Store - the P7 protocol.

A Message Store has a one-to-one relationship with a UA; that is, a Message Store must be set up for each UA. There is no restriction on the number of Message Stores that can co-exist.

Messages to and from a UA are stored in an associated database by the Message Store until they are ready for collection or transmission. There is one Message Store database per UA. Each database forms an integral part of the Message Store.

The relationships between the Message Store implementation, its databases and the MTA and UAs are shown in Figure 1.3, “Message Transmission and Retrieval”. Refer to Section 1.2, “Services used by the Message Store” for more information on the services used.

Figure 1.3. Message Transmission and Retrieval



1.2 Services used by the Message Store

Four services are used by the Message Store:

- Message Store Abstract Service for communication with User Agents.
- Lightweight Directory Access Protocol (LDAP) for communication with the Directory for User Agent authentication.
- Directory Access Protocol (DAP) for communication with the Directory for message index access.
- Message Transfer System Abstract Service for communication with the MTA.

The first three enable the Message Store to provide services to User Agents and the fourth is used by the Message Store in the provision of those services.

In addition, the Message Store uses direct file access when manipulating messages stored on disk.

Chapter 2 Management

Routine management of the message store is described, including using MConsole as a management tool and describing routine maintenance tasks.

2.1 Message Store Server

The Message Store consists of a single server process (isode.pumice) which allows P7 User Agents (UAs) to access the Message Store (performing the **List**, **Summarize**, **Fetch**, **Delete** and **Register-MS** operations) and to (indirectly) submit messages and probes to a Message Transfer Agent (MTA). This server process is also used for remote (P3) delivery of messages and reports to the Message Store by the Message Transfer Agent. The server process also supports a completely separate management protocol (using sockets-based communication, rather than the OSI stack), which allows remote management and monitoring of the Message Store.

Each mailbox supported by the Message Store contains ‘inbox’ and ‘outbox’ folders, which are used to store delivered and (optionally) submitted messages. Each message is stored as a single binary file which includes both the envelope and content of the message.

The Message Store uses the X.500 Directory to maintain an index entry for each message within the Store database. The index entry is used to maintain the **ms-entry-status** attribute for each message, indicating whether it is a new message or has been listed or processed (fetched), and to provide fast and efficient searching and filtering. The Directory is also used to store various per-mailbox message counts and other items of management information across Message Store restarts. The Directory Access Protocol (DAP) is used for all of this functionality.

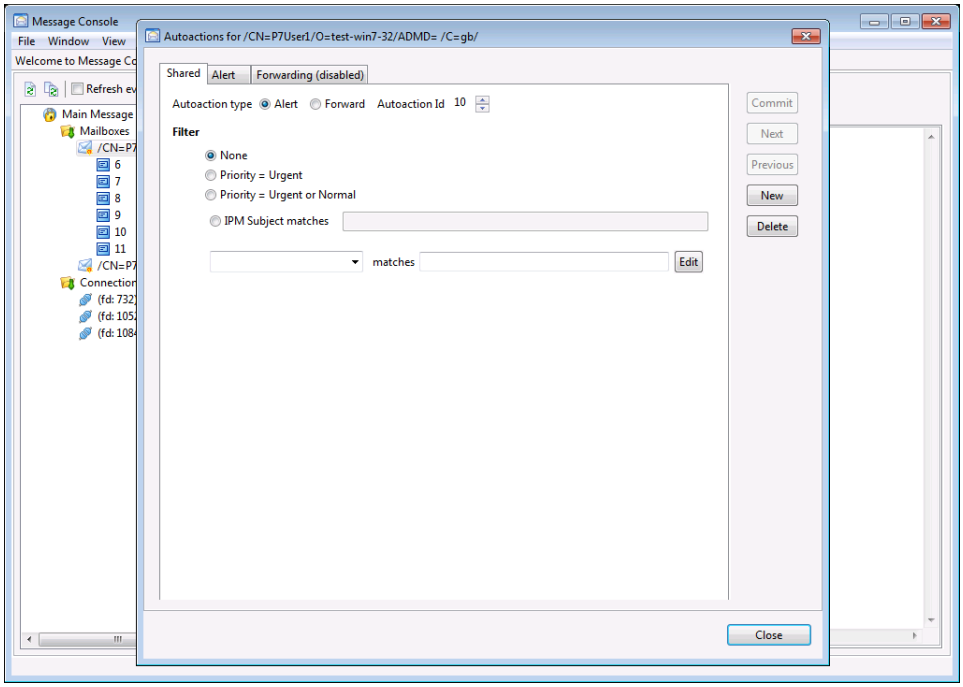
In addition, the Message Store uses the Directory for User authentication, accessing information which is set up using MConsole. Refer to the *M-Switch Administration Guide* for information on MConsole. This authentication information is accessed using the Lightweight Directory Access Protocol (LDAP).

Refer to Figure 1.1, “Message Transfer without a Message Store” for a schematic of the relationships.

2.2 Message Store Console

The isode.pumice server process provides a management and monitoring port in addition to the P7 protocol access described above. A graphical monitoring and management tool for the Message Store which accesses this port is provided as the **X.400 Message Stores** view within the MConsole tool. An example of this view is shown below.

Figure 2.1. Example X.400 Message Stores View



When connected to a Message Store in “unauthenticated” mode, only monitoring functions are available. These allow information about the Store as a whole, mailboxes, messages and active connections to be displayed. If an “authenticated” connection is made, specifying the Store Manager ID and password values configured for the Message Store via the **Switch Configuration** view in MConsole, management functions become available as well. These allow:

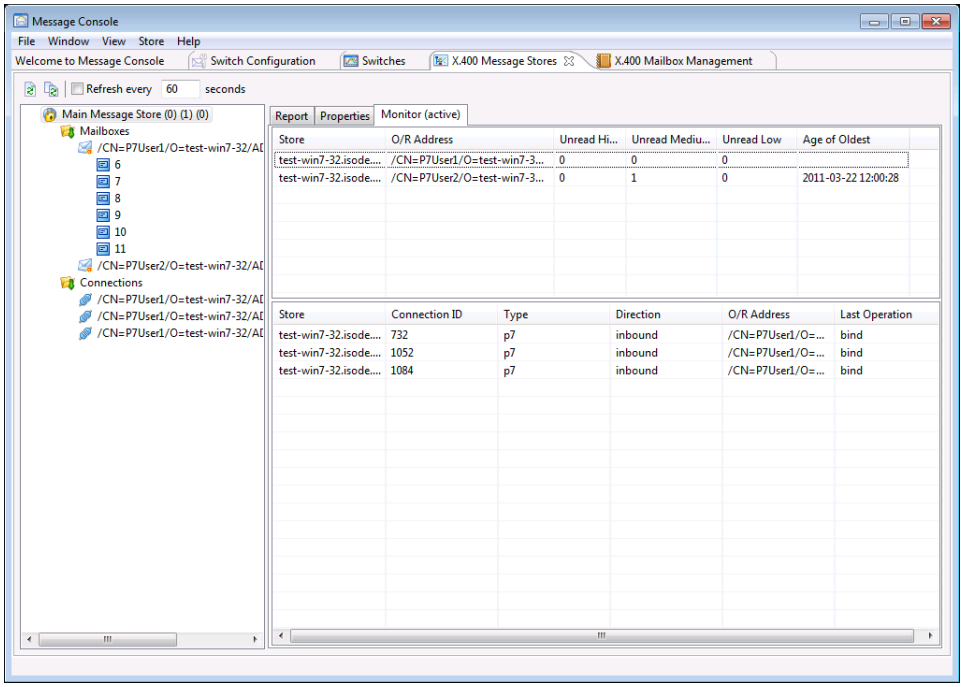
- The Message Store to be stopped, aborted or restarted.
- Specific operations to be blocked or unblocked.
- Active connections to be disconnected.
- Mailboxes to be deleted, backed-up or restored from backup.
- Store-wide statistics to be regenerated.
- Resynchronization of the Message Store’s Directory-based index with the messages in the users’ mailboxes.
- Messages to be deleted, moved between mailboxes or have their status changed.

Multiple Message Stores may be monitored and managed by a single instance of MConsole.

2.2.1 Monitoring tab

A **Monitor** tab is provided by MConsole’s **X.400 Message Stores** view. When monitor mode is enabled for a Message Store (via the **Start Monitoring** menu option on the Store object), the Message Store process informs MConsole whenever a connection is opened or closed, and whenever a message is delivered, read or deleted. This allows a real-time display of the current connections and status of mailboxes to be maintained.

Figure 2.2. Monitor Tab



2.2.1.1 Registering Autoactions

Autoactions are a feature of the service provided by an X.400 Message Store to clients through the P7 protocol. Isode release 11.2 and onwards added a range of management features around autoactions to enable Isode customers to make effective use of this feature.

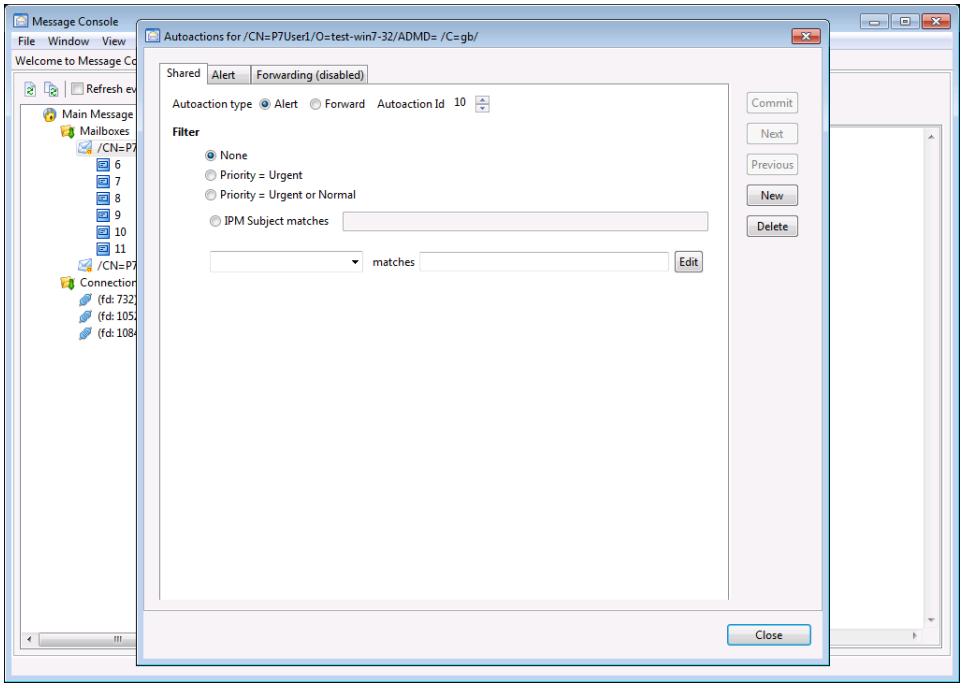
There are two autoactions that can be configured by the **X.400 Message Stores** view within MConsole: AutoForwarding and AutoAlert.

2.2.1.1.1 AutoForwarding

The AutoForward action forwards a delivered message to one or more recipients specified in the autoaction, and marks the message as AutoForwarded (an X.400 feature). It may be set to include a cover note. There is an option in AutoForward to delete the message after it is AutoForwarded. AutoForwarding is useful to ensure that messages get processed when a user is away for a period. It can also be helpful to distribute messages to multiple recipients (essentially an alternate approach to using distribution lists), which allows more user control. This will be desirable for some AMHS users, as this model reflects the way that a lot of AFTN systems operate.

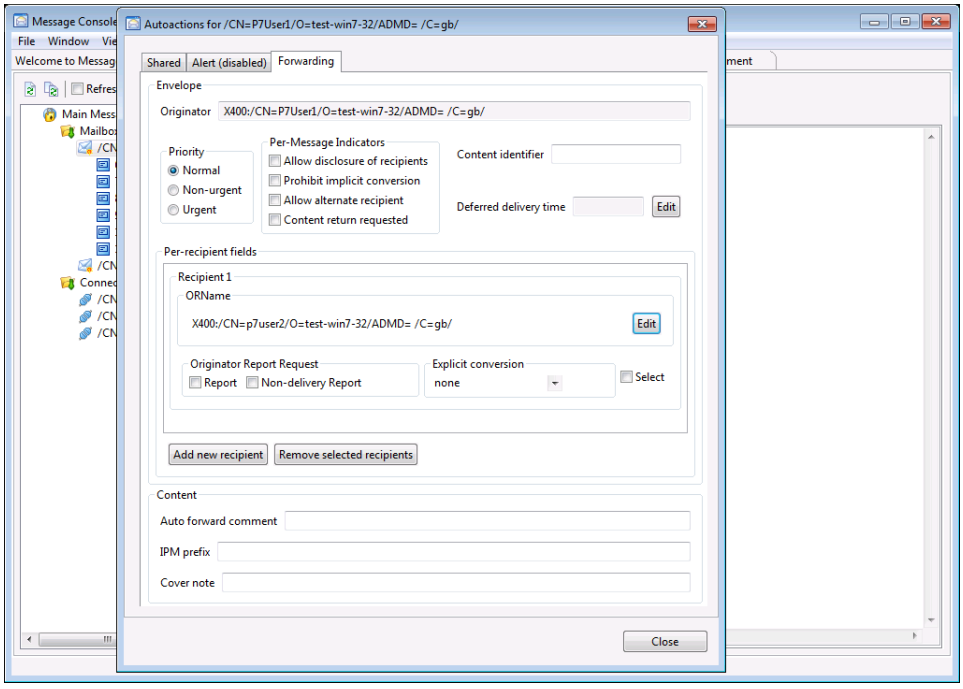
To set it in MConsole, connect to a Message Store in authenticated mode, select the mailbox user that you want to edit, right click on it and select **Configure autoactions**. A window similar to the one illustrated in Figure 2.3, “Configuring Autoactions” will appear.

Figure 2.3. Configuring Autoactions



Click on the **New** button and then change the radio button to **Autoforwarding**. The **AutoForwarding** tab will now be activated. If you select this tab, you will see the window illustrated in Figure 2.4, “Configuring AutoForwarding”.

Figure 2.4. Configuring AutoForwarding



Then click on **Add new recipient** and enter the O/R address of the user which will receive the forwarded messages in the **Address** field.

There are many options that you can set for the new message that is generated, such as setting its **Priority**, **Content Identifier** and **Per-Message Indicators**. If you do not set a value within the **Autoaction Registration**, the corresponding value from the message which is being forwarded will be used.

Don’t forget to click on **Commit** to save the changes that you have made.

2.2.1.1.1 Delayed auto-forward

A variant of AutoForward, which was defined for the US Defense Messaging System, is to perform the AutoForward in the event that the message has not been fetched from the message store after a defined period (i.e., has not been read by the intended recipient). This mechanism can be used to help ensure that messages get read promptly. The filter mechanism can be useful here, for example to have a shorter AutoForward delay for urgent messages.

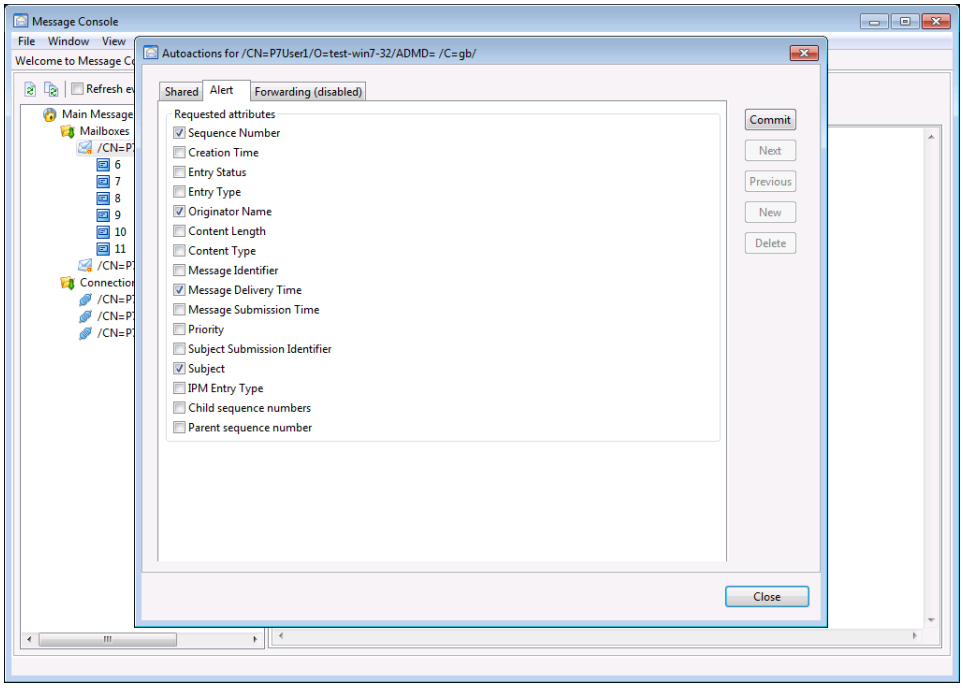
2.2.1.1.2 AutoAlerts

Most P7 operations are client initiated, and normal P7 operation is for the client to poll the server at intervals to check for new messages. The AutoAlert autoaction causes an alert operation to be sent from the server to the client, which provides information on a newly arrived message. Use of AutoAlert allows a client to get new messages more quickly, and improve efficiency by reducing the frequency of polling the store.

AutoAlert autoactions are normally registered by the User Agent, but they can also be registered manually using MConsole.

To register the AutoAlert autoaction, click on the **New** button and then select the **AutoAlert** tab; you will see the window illustrated in Figure 2.5, “Configuring AutoAlert”.

Figure 2.5. Configuring AutoAlert



You can choose which attributes to request in the alert. The ability to configure the attributes requested in an alert is currently not supported by the X.400 Client API’s RegisterMS interface, so until this is available, you can configure the attribute set using MConsole if required.

Don’t forget to click on **Commit** to save the changes that you have made.

2.3 What to run each night

Here are some procedures things you should perform regularly, ideally each night.

2.3.1 Backup the mailboxes

You should regularly back up the mailboxes to another disk, or even to another system.

2.3.1.1 Message files only

The script *(SBINDIR)/mstore_backup.tcl* does the backup of one or all the mailboxes, making sure that there are no conflicts even if a message is about to be delivered to or deleted from a mailbox. This script needs to contact the Message Store, so the service needs to be running. If the Message Store is not running, then you can copy the mailboxes simply with **cp -pr** or a similar command.

```
(SBINDIR)/mstore_backup.tcl
Usage: mstore_backup.tcl -user <user>
-password <password> -destination <dest>
```

`-host` *<host>*
: The host name of the machine where the Message Store is running

`-port` *<port>*
: The port number which the Message Store is using for management connections, it defaults to `3002`.

`-user` *<user>*
: The user name to use to connect to the Message Store management. Usually pp or msadmin.

`-password` *<password>*
: The password to use to authenticate the above user.

`-verbose`
: Output more information about the mailboxes that are being backed up.

`-destination` *<dest>*
: The destination directory under which all the mailboxes will be backed up. It must exist.

`-all`
: Choose to back up all the mailboxes.

`-mailbox` *<mbx>*
: Choose to back up only the mailbox associated with a specific O/R address. You can specify the `-mailbox` flag multiple times on the command line, to select several mailboxes for backup.

Restoration of messages backed up in this way is covered in Section 2.4.1.1, “Restoring message files only”.

2.3.1.2 Mailboxes and Message Store index

An alternative approach (new for R15.0) is to back up both the contents of the mailboxes and the corresponding index area in the DSA. The mailbox backup should be performed using a standard system tool such as **tar**, ideally while the Message Store server is stopped. The backup of the DSA can be performed using one of the techniques described in the

M-Vault Administration Guide. Restoration of data backed up in this way is described in Section 2.4.1.2, “Restoring message files and indexes”.

There are several benefits to using this approach:

- Message sequence numbers are preserved.
- Message status (new/listed/fetched) is normally preserved.
- Message Store downtime (i.e. the time taken to set up a new mailbox filesystem and reload messages from backup, restore messaging configuration and message indexes and get the Message Store back to a position where users can connect and start sending and receiving messages) is likely to be significantly less than using the “backup and restore messages only” approach.

2.3.2 Remove old messages from the mailboxes

Another important task that should to be performed regularly is removing old messages from the Message Store mailboxes. This not only prevents filling up the disk, but it also allows the Message Store to perform delivery more efficiently.

The script *`(SBINDIR)`/mstore_tidy.tcl* can be run to delete messages that are older than a certain age (in days).This script needs to contact the Message Store, so the service needs to be running.

```
(SBINDIR)/mstore_tidy.tcl
Usage: mstore_tidy.tcl [-host <host>] [-port <port>] [-age <age-in-days>]
-user <user> -password <password> [-verbose] [-state <messagestate>]
[-class <entryclass>] [-mailbox <mailbox>]
```

`-host` *`<host>`*
: The host name of the machine where the Message Store is running. Defaults to `localhost`.

`-port` *`<port>`*
: The port number which the Message Store is using for management connections; it defaults to `3002`.

`-user` *`<user>`*
: The user name to use to connect to the Message Store management. Usually pp or msadmin.

`-password` *`<password>`*
: The password to use to authenticate the above user.

`-verbose`
: Output more information about the mailboxes that are being backed up.

`-age` *`<days>`*
: The maximum age in days of messages to leave in the mailbox. All older messages will be deleted. Defaults to 7 days.

`-state` *`<state>`*
: Only tidy up messages with state `new`, `listed`, `processed` or `any`. Defaults to `any`.

`-class` *`<entryclass>`*
: Only tidy up messages of entryclass `submitted` (i.e. in the Outbox), `delivered` (in the Inbox) or `any`. Defaults to `delivered`.

`-mailbox` *`<mbx>`*
: Allows a single mailbox to be processed, rather than applying the operation to all mailboxes. The mailbox is specified using its complete O/R Address.

2.3.3 Save the old logs

Saving logs depends very much on the individual policies for each site. If you are concerned about keeping the information, you should backup the logs each night. In any case, after a sensible period (e.g. a week) logs should be removed from the *(LOGDIR)* directory.

The `xms-audit` log output can be processed to show statistics.

2.4 Troubleshooting

2.4.1 Restoring mailboxes from a backup

2.4.1.1 Restoring message files only

The script *(SBINDIR)/mstore_restore.tcl* restores one or all the mailboxes which have been backed up using *(SBINDIR)/mstore_backup.tcl*, making sure that there are no conflicts even if a message is about to be delivered to or deleted from a mailbox.

This script needs to contact the Message Store, so the service needs to be running.

```
(SBINDIR)/mstore_restore.tcl
Usage: ./mstore_restore.tcl -user <user> -password <password>
[-host <host>] [-message <msg> | -source <src>]
[-port <port>] [-all | -mailbox <mbx>] [-verbose]
```

`-host` *<host>*
: The host name of the machine where the Message Store is running. Defaults to `localhost`.

`-port` *<port>*
: The port number which the Message Store is using for management connections, it defaults to `3002`.

`-user` *<user>*
: The user name to use to connect to the Message Store management. Usually pp or msadmin.

`-password` *<password>*
: The password to use to authenticate the above user.

`-verbose`
: Output more information about the mailboxes that are being restored.

`-source` *<src>*
: The source directory under which all the mailboxes are backed up.

`-message` *<msg>*
: The specific message that has to be restored.

`-all`
: Choose to restore all the mailboxes

`-mailbox` *<mbx>*
: Choose to restore only the mailbox associated with a specific O/R address. You can specify the -mailbox flag multiple times on the command line, to select several mailboxes for restoration.

This method of backing up and restoring messages is intended for use when the corresponding Message Store index entries in the DSA have been lost - the restoration

process will create new index entries. In the process message sequence numbers will be reallocated, and all messages will be returned to the “new” state.

2.4.1.2 Restoring message files and indexes

If you have chosen to back up the contents of the Message Store’s mailboxes using **tar** or a similar system tool, and have also backed up the corresponding Message Store index entries from the DSA, then it is possible to restore these two sets of information and resynchronize them. The procedure to follow is:

1. Prepare a new filesystem for the mailboxes and DSA database, if necessary.
2. Restore the DSA database from backup - refer to the the *M-Vault Administration Guide* for details on how to do this.
3. Restore the Message Store mailboxes from backup, e.g. using **tar**.
4. Ensure that you have a valid *pumicetailor.xml* file in *(ETCDIR)*; you may need to run MConsole to recreate this file if it has been lost.
5. Start the Message Store executable with the `-r` command line switch. The Message Store will be available for User Agents to connect to and start performing P7 operations, but a background resynchronization process will also be initiated. The background task will examine the set of messages in each mailbox, and compare this with the corresponding set of index entries in the DSA. New index entries will be created for any messages which are found to be missing them, and any index entries which do not have a corresponding message file (i.e. “orphan” entries) will be deleted.

The resynchronization process can also be triggered at any time using the SOM protocol: MConsole provides a **Resynchronize all mailboxes** option for a Message Store in **Message Store View** which enables you to trigger this.

2.4.2 Message Store sequence numbers

Each message created in a given mailbox (whether stored on submission or delivered) is assigned a sequence number. The maximum sequence number is (2**31 -1), i.e. over 2,000,000,000. There is no requirement in the P7 protocol for the sequence numbers to be assigned sequentially, only that they are in ascending order. There are situations in normal Message Store operation where (apparent) gaps in sequence numbering may appear - for example, if you have configured an autoforwarding auto-action which is set to delete the original message after forwarding.

When a backup/restore operation has taken place, message sequence numbers may appear to be reused, as all knowledge about the sequence numbers assigned to the messages which were delivered between the backup and restore operations will have been lost. This may cause problems for User Agents which “remember” the highest sequence number already fetched.

To assist in this situation, the Message Store can be started with a command line argument of `-i <increment>`. This will cause the Message Store to increment the “starting point” for each mailbox’s sequence numbers by the specified amount, before normal operations are enabled.

2.5 Obfuscation of passwords

The *pumicetailor.xml* file which configures how the Message Store connects to the Directory can contain passwords. These can be obfuscated using the Service Key facility. To do this:

- Use MConsole to create a Service Key for your Message Store. On Unix platforms, you will need to be running MConsole as root. In all cases, you need to be running MConsole on the system on which your Message Store will actually run.

  To create a Service Key, open the **Switch Configuration View** and select the Message Store for which you wish to create the Service Key. The right-mouse menu for this object includes options for creating and deleting Service Keys. You will need to enter a passphrase, which will be used to encrypt the Service Key, and (on Unix only) specify the userid under which the Message Store will run.
- Use MConsole to create a new *pumicetailor.xml* file for your Message Store (via the same right-mouse button as used to create the Service Key). If you have created a Service Key for the Message Store, you will be prompted to enter the passphrase which you entered in the previous step, and passwords will be encrypted using the Service Key before they are written into the pumicetailor.xml file. A password which has been encrypted in this way is prefixed with *{spcrypt}* when stored.
- On Unix, the Service Key will be stored in a file in *`$(ETCDIR)/servpass`*. Standard Unix file protection is used to limit read access to the userid under which the Message Store runs (e.g. root). This will mean that any standalone tools which also read pumicetailor.xml, such as ms_dump, will need to be run under the same userid - otherwise they will not be able to use the information in the Service Key to decrypt the passwords in *pumicetailor.xml*.
- On Windows, OS-specific mechanisms are used to protect the Service Key, and the restrictions on the userid do not apply.

2.6 Disaster recovery

There are several Message Store configurations which can be set up in order to provide resilience against hardware or system software failure.

2.6.1 Fail-over clustered configuration

In a fail-over clustered configuration, two (usually identical) computers have access to a single shared disk, which is used to hold the MTA's message queue, the Message Store's mailboxes and the DSA's database. One (primary) provides the service in normal situations. A second (failover) computer is present in order to run the service when the primary system fails. The primary system is monitored, with active checks every few seconds to ensure that the primary system is operating correctly. The system performing the monitoring may be either the failover computer or an independent system (called the cluster controller). In the event of the active system failing, or failure of components associated with the active system such as network hardware, the monitoring system will detect the failure and the failover system will take over operation of the service.

2.6.2 Hot standby Message Store configuration

There are situations where it is desirable to provide a hot standby Message Store which is physically separated from the primary system, and which does not rely on sharing a disk with the primary. There are three main problems to be addressed with this approach:

1. How to give the standby system access to the same per-user configuration information as the primary system.
2. How to manipulate the information used by the associated MTA to perform delivery, so that the MTA switches delivery from the primary Message Store to the standby system.

3. How to load messages which have been delivered into the standby Message Store back into the primary system, once it has been brought back on line.

2.6.2.1 Primary system configuration

No special configuration is required for the primary Message Store: its configuration is created as normal using MConsole, as part of a Messaging Configuration which will also contain an X.400 M-Switch MTA configuration and the standard per-user configuration. The Message Store executable itself could run on the same system as the DSA and MTA, but would more sensibly be located on a separate machine together with the file system used for its mailboxes.

2.6.2.2 Standby system configuration

The standby Message Store should be configured in the same way as the primary system. It must be configured with its own MS Index Root.

Once configured, the standby Message Store can be started if desired, or left stopped.

2.6.2.3 User configuration

X.400 Message Store users should be configured as normal, using MConsole, with the primary specified as their Message Store. Use of relative mailbox paths is recommended, otherwise it will be necessary to have identically-named filesystems on the primary and standby systems.

2.6.2.4 Switchover from primary to standby

Switchover from the primary Message Store to the standby is performed by using the **Redirection** control on the primary Message Store's editor within MConsole's **Switch Configuration** view. Selecting the standby Store as the target of the **Redirect to Message Store** control and pressing **Apply** will have the following effects:

- All messages for Message Store recipients which are currently in the MTA's queue (excluding those which are queued on P3 delivery channels which are currently bound to the primary Message Store) will be delivered into the standby Message Store.
- The standby Message Store will accept binds from all of the X.400 Message Store users configured for the primary Message Store. Users will be given access to an empty mailbox into which new messages will be delivered, and will be able to submit messages as normal.

Message Store servers check for a redirection being set or cleared on startup and every 60 seconds subsequently. This means that a maximum interval of 60 seconds will elapse between performing the switchover using MConsole and the standby Message Store starting to accept connections from User Agents and MTAs.

2.6.2.5 Restoring messages from standby after recovery

Once the primary Message Store has been restarted, the following steps must be followed:

1. Clear the redirection, using MConsole as described above. This will mean that messages start being delivered into the primary Message Store again, and the standby Message Store will stop accepting new connections. User Agents which are already bound to the standby Message Store will continue to be able to read and submit messages.
2. Shut down the standby Message Store, forcibly disconnecting any User Agents which are bound to it. The User Agents should rebind to the primary Message Store.
3. Run the **ms_dump** utility on the standby Store system. This will produce a backup file tree which can then moved to a suitable temporary location on the primary system.

   The **ms_dump** utility takes the following flags:

`-o <output folder>`
specifies where the backup file tree should be created.

`-t <pumicetailor file>`
allows a *pumicetailor.xml* other than the default one to be used.

`-v`
switches on verbose mode

`-p`
overrides the default pagesize of 100 entries when reading from the Directory

`-r <DN>`
specifies the Directory Name of the Message Store from which the standby was redirected - in other words, the DN of the primary Message Store.

`-c`
specifies that the standby Message Store should be ‘cleaned’ as the dump is performed - i.e. all messages and corresponding index entries should be deleted.

The backup file tree will include status information about each message, indicating whether it is new, listed or fetched.

4. After copying the backup tree to a suitable temporary location on the primary system, its contents can be imported to the primary Message Store. In MConsole’s **X.400 Message Stores** view, select the primary Message Store and choose **Import mailbox dump** from the right-mouse-button menu. You will be prompted to enter the location of the backup tree. The Message Store server will then import the messages in the dump into its database, making them available to User Agents. The new/listed/fetched status of the messages will be preserved, although they will be assigned new sequence numbers during the import process. The import runs “in the background” and will not interfere with normal P7 operations.

Chapter 3 Configuration

Configuration of all elements of the Message Store, including the configuration of logging, is described in this chapter.

3.1 Message Store tailoring

A tailoring file, *pumicetailor.xml*, located in `(ETCDIR)`, provides the configuration information that the isode.pumice process need to be able to access the Directory. An example file is shown in Example 3.1, “Example pumicetailor.xml File”. In releases prior to R14.6, this file was called *pumicetailor*, and was structured as `key:value` pairs. The R15.0 isode.pumice will still read this old style of configuration file if found, although a warning message will be logged.

Configuration of various operational aspects of the Message Store is held in the Directory, and can be modified using MConsole. This is described in the *M-Switch Administration Guide*. MConsole can also be used to create *pumicetailor.xml* files which contain the correct connection information for individual Message Stores.

Section 3.1.1, “Tailoring elements” describes each component of the file.

3.1.1 Tailoring elements

The tailoring elements fall into two groups: those which are always applicable and those which apply to standard Isode tailoring variables.

3.1.1.1 General tailoring

`bind_name`
: This holds the Directory Name with which the isode.pumice process will bind to the Directory.

`bind_password`
: OPTIONAL. The password which the isode.pumice process will use when binding to the Directory. An anonymous bind will be attempted if no value is supplied.

`p7server_name`
: This holds the Shared Message Store’s own Directory Name. This will normally be the same as the value for `bind_name`.

`dsa_address`
: This holds the Presentation Address of the Directory’s DAP listener. If not specified, it defaults to `"localhost"` on port `19999`.

`single_p7_bind`
: Configures whether the Message Store will allow multiple concurrent P7 Binds using the same mailbox. The default setting is to allow multiple binds: to prevent this, set the value to `"true"`.

3.1.1.2 Isode tailoring

Isode base library general variable tailoring is available from the Message Store’s tailoring file as follows:

`isode`
: allows any Isode tailoring variable to be set. Overrides any value configured in the *isotailor* file. For example, the tailoring element: `<isode`

`name="etcpath">/var/etc</isode>` would set the Isode tailoring variable `etcpath` to value `"/var/etc"`.

Two specific Isode tailoring elements will need to be set:

`<isode name="mhsds_ldap_port">19389</isode>`
: This configures the port on which the Message Store will attempt to contact the LDAP server. You will need to modify this if your LDAP server is listening on a different port. If not configured, this will default to `"389"`.

`<isode name="mhsds_ldap_host">localhost</isode>`
: This configures the hostname on which to contact the LDAP server. You will need to modify this if your LDAP server is on a different system from your Message Store.

Example 3.1. Example pumicetailor.xml File

```
<pumicetailor>
  <servpass:info service="isode.pumice"/>
  <bind_name>&lt;cn=Main Message Store,cn=Messaging
Configuration,o=Isode Limited,c=GB&gt;</bind_name>
  <bind_password>secret</bind_password>
  <p7server_name>&lt;cn=Main Message Store,cn=Messaging
Configuration,o=Isode Limited,c=GB&gt;</p7server_name>
isode name="mhsds_ldap_host">localhost</isode>
  <isode name="mhsds_ldap_port">19389</isode>
</pumicetailor>
```

The `servpass` element in the example above is used by the Isode ServPass API. This provides a mechanism which allows encrypted passwords to be used in configuration files such as *pumicetailor.xml*, instead of storing the password 'in the clear' as has been done in this example. For details on how to make use of this facility, please refer to Section 2.5, "Obfuscation of passwords".

3.1.1.3 Tailoring for use of separate index DSA

The default configuration for M-Store is to use the same DSA to hold both the Message Store Index and Messaging Configuration. If you wish to use a separate DSA to hold the Index area, the following extra configuration items must be added to the *pumicetailor.xml* file:

`index_dsa_address`
: The Presentation Address of the Index DSA's DAP listener.

`index_bind_name`
: The Distinguished Name to be used when binding to the Index DSA.

`index_bind_password`
: The password to be used when binding to the Index DSA.

`index_ldap_host`
: The hostname of the Index DSA.

`index_ldap_port`
: The port on which the Index DSA is listening for LDAP connections.

Note: Use of a second DSA in this manner can only be configured via the *pumicetailor.xml* file - there is no mechanism for setting it via MConsole.

3.2 Message Store logging

This section gives an overview of the general structure of the Isode logging subsystem, and a description of the GUI which is provided to enable configuration of the Message Store's use of this subsystem.

3.2.1 Getting started

If you wish to modify the default logging settings for the Message Store, you should:

1. Copy the file *xmslogging.xml* from *(SHAREDIR)* into *(ETCDIR)*.
2. Run the GUI:

   On Windows, a shortcut to the **Log Configuration Tool** will have been set up in the **Isode** folder on your **Start** menu.

   On Unix, you should run */opt/isode/sbin/logconfig*.
3. Once the GUI is running, open *xmslogging.xml* from *(ETCDIR)*. You will see a display of a number of predefined logging streams used by the Message Store, which can be modified as required. For full details of the options available, see Section 3.2.3.1, "File streams".

3.2.2 How logging works

3.2.2.1 Record types

Isode server programs (like the isode.pumice process) write two types of log records during normal execution - Audit records and Event records.

Audit records are used to record "auditable events" - message submission, for example. They do not have a severity level associated with them, and have a well-defined format, so that they can be easily parsed. Audit records normally consist of an event-type indicator, followed by a list of `key=value` pairs.

Event records are used to record errors, normal program operation, or to provide debugging information. They are associated with a particular severity level, and contain freeform text with substituted data items. The freeform text is contained in a separate dynamically-loaded library (on Windows) or a message catalog (on Unix), which makes it possible to replace the standard set of English messages with equivalent text in other languages simply by substituting a suitable message file.

No output mechanism is directly associated with log records. When an event or audit record is generated by an application, then whether or not it is logged, where it is logged to, and what the output of the log looks like, depends on what output streams have been configured.

3.2.2.2 Output streams

An output stream is a description of how a particular set of event and audit records should be recorded or displayed. Multiple output streams may be configured for an application, and whenever an event or audit record is generated, the logging subsystem checks to see which, if any, of the available output streams is eligible to process it.

As well as defining which records are eligible to be logged, the configuration of an output stream also determines the format of the messages that are produced by the stream.

This means that a single event or audit record may be processed by one or more separate streams (or by no stream at all), and that, in the case of multiple streams, the messages output by the streams may be of differing formats, containing more or less detail. For example, it would be possible to configure one output stream to generate a brief message about all "warning" level events, and another to generate a detailed message about a specific "warning" event which is of particular interest.

Four stream types are currently available: the `file` type, where the records are output to a file, the `system` type, where the records are passed to the system event log (syslog on Unix-type systems and the Application Event Log on Windows), the `tty` type, which is identical to file type, except that the records are written to either `stdout` or `stderr`, and the `mpp` type, which sends records to the Isode Server Watch Daemon using either a Unix named pipe or via TCP.

3.2.2.3 Configuration storage and loading

Information about output stream configuration is stored as XML. All Isode applications will load the XML contained in the file *logtailor.xml*, located in *`(ETCDIR)`* or *`(SHAREDIR)`*, if it exists, at startup. This filename and location can be overridden if required by defining the environment variable `LOGTAILOR` to be an alternative filename or filepath.

An application may then load a private stream configuration. In the case of the Message Store, this is contained in the *xmslogging.xml* file. A default version of this file is located in *`(SHAREDIR)`* - if you wish to make changes, copy this file into *`(ETCDIR)`* and modify this version. If the configuration file exists in both *`(ETCDIR)`* and *`(SHAREDIR)`*, the version in *`(ETCDIR)`* will be used.

3.2.2.3.1 Format of messages in output streams

When a given audit or event is generated, then for each output stream that is configured to process records of that type, the settings for the output stream determine the format of the message that is output. In the case of file and tty streams, the stream may be configured to contain any combination (including none) of the following fields:

date and time
: The format of date and time is configurable on a per-stream basis

program name
: The name of the program generating the message. Any "isode" prefix will have been removed, and the program name will be truncated to 8 characters

process id
: Identifies the process

thread id
: This field may be useful to distinguish separate threads in the same process

username
: The username of the process which generated the record. This field is only meaningful on Unix systems. If the username cannot be established, then a numeric UID is logged.

severity
: Audit records have no associated severity, but event records always have a severity, which, if displayed, is represented using one of the following single letters, as follows:

  - `I` - Info
  - `N` - Notice
  - `S` - Success
  - `D` - Detail
  - `W` - Warning
  - `E` - Error
  - `F` - Fatal

- `C` - Critical
- `L` - AuthOK
- `A` - Authfail
- `X` - Debug P - PDU

facility code
: The name of the facility which generated the message. Audit records are not associated with a particular facility.

message identifier
: An identifier representing the event. Audit records do not have a message identifier

text
: The formatted text describing this event. Audit records do not have a text field

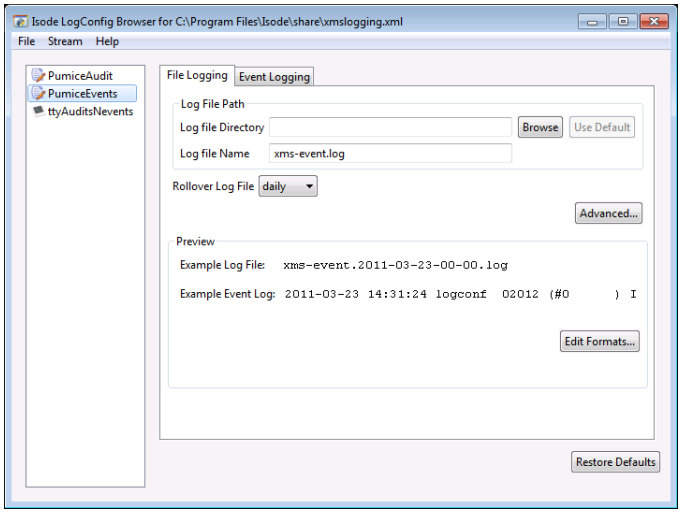
supplementary audit record parameters
: For certain types of audit records, extra information may be associated with the record, and if the stream is suitably configured, this will be included as a sequence of "key:value" pairs on the end of the message.

3.2.3 Logging configuration

3.2.3.1 File streams

For file streams, the following parameters can be configured:

Figure 3.1. File tab



Log file Directory
: If set, overrides the default system-wide logging directory.

Log file Name
: The name of the logging file. Note that if you have rollover configured (see below) the actual filename will in part be derived from the rollover interval.

Rollover Log File
: This allows you to choose between three predefined intervals at which a new logfile will be created. This may be useful, for example, in the event that you wish to purge old logfiles selectively.

3.2.3.1.1 Advanced

This gives access to a popup which allows the file logging to be specified in greater detail:

File Permissions
: By default, log files are created so that any process can write to them. If different process owners are writing to the file, then this is necessary. However, the file mode,

in octal, can be configured here. To set the value so that only the creator can write to it, but others can read, for example, the value set should be `'644'`.

Log to Open File
: This enables you to log to an open file descriptor. The integer value of the file descriptor is set in this field.

Close file after a message is written
: The file is opened for each message and then closed after the message is written. This can be used to ensure that the data is secure but there is a significant performance penalty.

Sync log messages to disk
: The operating system is asked to ensure that the message is written to disk. This can be used to ensure that the data is secure, and there is some performance penalty.

(Windows only) Lock file prior to writing
: The file is locked prior to writing the message, and then unlocked afterwards. This ensures that when multiple processes are logging to the same file that the messages are not mixed. It is only used on Windows, and the default is to lock.

Rollover settings
: You can configure file logging so that a new logfile will be created at regular intervals. You configure rollover by specifying:

  rollover interval
  : This control specifies how frequently a new file should be created. Note that the name of the generated logfile will include the date and time at which it was created (for example *"xms-event.2005-06-13-00-00.log"*).

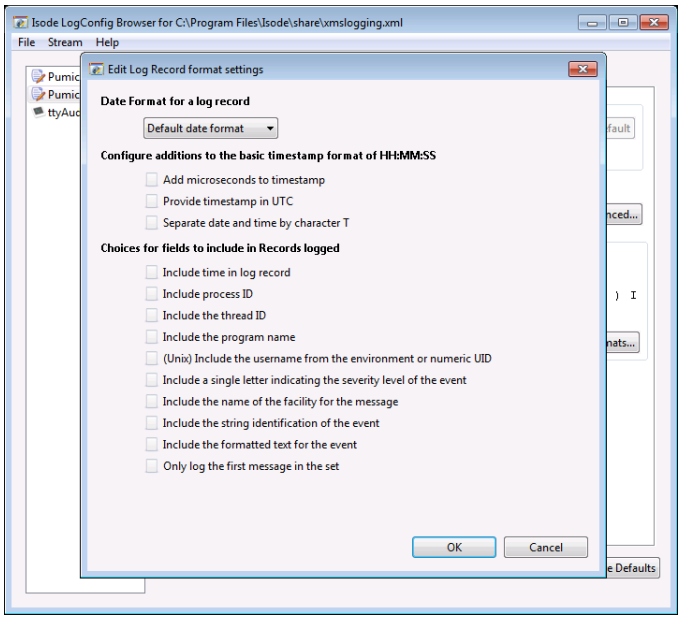
  rollover offset
  : Normally the time interval for rollover coincides with a standard time division. E.g. midnight for intervals of a day or more. The offset enables you to move this start time.

Preview
: the preview area shows you what the filename of the event log file you have defined will look like, and an example of an event log entry which will be written to it.

3.2.3.1.2 Edit Formats

This button brings up an editor which allows you to specify in detail how event log entries will be constructed, as illustrated below:

Figure 3.2. Edit log record format settings



Date Format for a log record
: This configures how the date element of the time-and-date field at the start of each log record is formatted. Available choices are:

  Default date format
  : This gives a date format which is the same as in pre-R15.0 releases - e.g. `" 2/8 14:35:06"`.

  No date field
  : No time or date field

  DD/MM format
  : Day of month and month number, i.e. `"29/12"`

  YY-MM-DD format
  : two-digit year, month and day in YY-MM-DD format

  YYYY-MM-DD format
  : as above, but four-digit year, i.e. `"2011-03-28"`

Time options
: Configure additions to the basic timestamp format of HH:MM:SS. Options are:

  - Add microsecond field to timestamp, so format is HH:MM:SS.UUUUU
  - Record all timestamps in UTC rather than local timezone. There is no difference in output format when this option is set.
  - When used in conjunction with the four-digit year option, provide an ISO standard timestamp, with the date and time fields separated by a “T” character rather than the default of single space, i.e. "12:34:10T2005-06-13".

message fields
: Configure what fields are included in records logged to a file stream. The default setting is for all fields to be included. Choices are:

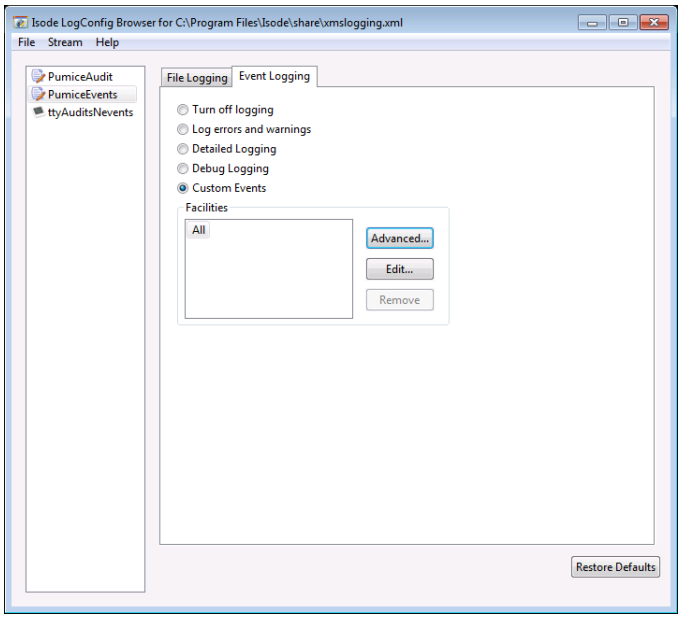
  - Include time and date in log record.
  - Include process ID.
  - Include the thread ID. This enables you to distinguish events logged by different threads within the process.
  - Include the program name. Any "`isode`" prefix is removed, and the program name is truncated to 8 characters.
  - (Unix only) Include the username from the environment. If not available then the numeric UID is logged.
  - Include a single letter indicating the severity level of the event. Letters are:
    - `S` - Success
    - `X` - Debug
    - `P` - PDU
    - `D` - Detail
    - `I` - Info
    - `N` - Notice
    - `L` - AuthOK
    - `W` - Warning
    - `E` - Error
    - `F` - Fatal
    - `C` - Critical
    - `A` - Authfail
  - Include the name of the facility for the message.
  - Include the string identification of the event.

- Include the formatted text for the event
- If a message set is being logged, only log the first message in the set.

3.2.3.2 Event logging tab

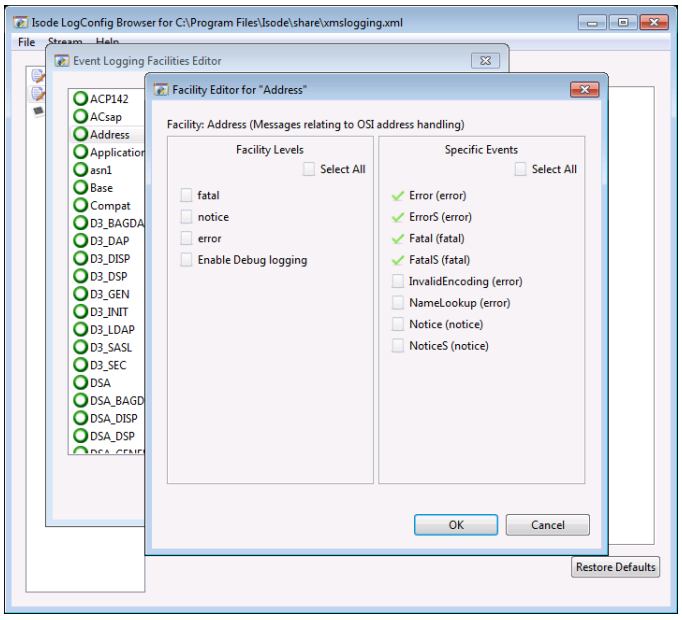
This tab is common to all types of stream, and enables you to configure which event records are to be sent to this stream. Configuration is done by facility. However, the '**All**' facility can be used to set which severity level are logged for all facilities, by default. This setting can then be overridden for each facility. In addition individual messages can be configured to be logged or not logged. Note that **pdu** and **debug** level logging is not available for **system** streams.

Figure 3.3. Event configuration tab



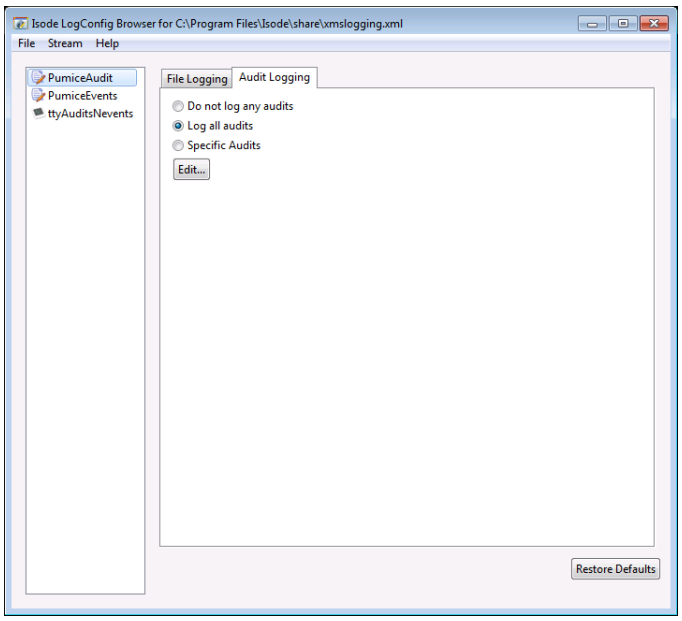
The **Advanced** button provides access to the **Event Logging Facilities Editor**, which allows fine control of messages and logging levels on a per-facility basis.

Figure 3.4. Facility editor



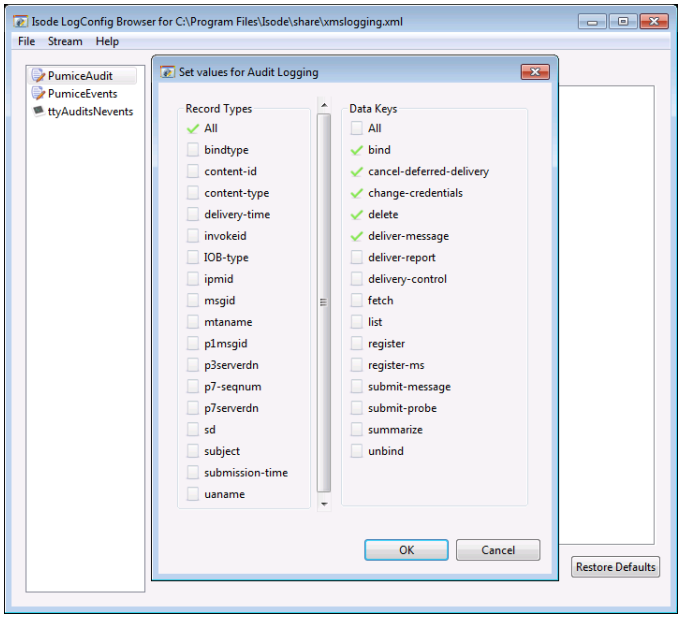
3.2.3.3 Audit logging tab

Figure 3.5. Audit tab



This tab allows a simple choice between logging all audits, no audits or specific audits; for this latter choice, the Edit button produces an editor which allows you to configure which of the available audit record types are to be sent to this stream, and which fields in the records are to be included. The default is for all records and all fields to be included.

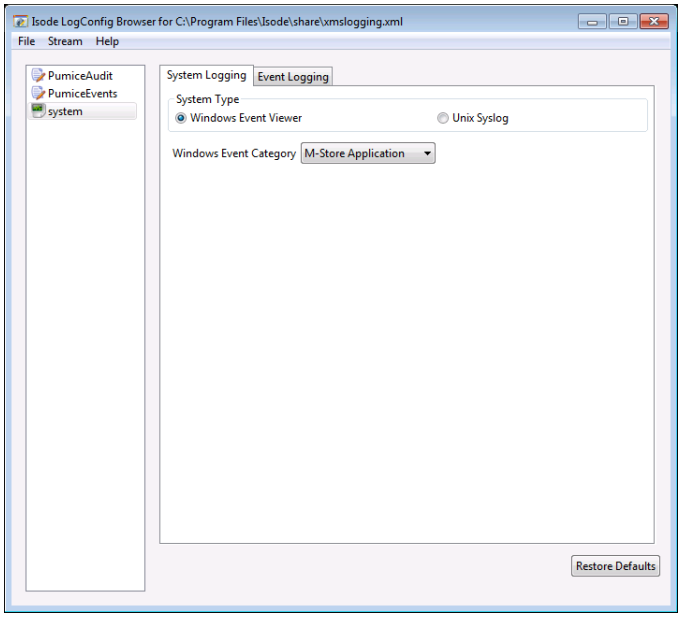
Figure 3.6. Specific audits configuration



3.2.4 System streams

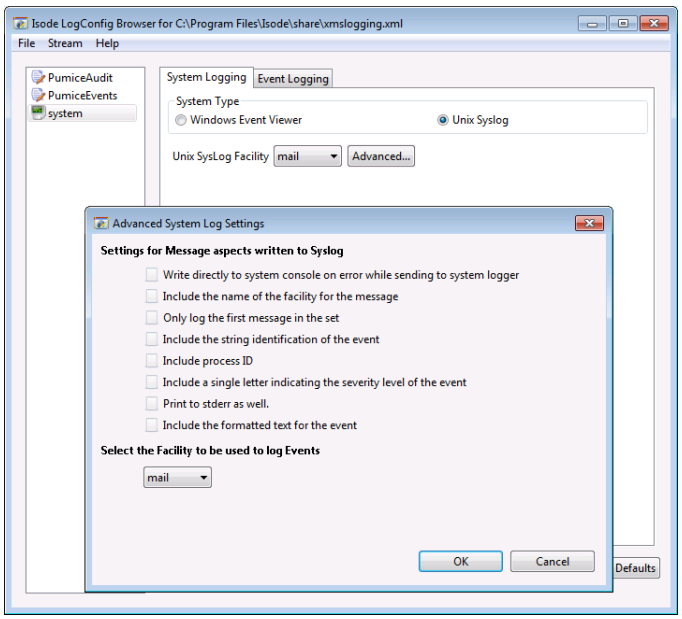
For system streams, the properties which can be configured vary depending on whether you are using the System Event Log on Windows or the Unix Syslog.

Figure 3.7. Properties Tab (Windows)



The **Windows Event Category** selector configures which of the predefined event categories will be used for the event log.

Figure 3.8. Properties Tab (Unix)



Basic configuration of a **System** stream on Unix allows a choice of the SysLog Facility which is to be used.

The **Advanced** button provides access to more detailed configuration options:

- Write directly to system console if there is an error while sending to system logger.
- Include the name of the facility for the message.
- If a message set is being logged, only log the first message in the set.
- Include the string identification of the event.
- Include process ID.
- Include a single letter indicating the severity level of the event.
- Print to `stderr` as well.

- Include the formatted text for the event.

3.2.5 MPP streams

For `mpp` streams, which send event information to the Isode Server Watch Daemon, the only configuration items are:

- The hostname to which events will be sent (on Windows) or the named pipe to which events will be sent (on Unix).
- On Windows only, the port to which events will be sent.

3.2.6 SNMP streams

An SNMP Stream causes event log entries to be written into an SNMP table, which can then be queried by an SNMP Manager application. By default the table size is unlimited, but the stream's configuration allows a limit to be set if wanted. The stream can also be configured to generate an SNMP Trap every time that a new entry is added to the table.

3.2.7 TTY streams

The only configuration item for a TTY stream is a choice of whether the output is directed to `stderr` or `stdout`.

3.3 Message Store audit logging

An Audit record is generated for each operation performed by the Message Store. Each record consists of:

- A keyword which identifies the operation being performed. These are: `BIND`, `LIST`, SUMMARIZE, `FETCH`, `DELETE`, `REGISTER-MS`, `SUBMIT-MESSAGE`, `SUBMIT-MESSAGE-RESULT`, `SUBMIT-PROBE`, `SUBMIT_PROBE_RESULT`, `CANCEL-DEFERRED-DELIVERY`, `DELIVER-MESSAGE`, `DELIVER-REPORT`, `DELIVERY-CONTROL`, `CHANGE-CREDENTIALS`, `REGISTER` and `UNBIND`.
- The identifier of the association performing the operation.
- A sequence of `key=value` pairs which is specific to the type of operation being recorded. These are described below.

3.3.1 BIND operation

`bindtype`
: identifies type of bind operation: value can be `"to MTA"` `"by MTA"` and `"by UA"`.

`p7serverdn`
: the DistinguishedName of the Message Store.

`p3serverdn`
: the DistinguishedName of the p3server entity to which the Store is binding, when `bindtype = "to MTA"`.

`mtaname`
: the name with which the MTA is binding to the Message Store.

`uaname`
: the ORName with which the UA is binding to the Message Store.

3.3.2 SUBMIT-MESSAGE operation

`invokeid`
: an identifier for the operation being invoked on the MTA - this allows correlation with a `SUBMIT-MESSAGE-RESULT` audit record.

`content-type`
: the content type of the message being submitted.

`content-id`
: the `content-id` from the message submission envelope, if present.

`IOB-type`
: For P2/P22 content types, indicates whether an IPM or IPN is being submitted.

subject
: For IPMs, the subject field from the IPM heading.

ipmid
: For IPMs, the this-IPM field from the heading. For IPNs, this will be the `subject-ipm` field.

3.3.3 SUBMIT-MESSAGE-RESULT operation

`invokeid`
: an identifier for the operation which was invoked on the MTA - this allows correlation with a `SUBMIT-MESSAGE` audit record.

`p1msgid`
: the `MTSIdentifier` from the submission result.

`submission-time`
: the time at which the message was submitted.

`content-id`
: the `content-id` from the message submission envelope, if present.

3.3.4 SUBMIT-PROBE operation

`invokeid`
: an identifier for the operation being invoked on the MTA - this allows correlation with a `SUBMIT-PROBE-RESULT` audit record.

`content-type`
: the content type of the probe being submitted

`content-id`
: the content-id from the probe submission envelope, if present.

3.3.5 SUBMIT-PROBE-RESULT operation

`invokeid`
: an identifier for the operation which was invoked on the MTA - this allows correlation with a `SUBMIT-PROBE` audit record.

`p1msgid`
: the `MTSIdentifier` from the submission result.

`submission-time`
: the time at which the probe was submitted.

`content-id`
: the content-id from the probe submission envelope, if present.

3.3.6 DELIVER-MESSAGE operation

`p1msgid`
: the `MTSIdentifier` of the message being delivered

`delivery-time`
: the time at which the message was delivered.

`p7-seqnum`
: the P7 sequence number assigned to the message after delivery.

3.3.7 DELIVER-REPORT operation

`subjectmsgid`
: the `MTSIdentifier` of the message to which the report being delivered pertains.

`content-id`
: the content-identifier field from the report.

`p7-seqnum`
: the P7 sequence number assigned to the report after delivery.

3.3.8 CANCEL-DEFERRED-DELIVERY operation

`msgid`
: the `MTSIdentifier` of the message whose delivery is being cancelled.

3.3.9 FETCH operation

`p7-seqnum`
: the P7 sequence number of the message being fetched.

3.3.10 DELETE operation

`p7-seqnum`
: the P7 sequence number of the message being deleted.

3.3.11 Other operations

None of the other operations generate operation-specific `key=value` pairs.

3.4 User agent configuration

Your User Agent will need to be configured with the Presentation Address on which the isode.pumice process is listening. The default value for this is `"3001"/Internet=<`*`IPaddr`*`>+3001` - i.e. the isode.pumice process will listen on port 3001 for incoming connections which specify a Transport Selector of "3001" (text). Note that the Message Store will only listen on the specific IP address for which it is configured.

3.5 Use of the X.500 Directory

The Message Store uses the X.500 Directory to authenticate and configure users. This is based on work in *RFC 1836: Representing the O/R Address Hierarchy in the X.500 Directory Information Tree* and *RFC 1801: MHS use of Directory to Support MHS Routing*, which allows configuration of Message Stores, Message Store Users and the protocol links to and from MTAs, using a specially designed Directory User Agent. The Lightweight Directory Access Protocol (LDAP) is used by the Message Store to access the Directory for authentication.

The Message Store also uses the X.500 Directory to provide its message indexing service. For this access it uses Directory Access Protocol (DAP).

3.6 Other configuration

The directory specified as a containing user's root folder must exist prior to Message Store use and must be writable by the Message Store process. The Message Store will create individual users' root folders and subordinate folders as required.

Appendix A References

Refer to the documents listed in this appendix for more information on issues raised or referred to in Isode documentation.

Where specific documents can be obtained electronically, this is shown in the reference; otherwise refer to Section A.3, “Obtaining documents”.

A.1 Cited documents

Refer to the documents listed below for more information regarding issues raised or referred to in this Guide.

ITU-T Recommendation X.400 | ISO/IEC 10021-1:1990
Information Technology - Text Communication - Message-Oriented Text Interchange System (MOTIS) - System and Service Overview.

ITU-T Recommendation X.402 | ISO/IEC 10021-2:1990:
Information Technology - Text Communication - Message-Oriented Text Interchange System (MOTIS) - Overall Architecture.

ITU-T Recommendation X.413 | ISO/IEC 10021-5:1990
Information Technology - Text Communication - Message-Oriented Text Interchange System (MOTIS) - Message Store: Abstract Service Definition.

ITU-T Recommendation X.419 | ISO/IEC 10021-6:1990
Information Technology - Text Communication - Message-Oriented Text Interchange System (MOTIS) - Protocol Specifications.

ITU-T Recommendations Series X.500 | ISO/IEC 9594-1: 1993
Information Technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models, and Services.

RFC 1278
A String Encoding of Presentation Address, S. Kille

RFC 1327
Mapping between X.400(1988)/ISO 10021 and RFC822, S. Kille

RFC 1777
X.500 Lightweight Directory Access Protocol, W. Yeong, T. Howes, & S. Kille

RFC 1779
A String Representation of Distinguished Names, S. Kille

RFC 1801
MHS use of Directory to Support MHS Routing, S.Kille

RFC 1836
Representing the O/R Address Hierarchy in the X.500 Directory Information Tree, S.Kille.

A.2 Other publications

Chadwick, D. W. *Understanding X.500 (The Directory)*. International Thompson Publishing, July 1996. ISBN 1-85032-281-3.

Gardner, Ella and Ginsburg, Elliot. *Defense Message System Unclassified Directory Schema*. Mitre Corporation, Washington C3 Center. 5 February 1993.

National Institute of Standards and Technology. *Announcing the Data Encryption Standard (DES)*. Federal Information Processing Standards Publication 46-2, Dec. 1993.

National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication 186, May 1994.

National Institute of Standards and Technology. *Secure Hash Standard*. Federal Information Processing Standards Publication 180, May 1993.

National Security Agency. *SDNS Directory Specifications for Utilization with SDNS Message Security Protocol*. Specification SDN.702, 8 December 1992. Revision 2.0.

National Security Agency. *Access Control Concept and Mechanisms*. Specification SDN.801, 12 May 1999. Revision C.

North American Directory Forum. *SD-5: An X.500 Naming Scheme for National DIT Subtrees and its Application for c=CA and c= US*.

OIW Implementor's Workshop. *Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 12 - OS Security*. September 1994. ftp://nemo.ncsl.nist.gov/pub/oiw/agreements/12S_9409.ps

Ousterhout, J. *An X11 Toolkit Based on the Tcl Language*. Winter 1991 USENIX Conference Proceedings. ftp://ftp.scriptics.com/pub/tcl/doc/tkUsenix91.ps

Ousterhout, J. *Tcl: An Embeddable Command Language*. Winter 1990 USENIX Conference Proceedings. ftp://ftp.scriptics.com/pub/tcl/doc/tclUsenix90.ps

Ousterhout, J. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994. ISBN 0-201-63337-X.

Roe, M. *PASSWORD R2.5: Certification Authority Requirements*. Nov. 1992. ftp://cs.ucl.ac.uk/password/r25.ps

Rose, M. *The Little Black Book: Mail Bonding with OSI Directory Services*. Prentice-Hall, 1991. ISBN 0-13-683219-5.

RSA Data Security Inc. *PKCS #1: RSA Encryption Standard*. Nov. 1993.

RSA Data Security Inc. *PKCS #6: Extended-Certificate Syntax Standard*. Nov. 1993.

RSA Data Security Inc. *PKCS #7: Cryptographic Message Syntax Standard*. Nov. 1993.

RSA Data Security Inc. *PKCS #8: Private-Key Information Syntax Standard*. Nov. 1993.

RSA Data Security Inc. *PKCS #9: Selected Attribute Types*. Nov. 1993.

Stonebraker, M. and Kemnitz, G. *The POSTGRES next-generation database management system*. Communications of the ACM, Oct. 1991, vol. 34, (no. 10): 78-92. http://db.cs.berkeley.edu/papers/ERL-M91-62.ps.Z

versit Consortium. *vCard. The Electronic Business Card Version 2.1*. September 18, 1996. http://www.imc.org/pdi/vcard-21.ps

Welch, B. B. *Practical Programming in Tcl and Tk*. Prentice Hall. ISBN 0136168302.

X.400 API Association and X/Open Company Ltd. *API to Electronic Mail (X.400) CAE Specification: Issue 2*. 1994.

X.400 API Association and X/Open Company Ltd. *Guide to Selected X.400 and Directory Service APIs*. 1991.

X.400 API Association and X/Open Company Ltd. *OSI-Abstract-Data Manipulation API (XOM) CAE Specification: Issue 2*. 1994.

X.400 API Association and X/Open Company Ltd. X/Open API to Directory Services CAE Specification: Issue 2. 1994.

A.3 Obtaining documents

A.3.1 All documents

You can obtain ITU-T (CCITT) Recommendations, ISO/IEC Standards and draft standards, and paper copies of RFCs from:

- Omnicom PPI Ltd.
  Forum Chambers
  The Forum
  Stevenage
  Herts SG1 1EL
  England

  Telephone: +44 438 742424

  Fax: +44 438 740154

- Phillips Publishing Inc.
  7811 Montrose Road
  Potomac
  MD 20854
  USA

  Telephone: +1 301 424 3338

  Fax: +1 301 309 3847

A.3.2 ISO/IEC documents

Contact your national Standards Organization.

A.3.3 ITU-T (CCITT) documents

CCITT/ITU-T Recommendations and draft documents can be obtained from:

International Telecommunications Union
General Secretariat - Sales Section
Place des Nations
CH-1211 Geneva 20
Switzerland

A.3.4 RFCs

Electronic copies of RFCs are available from the following servers:

- http://ftp.isi.edu/in-notes/
- http://222.rfc-editor.org/

A.3.5 XOpen documents

You can obtain X/OPEN Company Limited and X.400 API Association documents by post from:

X/OPEN Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire
RG1 1AX
England

or by electronic mail submission to:

XoSpecs@xopen.co.uk

Glossary

The terms and abbreviations necessary for use of this book are explained below.

Distinguished Name
A set of attributes that uniquely identifies an object or user within a global set of network addresses. It is constructed of concatenated values, in a similar way to the construction of a file name. In the case of a Distinguished Name, the values identify such things as the country, organization, organizational group and user name associated with the user.

Directory Information Tree
A way of defining relationships between, and access routes to, the complete set of information contained in the X.500 Directory.

DIT
See Directory Information Tree.

DN
See Distinguished Name.

LDAP
See Lightweight Directory Access Protocol.

Lightweight Directory Access Protocol
Protocol used to provide simple access to the X.500 Directory.

Message Store
A means of providing a continuously available, secure storage place for messages being transferred either to or from a UA. Each Message Store services only one O/R address. However, a Message Store database consisting of many Message Stores can be configured to service multiple O/R addresses.

MS
See Message Store.

Message Transfer Agent
A component of the MTS, which, in co-operation with other MTAs, transfer and deliver messages to the intended recipients.

Message Transfer System
The MTS delivers, to one or more recipients, messages submitted to it by UAs. An MTS consists of a number of MTAs

MHS-DS
An abbreviation used (here) to refer to a method of configuring MTAs and Message Stores using an X.500 Directory.

MTA
See Message Transfer Agent.

MTS
See Message Transfer System.

O/R
Originator/Recipient - the User Agent sending a message or to whom a message is to be delivered.

O/R Address
A collection of information (an attribute list) that uniquely identifies the User Agent to whom a message is to be delivered or notification of delivery returned. The O/R address also identifies the user’s point of access to the MHS.

Open Systems Interconnection
Set of design rules and protocol specifications defined to allow systems from any source to talk to each other, thus evading problems of hardware and software incompatibility.

OSI
See Open Systems Interconnection.

P3
The X.400 protocol used in communications to and from an MTA.

P7
The X.400 message handling protocol used to access a Message Store.

UA
See User Agent.

User Agent
A User Agent is an application process that interacts with the MTS or a Message Store, to submit messages for a single user. It can also accept delivery of messages, either directly from the MTS or by retrieving them from a Message Store.