xisoftware

Xi-Text Release 23

Introduction and User Guide

1 What is Xi-Text?

Xi-Text is a fully functioned, high performance Print Spooler and Management System for a wide rang e of Unix and Linux platforms.

It acts as a complete replacement for the standard Unix printing system and provides a wide variety of print post processing options, form type handling, alignment pages and numerous other facilities.

In addition, **Xi-Text** runs completely transparently over networks of hosts, sharing jobs on any host with printers on any other host.

Control of and access to all the **Xi-Text** facilities is made available through a variety of interfaces including:

- Command line or "shell commands"
- Full screen text-based using "curses"
- X-Windows (Motif toolkit)
- MS Windows clients
- Web Browser interfaces (two variants)
- Unix or Linux based API (supporting C and C++)
- MS Windows based API (supporting C and C++)

In all cases apart from "command line" the access is fully interactive and the display is automatically updated to take account of changes to jobs and printers on the screen whether by the user, another user or by print jobs starting and finishing.

A lot of effort has been made to make **Xi-Text** fully configurable on a per user or system-wide basis. You can alter the format and content of screen displays, help messages, command keystrokes and option names to suit your requirements such as foreign language support or personal taste.

Common sets of options and parameters may be saved by the user to provide a local or global set of defaults for particular tasks.

Xi-Text is implemented so as to minimally impact upon the system. In an idle state, it typically uses less resources than **lp** or equivalents. It is implemented in ANSI C, apart from the MS Windows client, which is in C++, and parts of the web browser interfaces, which are in JavaScript. All of the product was developed and written by Xi Software. No additional third-party database or other software is used or relied upon.

All current Unix and Linux platforms are supported and different platforms may run the product concurrently inter-networking with each other.

2 Jobs

Each print request is referred to as a *print job* and consists of two parts:

1. The actual data to be sent to the printer (or other output device such as a fax modem).
2. Various information about the job such as the number of copies, the form type required and so on.

A job is created, or "submitted" from a variety of interfaces, including from a remote host or Windows Client.

The job may be printed on any printer on the network which is available to print that type of job, or it may be restricted via the various parameters to only print on one or more specific printers when available.

Jobs are held in a single queue and "matched" to a suitable printer when it becomes available.

A comprehensive permissions and privileges system determines whether various users may access other user's jobs and may be used to limit what a given user may do.

Jobs may be taken off the queue and archived, subsequently being restored to the queue if required.

3 Printers

The second kind of entity which Xi-Text supports is that of a printer, which may of course be any output device.

Each printer to be supported by Xi-Text is represented by a separate entry in the table of printers managed by the product. This may be a directly connected (e.g. serial or parallel port) printer or a network printer.

Any number of printers may be supported and these may be transparently shared across the network and jobs automatically sent to them from other hosts running Xi-Text.

Each printer has a *formtype* associated with it. Generally this conforms to the paper type currently loaded on the printer, but as is explained later, there are several refinements which may be made to how individual jobs are printed.

Alignment routines to ensure that (for example) continuous feed paper is properly set up before jobs are printed.

If the printer stops printing in the middle of a job for some reason, when printing is resumed, the job will be resumed at the interrupted page.

4 A quick tour

Before going on to the submission of jobs, let us look at the interactive display of jobs and printers.

At the most basic level is the shell-level interface. The command sqlist is used to display jobs and the command splist is used to display printers, thus:

```
$ sqlist
avon:29746 jmc Job1 scrap 1443 1 165
15033          Job2 scrap 1385 1 150
15034     jmc Job3 scrap 830 1 150
$ splist
deskjet     lp0    a4.ps halted
kira:hp5     <hp5> plain halted
kira:tony    <axis> plain halted
kira:starmat lp1    labels idle
avon:demo    <dh>  scrap halted
$
```

In this example there are 3 print jobs and 5 printers on various machines. Nothing is actually being printed at the moment.

In common with many Unix commands the default output from these programs is rather terse. You may well want to start off with column headings which can be displayed using a program option thus:

```
$ sqlist -H
Jobno    User Title Form  AtPg Pgs Cps Pri Printer
avon:29746 jmc Job1  scrap   1443   1 165
15033    jmc Job2 scrap   1385  1 150
15034    jmc Job3 scrap   830  1 150
$ splist -H
Printer   Device  Form  State:Msg Jobno User
deskjet   lp0   a4.ps  halted
kira:hp5   <hp5>   plain  halted
kira:tony  <axis>  plain  halted
kira:starmat lp1   labels idle
avon:demo  <dh>    scrap halted
$
```

It is possible to save away the "`-H`" as part of your own personal set of defaults so that you do not have to specify it every time. You can also respecify which fields of jobs and printers are to be displayed and in which order and make those part of your own personal defaults.

In this display, the first column in the sqlist output is the job number. This is a unique number, assigned when the job is submitted. Jobs on the queue hosted by other machines are prefixed by the machine name and a colon, as with "`avon:29746`" as above.

There then follow the user who submitted the job, the title, and the form type. This is then followed, by two fields. The second gives the size of the job in pages or in bytes (if there are no clearly defined pages) and the second gives the position (page number or byte) reached.

The next field is the number of copies to be printed.

Then comes the priority, a number which may be used to determine the starting position on the queue. Finally the destination printer is shown. This may be blank to mean "don't care". As the job starts printing, the printer actually selected is put in here.

The printer display gives the name assigned to the printer, preceded in the case of another machine by the host name as in four of the cases, a "device name" or identity, the loaded form type and the state the printer is in. Only one printer is ready to run, or idle, the others are halted. If a problem arises and the printer goes offline, a message may be displayed here explaining the problem.

The final two fields apply when jobs are printing, and give the job number being printed and the user name of the job owner.

If you want to change something, such as to request additional copies of a job, you can use utilities such as sqchange, for example:

```
$ sqchange -c3 15033
$ sqlist -H
Jobno    User Title Form  AtPg  Pgs Cps Pri Printer
avon:29746 jmc Job1 scrap    1443  1 165
15033    jmc Job2 scrap    1385  3 150
15034    jmc Job3 scrap     830  1 150
$
```

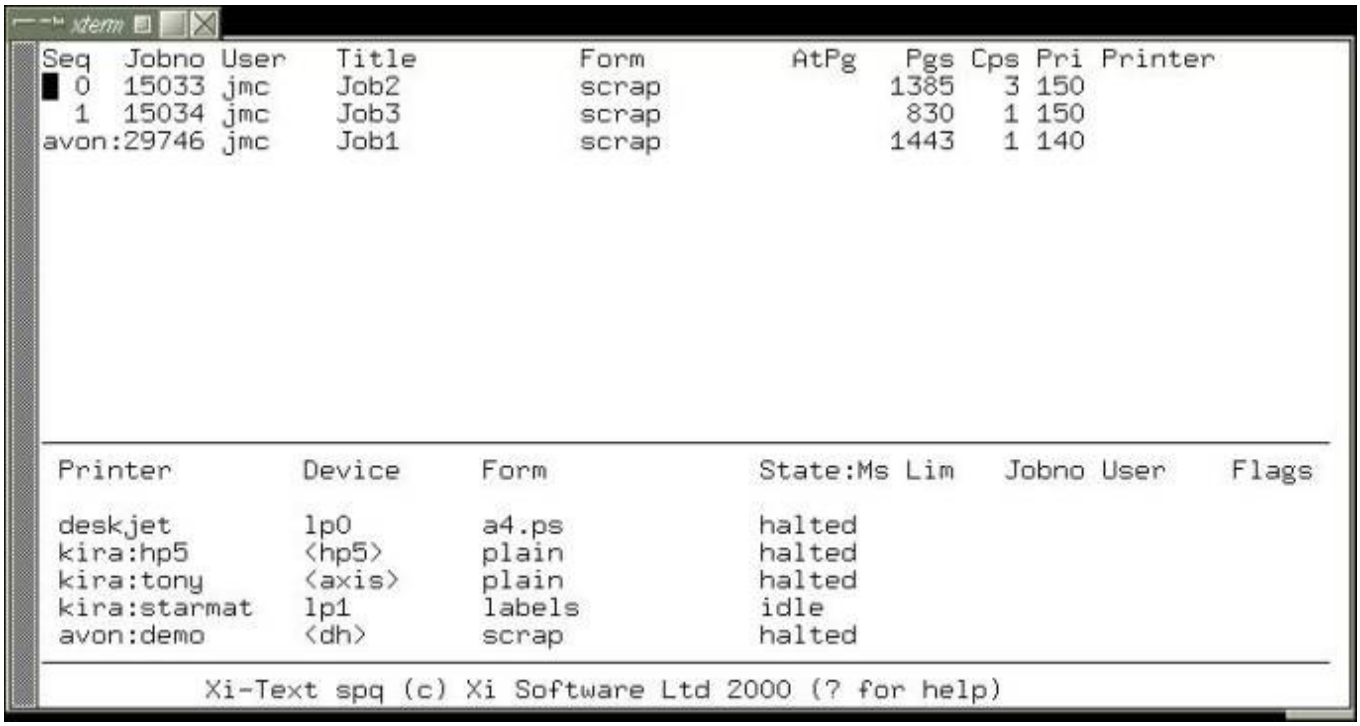
All the operations, including changing and deleting jobs, may be done with remote jobs just as much as with "local" jobs, for example to change the first job in the list, reducing the priority, you could type:

```
$ sqchange -p140 avon:29746
$ sqlist -H
Jobno    User Title Form  AtPg  Pgs Cps Pri Printer
15033    jmc Job2 scrap    1385  3 150
15034    jmc Job3 scrap    830   1 150
avon:29746 jmc Job1 scrap   1443  1 140
$
```

Of course this manual method of changing job details is tedious and error-prone and so a variety of other interfaces are provided. These have the additional benefit of being interactive, with changes, whether by yourself or other users being immediately reflected on the screen. Likewise to perform operations the relevant jobs or printers are selected and the operation applied.

4.1 Spq

The basic interface is the full-screen character interface, spq. The above list of printers and jobs would look like the following:



To operate upon a job or printer, it should be selected by moving the cursor to it and typing one of various single-keystroke commands.

Jobs may be displayed on the screen, and page ranges and other attributes set if required.

Context-sensitive help is available in all parts of the screen, by typing ? (although the help function may be bound to alternative or additional characters).

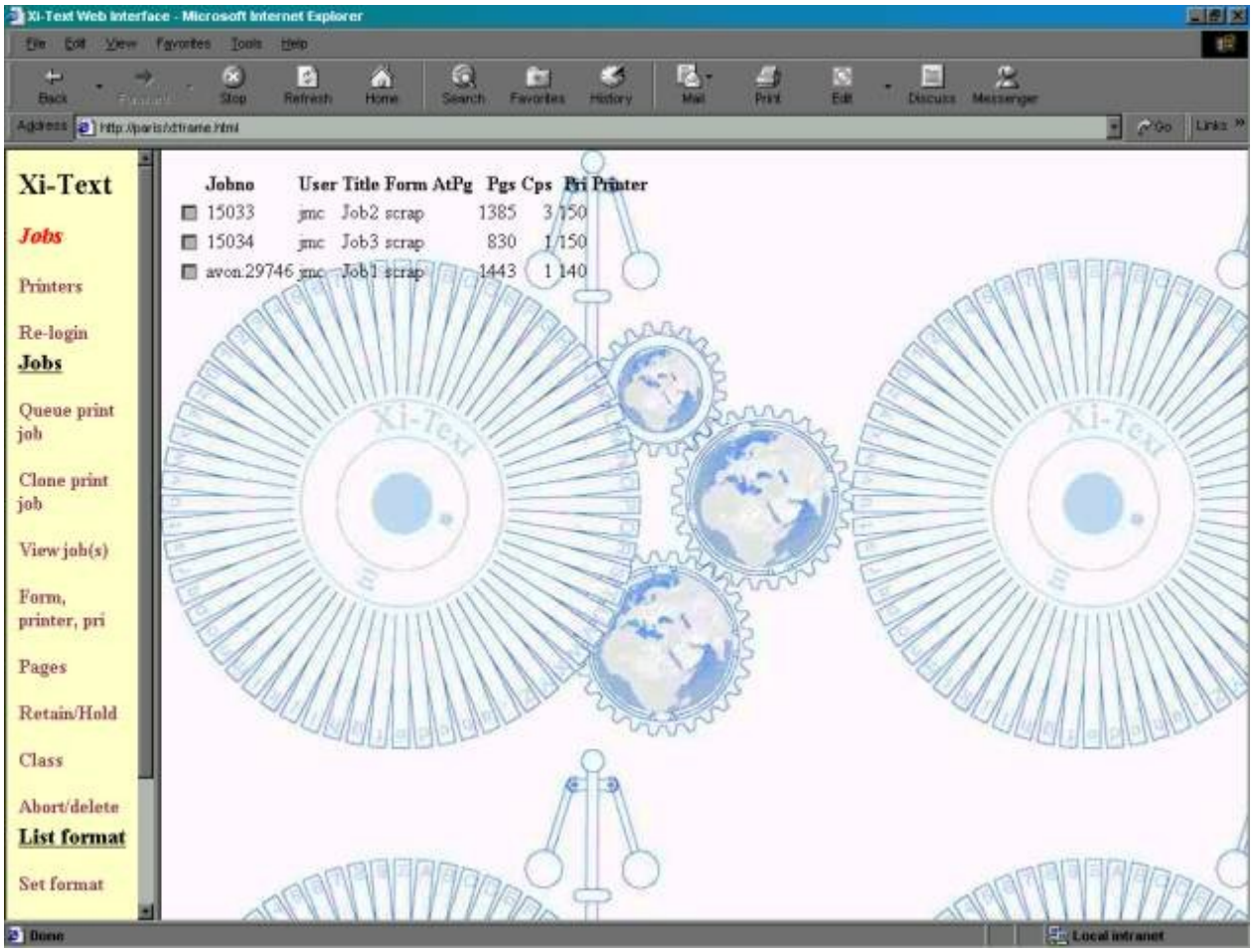
4.2 Xmspq

Also available is the X-Windows interface. The same job and printer display would look like the following:

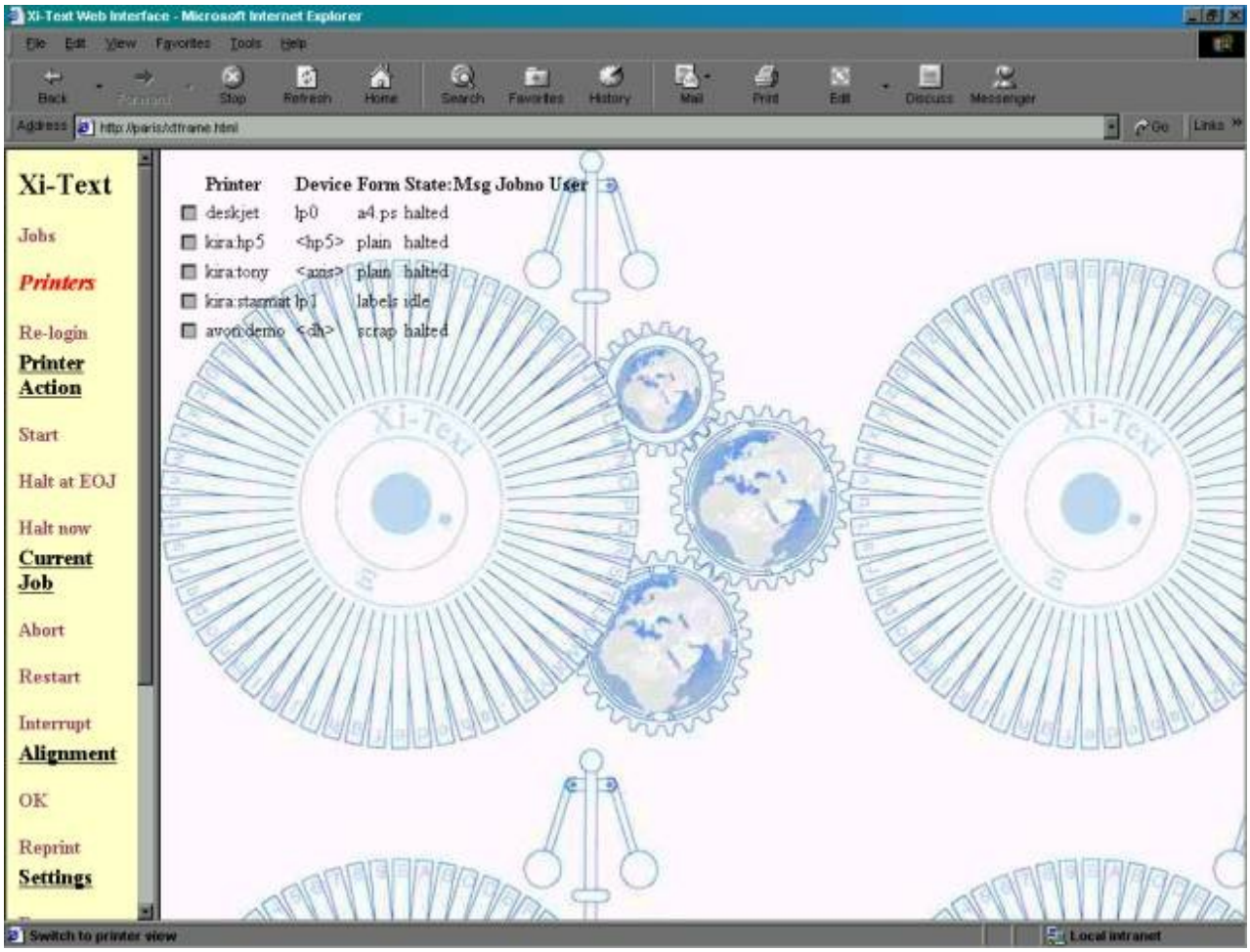
Notice how printers in different states may be represented in different colours.

4.3 Web Browser

Alternatively the Web Browser Interface is available, this handles jobs and printers separately, the jobs thus:

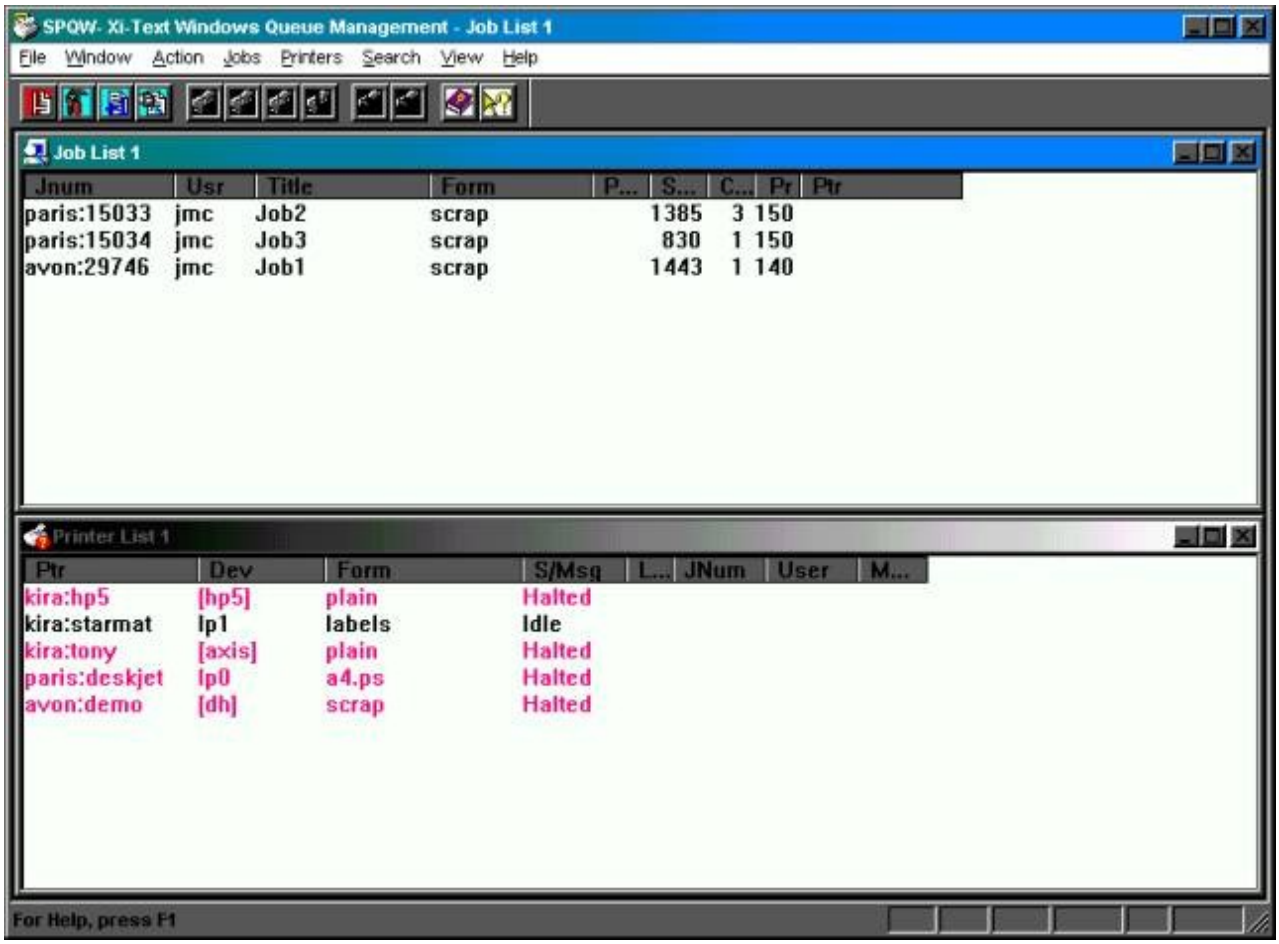


and printers thus:



4.4 Windows Client

Finally a PC Client Interface allows jobs and printers to be managed from a PC client, with the above jobs and printers looking like the following:



Again sets of colours may be assigned to printers in different states.

Various options, commands and dialogs allow you to select jobs and printers and modify various parameters of jobs and printers subject to the security restrictions.

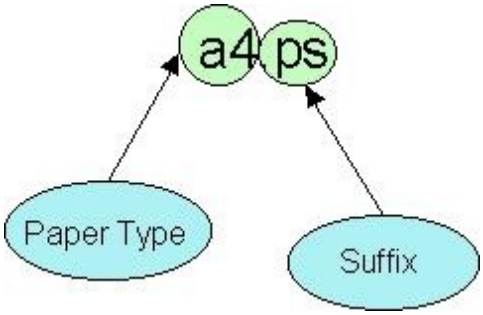
All these interfaces allow the user to display the job and where appropriate set or reset start and end page ranges from the currently displayed page.

Additionally, an API, with supporting C and C++ is available for both Unix and MS Windows, with all the above facilities available.

5 Form Types

One of the key aspects of the function of **Xi-Text** is that of form types. These enable the user to match printing requirements both to the type of paper being used and control variations in printing, such as automatically selecting orientation or input and output bins.

A form type consists of two parts, the parts being separated by a fullstop or possibly a minus sign.



The paper type denotes the physical paper on the printer, in this case "a4". The optional suffix, in this case ".ps", denotes additional handling instructions, possibly for processing postscript data.

You have to stop and restart a printer in order to change the paper type. This corresponds to the fact that you have to physically touch the printer and load new paper into it.

On the other hand some functions, such as selecting "portrait" versus "landscape" printing, or selecting output bins can be performed under software control by sending an escape sequence to the printer. It would be pointless stopping and starting the printer between jobs with different requirements.

Thus a job is "matched" to a printer if the "paper type" part of its form type is the same as that of the printer. The suffix, if any, is ignored at this stage.

The suffix is then interpreted by a control file for the printer when printing takes place. This may cause actions to be taken at various stages of printing the job in order to achieve the desired effect. These actions may be any combination of transmitting strings or executing a command when:

- The printer is started.
- The printer is halted.
- The suffix is first selected.
- The suffix is deselected (because some other suffix is selected).
- The start of each job.
- The end of each job.
- The start of each page.
- The end of each page.

- If the job is aborted during printing
- If the job is restarted from the beginning during printing.

In addition the whole job may be translated or post-processed as it is sent to the printer.

Alignment pages, header pages, page delimiters and numerous other facilities may also be selected according to the suffix (or made standard for the form type).

6 Submitting print jobs

Print jobs may be submitted to **Xi-Text** using the following methods:

- From the command line, by running the utility spr and giving the file to be printed.
- From a remote system, using the utility rspr, otherwise similar to spr.
- Using the "LP emulator" to simulate lp translating the parameters to be appropriate for **Xi-Text**.
- Using the LPR-style interface to receive print files from an external system in LPR format, translating to **Xi-Text** format.
- From the Windows interface, enabling Windows programs to print directly to a "printer" which translates jobs to **Xi-Text** format and submits them.
- From the API (Unix or Windows versions).

Once queued, jobs and their parameters are saved to disk and regularly checkpointed in case the system crashes.

7 Security Features

Two distinct forms of security are provided, both to prevent unauthorised users from making inappropriate use of or changes to the system and to provide a mechanism for dividing up the facilities between groups of users.

7.1 Privileges

Each user has a set of privileges, which allow or deny permission to do such things as:

- View jobs which do not belong to him or her
- Make changes to jobs which do not belong to him or her
- Stop or start printers
- Add new printers
- Use form types other than a specified set of defaults
- Use printers other than a specified set of defaults
- Access remote jobs
- Access remote printers
- "Unqueue" jobs, i.e. make a copy of a job on the queue, deleting the original.
- Edit the privileges and other attributes of users' profiles with **Xi-Text** (including the privileges).

Attempts to perform other than the specified set given by the permissions will be rejected.

Privileges may be set using the utilities spulist, spuchange, spuser and xmspuser, or via the API.

7.2 Class Codes

Class codes provide a mechanism for subdividing users, jobs and printers.

Each user, each job and each printer has a class code, which is 32 bits, represented by the letters A to P and a to p. The class code is set for each user at the same time as his or her privileges are set, using the utilities spulist, spuchange, spuser and xmspuser, or via the API, and each user has at least one of these bits set.

Unless specified otherwise, any job submitted by the user and any printer created by the user will have the same class code as that of the user. In general, a user may remove bits from the class code of a job or printer he or she creates, except that at least one bit must be left, but he or she cannot add extra bits (unless he or she has so-called "override class" privilege, not usually given).

The user can only "see" jobs or printers which share at least one class code bit with his or her own personal class code. Likewise a job is only printed on a printer which shares at least one bit of the class code with the job (the user's class code is not considered at this point).

In this way, jobs and printers to which one group of users should have no access may be made "invisible" and other jobs and printers "visible" and the converse to a different group. For example suppose there are three groups of users:

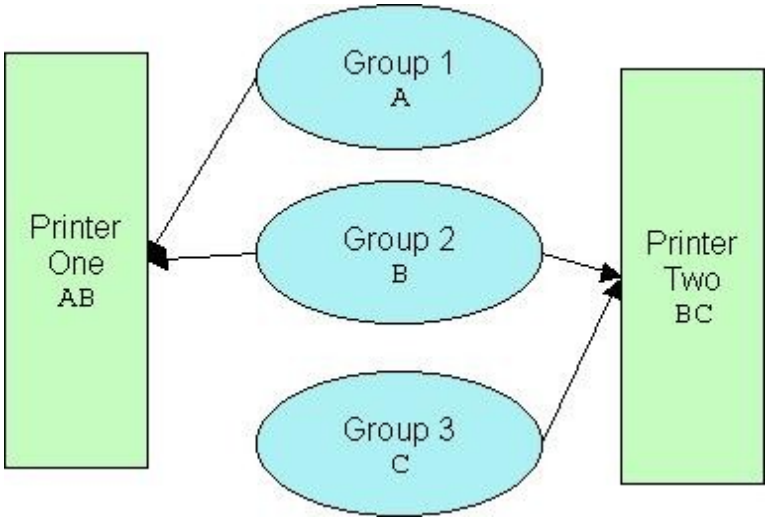
1. Group one has class code bit **A** set (only).
2. Group two has class code bit B set.
3. Group three has class code bit C set.

Likewise suppose there are two printers.

1. Printer one has class code bits A and B set.
2. Printer two has class code bits B and C set.

In this case any jobs submitted by users in group one will have bit A set and will therefore only visible to members of that group, and likewise for groups two and three.

On the other hand printer one will be "visible" to members of groups one and two, and will print jobs for either of those groups. Printer two will be "visible" to groups two and three, and may print jobs for either of those groups.



Also possible might be a "manager" of groups 1 and 3 with class codes A and C set enabling him or her to "see" both printers and jobs for groups 1 and 3.

Clearly by combining combinations of class code bits, arbitrarily complicated subdivisions of the jobs and printers can be achieved.

8 Conclusion

This User Guide is only intended to give readers an overview of the facilities of **Xi-Text**. For more detail on the various parts of the product, please see the *Reference Manual*. For installation details, please see the *Administration Guide*.

Additional manuals cover the MS Windows interface and the API.

Xi-Text is a product which has been continually developed and enhanced since 1984 and we have striven to make it as easy to use, comprehensive and reliable as possible.

We welcome the opportunity to supply evaluation copies and assist new users with implementing their printing requirements efficiently.