



Outputware *ONLINE Documentation*

Xprint V6.0

Application Programming Interface (API)

User Guide

Edition June 2000

Copyright © 1999: Fujitsu □ Siemens □ Computers □ GmbH
Identification: U8843-J-Z915-6-76

Copyright and Trademarks

All rights reserved. □

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

1 Preface

This guide describes the Application Programming Interface (API) of the Xprint V6.0 system. The API allows programmers to access the functions of the Xprint system by means of a C language programming interface. This guide includes information about the C functions and C data structures that make up the programming interface.

1.1 Target group

This guide is written for application programmers who are proficient in using the C programming language. A knowledge of Xprint administration and general use is assumed.

1.2 Notational conventions

The following conventions are used throughout this guide:

Convention	Use
"double quotes"	For references within the manual and to other manuals.
<i>italics</i>	In continuous text: for commands, options and file names. In examples: for variable parts of commands.
fixed pitch	For constants, proper names, screen output and menu texts as well as invariable parts of examples.
bold	For emphasis in continuous text.
fixed pitch and bold	For your input in examples of dialogs.
	Actions you must perform.
	Refers to important information which you should note.

Table 1: Notational conventions

1.3 Summary of contents

This guide consists of this preface, five chapters and an appendix.

This preface introduces both the Xprint application programming interface and this guide.

Chapter 2, "**Preface**", introduces the main concepts behind the Xprint API.

Chapter 3, "**Preface**", provides an overview of database objects and data structures that support the use of the Xprint database.

Chapter 4, "**Preface**", explains the four API functions *Xp_alloc()*, *Xp_free()*, *Xp_clean()* and *release_cli()* that enable application programs to manage memory efficiently.

Chapter 5, "**Building an application**", includes hints on building your Xprint application.

Chapter 6, "**Function descriptions**", describes the functions of the Xprint API.

Chapter 7, "**Preface**", describes how to avoid conflict between Xprint and other applications using the same port.

The appendix consists of several main sections. The first two sections "**The client.h file**" and "**The xprint.h file**" include definitions of data structures from the header files *client.h* and *xprint.h* that are used directly by application programs. Then follows the section "**xp commands and corresponding structures**", which lists the structures or structure elements to be specified with the corresponding API functions for the various possible options of the individual Xprint commands. The last section "**List of messages**" contains a complete list of the Xprint error, warning and information messages.

1.4 Other publications in this set

In addition to this guide, there are three other guides in the Xprint V6.0 document set. These are:

- The "Xprint V6.0 - User and Administrator Guide"

This manual, which concentrates on the Xprint commands, describes the tasks which can be executed using Xprint and the required procedures.

- The "Xprint V6.0 - Reference Manual"

This guide provides an alphabetical description of all Xprint commands and the possible options. Furthermore, it contains information on the following topics:

- Xprint users
 - Object naming conventions
 - Selecting Xprint objects
 - Messages
- Xprint Management V1.0A. Graphical User Interface

This manual provides an introduction to Mercator, the new graphical user interface for Xprint.

2 Introduction

The Xprint API provides an interface for 'C' language programs to access the functions of the Xprint system. The API consists of a library of functions and a set of data structures and definitions of constants. Together, these enable an application program to access the functions of the Xprint system.

2.1 Database

Xprint is a central printing system with distributed access. This system contains objects relevant to a print spooler, such as printer devices, fonts, forms and user descriptions.

The entire functionality of Xprint is managed through the handling of these objects; for example, submitting a file to print adds a job to the system. The objects are stored in two different areas; the central database for the "static" objects (i.e. all objects except the jobs) and the server job queues for the jobs.

The central database, which is referred to simply as "the database" in this guide, can be accessed from any host in the system (distributed access) and is supported by sophisticated management functions of Xprint to ensure its integrity.

The server job queues, referred to as the "spool queues" in this guide, exist on each host executing an Xprint server.

The management of the database and spool queues is fully supported by the Xprint system and should not directly concern the application program or its use of the API.

2.2 Functions

The Xprint API provides an interface to a group of 'C' language functions that allow manipulation of objects in the Xprint database and spool queues. Although there are different functions to access the print queue and the database, they are similar in purpose and format. For example:

add_all_cli() adds an object to the database

add_job_cli() adds a job to a printer queue

The individual functions are described in [Chapter "Function descriptions"](#).

2.3 Data structures

The data structures describe the database objects and must be consistent for successful use of the API. The Xprint API is supported by 'C' language structures. These structures, together with the defined constants, are supplied by two header files, *xprint.h* and *client.h*. The use of data structures is described in [Chapter "Preface"](#). The definitions of the most commonly used data structures are reproduced in the sections "[The client.h file](#)" and "[The xprint.h file](#)" in the appendix.

2.4 Memory management

The Xprint API provides functions to assist the application program and to enhance the security of the Xprint environment. Memory management is described in [Chapter "Preface"](#).

2.5 Xprint architecture

The Xprint architecture is introduced in this section for your interest, although it is not essential information. The Xprint spooler system is composed of five basic units. These are:

- client
- server
- supervisor
- database manager
- Xprint daemon.

2.5.1 The client

"Client" is the term used to describe the part of Xprint executed in the process of calling the API. Usually, the client checks the syntax and semantics of the parameter list of the API and interacts with other Xprint components, which run as independent processes. Every interaction with Xprint includes the client component. The command line interfaces are compiled by the API and consequently use the client component.

2.5.2 The supervisor

A supervisor is responsible for executing jobs on printers it controls. As part of the job execution process, the supervisor is also responsible for:

- transforming the data (if specified as part of the *add_job_cli()* function)
- transmitting the data to the printer
- controlling the printer and carrying out job recovery in the event of printer failure.

In addition to job processing, the supervisor is always able to receive requests for job interruptions, terminations or any other information which may be required by its server. When the supervisor initializes itself to drive a printer, it gets a set of job recovery rules. If an error occurs during the execution of a job, the supervisor can handle the job recovery. The supervisor can also manage a controlled termination of a print job in the event of a termination request.

2.5.3 The server

The server is an administrative process within Xprint, which controls a number of supervisors (printers) and schedules the print jobs to be processed by supervisors. It is responsible for:

- administering a number of supervisors and the jobs processed by them
- scheduling jobs between different supervisors
- managing its job queue by:
 - adding new jobs when requested by a client
 - modifying existing jobs when requested by a client
 - modifying a job's status when requested by either a client or a supervisor
 - deleting jobs from the database when requested by either a client or a supervisor.

2.5.4 The database manager

The Xprint database manager stores all the information about the Xprint objects, except job information, which is managed by the server. Each host on the network holds a back-up copy of the Xprint database, with one host being nominated the master database manager. Although each local host can manage both interrogation and update requests, only interrogations can be satisfied by the local host. Update requests are converted into a request to the master database manager which, after updating its own database, broadcasts the new information to the other hosts on the network. Database managers can communicate with each other to receive update information without blocking incoming requests.

Although only one host can be the master database manager at any one time, a hierarchy is established by the Xprint administrator to provide secondary and subsequent levels of recovery in the event of the master failing. This point is covered more fully in the "Xprint V6.0 - User and Administrator Guide".

2.5.5 The Xprint daemon

During startup of the Xprint system on each host, the Xprint daemon is the first process to run. It creates the database manager and waits for requests from clients to start server or supervisor processes. Each Xprint daemon is responsible for maintaining the availability of processes on its host and for controlling the correct termination of those processes when Xprint terminates. To assist with these management tasks, the Xprint daemon maintains a process table with entries for each active process.

3 Objects and data structures of Xprint API

This chapter provides an overview of the database objects and data structures that are used to communicate between an application program and Xprint. This overview also shows the logic that links the database objects and data structures to their usage by an application program.

Xprint is object-oriented. All interaction with the Xprint system consists of adding, deleting, modifying, displaying and copying objects.

Xprint provides data structures for all the objects that may be held in the database or the spool queues. Besides object data structures, Xprint makes use of data structures for communication; for example, Xprint will respond to a request for the status of objects with a data structure filled with the object status information.

3.1 Objects

The Xprint system recognizes 17 objects. Each object has an associated structure and a two character identifier (identifiers are upper case names equated to numeric values by *define* statements). Identifiers and structures are defined in the *xprint.h* file. The following table also contains additional structures in the column "sta structure". Their function and relationship to objects are described later in this chapter in the [Section "Result structures"](#).

Object	Identifier	Object structure	sta structure
Server process	SV	Xp_server	sta_sv_ext
Supervisor process	SP	Xp_superv	sta_sp_ext
System	SY	Xp_sys	
Printer device	DV	ECM_printer	sta_dv_ext
Printer device group	DG	Xp_dgroup	sta_dg_ext
Gateway	GW	Xp_gateway	sta_gw_ext
Font description	FT	Xp_font	
Form description	FM	Xp_form	
Permission list	PN	Xp_perm	
User definition	US	ECM_security	sta_us_ext
Filter descriptor	FL	Xp_filter	
PCL	PL	Xp_pcl	
Recovery rule	RR	Xp_rec_rule	
Host descriptor	HO	Xp_host	sta_ho_ext
Job template	JT	Xp_job_temp	
Job	JO	Xp_job	sta_jo_ext
Loadable resource	LR	Xp_soft_load	

Table 2: Objects Xprint can manage

This table lists all the objects Xprint can manage. The system has two kinds of storage area; a main database that can be viewed and managed from any host within the Xprint system (i.e. distributed access) and the local job queues of the servers. This guide refers to these two storage areas as the database and the spool queues. The database and spool queues are accessed in a similar manner, although different function names are used to access the spool queue, for example *add_job_cli()* will add a job to a spool queue and *add_all_cli()* will add any other object to the database.

3.2 Object structures

Each object has a data structure used to represent the description of a particular instance of the object. Xprint provides the definitions for these data structures, for the convenience of the application programmer and for increased reliability of the system.

All API functions that manipulate objects within the database or spool queue require at least one object data structure to be passed as a parameter by the application code. To allocate the memory for these structures and to initialize the structure, the function `Xp_alloc()` is used. Refer to the [Chapter "Preface"](#) for more information about allocating structures.

Object structures may be used in one of three ways:

- to define an entry of the particular object type to be added to the database or a spool queue. In this case the data structure is known as an add structure.
- to select an entry, or entries, of the particular object type held in the database or a spool queue to be changed, copied, deleted, modified or have its status provided (show). When used in this way, the data structure is known as a select structure.
- to provide the revised information for entries that are modified. In this case the structure is known as a modify structure.

The data structure definition does not change with the use of the data structure. However, the content of the data structure does vary with its use.

The definitions of the data structures are contained in the `xprint.h` file and are also reproduced in the appendix. The uses of the data structures in the three forms are described in the following sections.

3.2.1 Add structures

Add structures are used to describe a particular instance of an object that is to be added to the database or spool queues. When a data structure is allocated (by `Xp_alloc()`), it will be initialized as empty. The application code must fill in the fields as necessary to describe the corresponding attributes for the instance of the object to be added. Where the application does not require the assignment of a meaning to a particular attribute, the corresponding field should be left as the empty value; Xprint will then allocate default values.

The data structure definition should be consulted to determine which fields (attributes) are available and the valid settings. These definitions can be found in the `xprint.h` file and are also reproduced in the appendix. It may also be useful to consult the corresponding add command for the object in the "Xprint V6.0 - Reference Manual" or for the job object in the "Xprint V6.0 - User and Administrator Guide".

Many *string* fields allow a list; these should have each item separated by at least one space or tab character.

3.2.2 Select structures

Select structures are used with functions that change, copy, delete, modify or show objects. The function of the select structure is to provide a set of criteria that is used to identify the target object or objects within the database or spool queue. Select structures are allocated by `Xp_alloc()` and initialized as not modified. Fields in the select structure that are left unmodified play no part in the selection of objects.

The criteria can be specific, to select a maximum of one object, or non-specific, in which case a set of objects may be selected. Specific selection is achieved by filling one or more fields for which objects (of the same type) must have a unique value. For example, the user name of a user definition object (structure `ECM_security`) must be unique; therefore, by entering the full user name in this field (`sc_usr_name`) in the select structure, at most just one object will be selected.

The selection of a set of objects can be done in two ways: by filling in only field(s) that are not unique, or by using wildcards. The following two examples, both of which are based on selecting user definition objects, should clarify this case.

Example 1. Filling in the maximum priority field (`sc_max_priority`) with, say, 10 and leaving all other fields unmodified will select all user definition objects where the maximum priority is 10.

Example 2. All user definition objects where the user name (`sc_usr_name`) starts with "jo" can be selected by setting this field in the select structure to the wildcard value "jo*".



Wildcard settings follow standard conventions, i.e., both "*" and "?" characters may be used. Wildcard settings apply to any field in the structure defined as a *string*.

Where more than one field of the select structure is filled, a logical *AND* is used to combine the criteria. The above two examples when combined into one selection would cause all user definition objects with a maximum priority of 10 and a user name beginning with the characters "jo" to be selected. This is demonstrated in the following code example, where the object information function demonstrates the use of the selection structure.

```
struct disp_mdI *ptr;
struct ECM_security *sec;
short size;
□
sec = (struct ECM_security *) Xp_alloc(US,SEL_OR_MOD);
sec->sc_usr_name = "jo*";
sec->sc_max_priority = 10;
size=shw_all_cli(US,sec,NULL,&ptr);
```

3.2.3 Modify structures

Modify structures are used to modify explicit fields (attributes) of selected objects. API functions that modify objects require both a selection structure and a modify structure.

Modify structures are allocated by *Xp_alloc()* and are initialized as unmodified. Fields in the modify structure that are left unmodified will leave the corresponding fields in the selected objects unmodified. Only fields for which a new value is entered in the modify structure will be modified (for all selected objects).

New values in a modify structure are the same as for the equivalent add structure with the following extensions:

Field type	Extension value
unsigned short	RESET_US
short	RESET_S
unsigned long	RESET_UL
long	RESET_L
string	RESET_ST

Table 3: Field types and extensions

These values request that the field is to be set to its default value (i.e. the value that would be allocated when adding the object if the field were left empty!).

3.2.4 Sequence structures

So far, object data structures have been described. However, there is a sub-structure that is defined for some object data structures that needs special mention. This is the sequence structure.

Some aspects of object definitions require a list, for example a printer group (DG) requires a list of printers. Such lists are variable in length and therefore cannot be accommodated directly in the object structure. This is resolved by use of a sequence structure. The object structure simply contains a sequence structure.



Sequence structures do not describe Xprint objects, but are only parts of the full object structures. Therefore, if memory needs to be allocated for a sequence structure the *Xp_alloc()* function should not be used, instead a standard memory allocation function such as *malloc()* should be used.

Note also that it is the responsibility of the application code to free the memory used by the sequence structure before freeing the object data structure with *Xp_free()*.

The definition of a sequence structure is as follows:

```

struct sequence
{ long seq_card; /* number of elements in the sequence */
  char *seq_ptr; /* an array of elements of dimension seq_card */
};

```

The pointer *seq_ptr* may point to an array of type *char* or an array of type *string* (defined as type *char**). To indicate which of these two is addressed, the convention demonstrated in the following two examples is used throughout the header files and by this guide:

```

struct /* string */ sequence dg_devlst; /* device list */
struct /* char */ sequence sv_schedpol /* scheduling policy */

```

Sequence structures are part of the object data structure definition and therefore exist in the three forms of *add*, *select* and *modify* structures. Use of the sequence structure in the *select* structure and *modify* structure types provides a powerful addition to the features already described for these structures. For example, selecting the set of printer groups (structure *Xp_dgroup*) that contain at least the printers *dv1* and *dv2* is not possible with the methods mentioned in the earlier description, but is easily achieved with a sequence structure.

When a sequence structure is used as a selection criterion, the following convention applies: The first element is an action tag and not a member of the sequence itself. It may take two values: either a char (for example, "+") for a sequence of character elements, or a string (for example, "+") for a sequence of string elements. The first element has the following effect when used to select objects (the second and subsequent elements form the comparison sequence):

- + indicates that the comparison sequence is considered a subset of the sequence to be matched.
- = indicates that the comparison sequence must form an exact match with the target objects to be selected.

For string sequences, the order of the elements is not significant. For character sequences, the order is significant. A comparison sequence element containing zero as cardinal and NULL as pointer will match every target sequence element.

Example

```

struct Xp_dgroup *dgr
static char * seq[3] = {"+", "dv1", "dv2"};
dgr = (struct Xp_dgroup *) Xp_alloc(DG,SEL_OR_MOD);
dgr->dg_devlst.seq_card = 3;
dgr->dg_devlst.seq_ptr = (char*)seq;

```

This select structure will select printer group objects containing at least *dv1* and *dv2* in their device list. By comparison, the following example will select printer groups with exactly *dv1* and *dv2* in their device list.

```

struct Xp_dgroup *dgr
static char * seq[3] = {"=", "dv1", "dv2"};
dgr = (struct Xp_dgroup *) Xp_alloc(DG,SEL_OR_MOD);
dgr->dg_devlst.seq_card = 3;
dgr->dg_devlst.seq_ptr = (char*)seq;

```

When using a sequence structure within a *modify* structure the following convention is used: the first element is an action tag that is not a member of the sequence itself. This element may take three values, either a char (for example, "+") for a sequence of character elements, or a string (for example "+") for the sequence of string elements. This element affects the corresponding sequence of selected objects in the way described below (the second and subsequent elements form the modify sequence).

- + indicates that the modify sequence is to be appended to the existing sequence in all selected objects.
- indicates that the modify sequence specifies elements that are to be removed from the sequence within

selected objects.



If elements in the modify sequence are not present in the selected objects, these elements are simply ignored.

=

indicates that the modify sequence is to replace the existing sequence in the selected objects.

Example

```
struct Xp_dgroup *dgr;
static char * seq[3] = {"+", "dv1", "dv2"};
dgr = (struct Xp_dgroup *) Xp_alloc(DG, SEL_OR_MOD);
dgr->dg_devlst.seq_card = 3;
dgr->dg_devlst.seq_ptr = (char*)seq;
```

The above structure used as a modify structure will append the devices *dv1* and *dv2* to the device list of the selected printer groups.

3.2.5 Scaled value structures

Some object definition parameters require scaled values. For example, the page height and the line length of a form object are specified using scaled values.

A scaled value structure is defined as follows:

```
struct_scaled_value
{ u_short value;
  □□u_short unit;
};
```

The *unit* field contains the unit in which the numerical value of the *value* field is given. This unit may be specified with XP_COLUMN (columns), XP_LINE (lines), XP_HUNDREDTH_OF_CENTIMETER (hundredths of a centimeter) or XP_HUNDREDTH_OF_INCH (hundredths of an inch).

The *value* field contains the numerical values of the scaled value. To pass on a decimal value (in inches, centimeters or millimeters), the value must be multiplied according to the unit: for centimeters by 100, for millimeters by 10 in order to obtain the value in hundredths of a centimeter (XP_HUNDREDTH_OF_CENTIMETER); for inches by 100 in order to obtain the value in hundredths of an inch (XP_HUNDRETH_OF_INCH). For example: to obtain the value 2.5 cm, you would have to specify it as 250 hundredths of a centimeter (i.e. with the values 250 for *value* and XP_HUNDREDTH_OF_CENTIMETER for *unit*).

If the numerical value is given as a number of columns (XP_COLUMN) or as a number of lines (XP_LINE), the value specified in the *value* field must be an integer and corresponds to the desired value. In this case, *value* can be specified directly.

Example:

```
struct Xp_form *frm;
frm = (struct Xp_form *) Xp_alloc(FM, ADDITION);
frm->fm_paper_size[0].value = 1100;
frm->fm_paper_size[0].value = XP_HUNDRETH_OF_INCH;
frm->fm_paper_size[1].value = 850;
frm->fm_paper_size[1].value = XP_HUNDRETH_OF_INCH;
```

These structures are used to define a page height of 11 inches and a page width of 8.5 inches (letter paper size).

3.2.6 Parameter structures

All the functions of the Xprint API that act on the database or spool queues have a parameter with the label *params* in their function call parameter list. This parameter is a pointer to a structure that provides additional or supplementary information for the function. This convention for the API functions avoids too many parameters

in the function call itself.

Not all the Xprint API functions currently use the *params* parameter. In these cases the parameter is reserved for future use and a NULL pointer should be passed.

Each function has its own parameter structure. The full set of structures is defined in *xprint.h* and *client.h*. The definitions and structure identifiers are also reproduced with each function description in the [Chapter "Function descriptions"](#). To allocate memory for a parameter structure, the function *Xp_alloc()* should be used, passing the parameter structure identifier as the type parameter. The allocated structure will be initialized with all fields empty.

3.2.7 Result structures

The functions that act on the database have a uniform method of providing the result of the actions carried out. The return value is always a positive integer value (type *short*) that specifies the size of a result array. The result array itself has one of two structures, depending on the function, and is given for each function in the [Chapter "Function descriptions"](#).

The structures are defined in *client.h* and reproduced below.

```
struct struct_ret_code
{ short ret_code;
  short reserved;
  string msg;
}
```

where *msg* contains a pointer to a message and *ret_code* a corresponding code. The actual values are dependent on the particular function called.

The field *msg* is a pointer to a message. The message describes the result of the function for the selected object in the national language selected for Xprint use. These messages can be used by the application program to report the action of the function call. The field *ret_code* is a message code, corresponding to the message in *msg*. This code can be used to identify the particular message and therefore can be the result of the function (see [Section "List of messages"](#) in the appendix to this guide). This code is unique for each message and remains the same regardless of national language.

```
struct disp_mdl
{ struct struct_ret_code diag;
  char *disp_obj; /* pointer to a structure containing the
  result of a show or get state request */
}
```

where *disp_obj* is a pointer to a structure containing *show* or *status* information for one selected entry. Each element of the array of *disp_mdl* structures corresponds to one selected object. The *disp_obj* pointer is only valid if there was no error detected by the function in carrying out its operation on this entry. The actual structure pointed to by *disp_obj* is dependent on the object type, and is independent of the function. Which structure can be assigned to each object type is listed in the table at the beginning of this chapter. For *status* functions, the returned structure is given in the "sta structure" column. For *show* functions, the returned structure is the object data structure itself. The definition of the *sta* structures is contained in *client.h* and that of the other structures in *xprint.h*, and they are reproduced in the appendix.

Success of the function call for each job is indicated by *ret_code*. A value of OKAY shows success. If not OKAY, the value is the error code for the particular error detected.

If an error is detected, *msg* will contain a pointer to the error message corresponding to the message code in the national language selected for Xprint use. Error messages are listed in the [Section "List of messages"](#) in the appendix.

Using selection criteria that result in more than one object being selected will cause the array of *struct_ret_code* or *disp_mdl* structures to contain multiple elements, one for each selected object. Processing such structures will generally be more complex than processing a single selected element.

Where selection criteria fail to select any object, the result array will contain one entry with an error message

to indicate that no object has been selected.

4 Memory management

Xprint V6.0 API provides four functions to help application programs to manage memory in a convenient and secure manner. These functions are summarized in the following table:

Function	Brief description
<code>Xp_alloc()</code>	Used to allocate and initialize memory for use by object structures or parameter structures
<code>Xp_free()</code>	Used to free memory allocated by <code>Xp_alloc()</code>
<code>Xp_clean()</code>	Used to reinitialize a structure already allocated by <code>Xp_alloc()</code>
<code>release_cli()</code>	Must be called to release memory allocated by Xprint functions for the result structure

Table 4: Functions to manage memory

For memory allocation associated with Xprint V6.0 API functions, explicit and implicit memory allocation are distinguished.

Explicit memory allocation is an allocation method for use with object structures or parameter structures (see [Section "Object structures"](#) and [Section "Parameter structures"](#) in [Chapter "Preface"](#)).

Implicit memory allocation occurs when a call to an Xprint V6.0 function creates a structure to return the results of the function call (see [Section "Result structures"](#) in [Chapter "Preface"](#)).

4.1 Explicit memory allocation

Explicit structures are always allocated by calling the function `Xp_alloc()`. The function is passed two parameters. One parameter defines the structure type and the other defines the intended use (add, modify or select) of the structure. This enables `Xp_alloc()` to allocate the correct size structure and to initialize the structure for its intended use.

Explicitly allocated memory must be freed by calling the function `Xp_free()`. This function will release exactly the memory allocated by the corresponding `Xp_alloc()`. Any substructures which might have been linked with an object structure by the application code are not freed with `Xp_free()` (especially sequence structures). It is the responsibility of the application to free, or otherwise manage, the memory for substructures.

Xprint V6.0 provides a further function for use with explicitly allocated memory; this is the `Xp_clean()` function. This function will initialize a structure ready for a new intended use. This new use may be the same as when originally allocated by `Xp_alloc()` or may be a different use, but the structure type must remain the same. Any pointers to sub-structures that the application may have attached to the original use of the structure will be lost by calling `Xp_clean()`. The application should either free such substructures or otherwise manage the memory they occupy.



Parameter structures are always initialized to all zero bits.

4.2 Implicit memory allocation

Implicit memory allocation occurs within Xprint V6.0 API functions. The allocation is necessary to hold structures used to return the result of the function. The address of these structures is passed to the application in the variable `rc_tab`. Implicitly allocated memory must be freed by the application code at the earliest safe point by calling the function `release_cli()`. The contents of implicitly allocated memory should be considered read only and NOT be modified by the application code.



`release_cli()` releases exactly the memory allocated within the particular function. This includes all elements of the `rc_tab` array and substructures addressed from this array.

5 Building an application

Building a Xprint application involves the two stages of compilation and linkage.

Required header files

The modules of an application program are compiled in the standard way using the appropriate supported compiler(s). Modules of an application that call the API functions or reference the Xprint data structures must include the two Xprint header files. This is achieved by including the following two lines at the beginning of the modules:

```
#include <xprint.h>
#include <client.h>
```

When running the compiler, the search path for the header files must be set to include the subdirectory within the Xprint directory containing the above two header files. The header files are always located in the following directory (even if Xprint was installed in a directory other than */var/spool/Xprint*):

/var/spool/Xprint/config/include/

Required libraries

Compiled modules of the application program must be linked with the Xprint libraries. To ensure full binary compatibility between the Xprint versions, the libraries have been renamed in version 5.0:

Library	Description
libXpCLI	contains all interfaces described in this manual
libXpNET	contains all network services used by Xprint

Table 5: Description of libraries

All these libraries are located in the following directory:

/var/spool/Xprint/config/lib

To simplify linkage, the library search path should be set as follows beforehand:

```
export LD_RUN_PATH=/var/spool/Xprint/config/lib:$LD_RUN_PATH
```

Compiling an application

Actual compilation of an Xprint application can then be effected as follows:

```
cc -I/var/spool/Xprint/config/include \
   -L/var/spool/Xprint/config/lib -lXpCLI \
   -lXpNET
```

The function *xpstart_cli* must be executed under *root* with the access rights 500 (-r-x-----).

6 Function descriptions

This chapter describes the functions of the Xprint program interface. The data structures referenced in the function descriptions are described in the appendix. Since the *param* data structure is unique to each function, these definitions are reproduced with each function description in this chapter. The actual definitions for the *param* data structure are located in the header file *client.h*.

The functions in this chapter are arranged alphabetically.

Many of the functions have equivalent commands for the Xprint administrator. The administrator commands provide examples of how to use the API and an appreciation of these may help in the understanding of the API functions. Most of the given examples are available in the directory */var/spool/Xprint/config/goodies/samples* and can be compiled directly after Xprint has been installed. The *readme* file in this directory represents the link between the command options and the corresponding fields in the data structures. In addition a list of the possible values is displayed for some fields.



The Xprint messages contained in the "Return values" sections of this chapter do not include the message identifier, only the message text (example: Job <%u> (<%s.128s>).

See the [Section "List of messages"](#) in the appendix for a full list of Xprint messages with relevant identifiers.

6.1 add_all_cli - adding an object

This function adds an object to the Xprint database. Object types job and system cannot be added with this function. On successful completion of this function, the object will be available for immediate use within the Xprint configuration.

When the object type is PL (Printer Capability List) or RR (Recovery Rules), the file containing the statements will be compiled within the *add_all_cli()* function.

Except for the object user only one object can be added at any one time; only with the object user, is it possible to add more than one at a time.

Synopsis

```
#include <xprint.h>
#include <client.h>
short add_all_cli (typ,obj,params,rc_tab)
unsigned short typ;
union Xp_object *obj;
struct add_all_params *params;
struct struct_ret_code **rc_tab;
```

where:

typ is the identifier for the object type to be added; this can be: DV (printer), DG (printer group), FL (filter), FT (font), FM (form), GW (gateway), HO (host), JT (job template), LR (loadable resource), PL (PCL), PN (permission list), RR (recovery rules), SP (supervisor), SV (server) or US (user). The [Table ""](#) contains a complete list of identifiers.

obj is a pointer to the structure, allocated as an add structure by *Xp_alloc()*, containing the description for the particular instance of the object to be added. The structure is dependent on the object type. The appropriate structure for each object type is listed in the [Section "xp commands and corresponding structures"](#).

params is not used in this release of Xprint and should be set to the NULL pointer.

rc_tab is used by *add_all_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and, except for when adding a user, contains only one element.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array (i.e. one for all object types except users).

The *rc_tab* structure has each element of its array set to show success or failure of the function executed for the corresponding object in the *obj* structure. On successful adding of an object, *ret_code* is set to the message identifier CL_OBADD and the *msg* field to the corresponding message <%.128s> added <%.128s> (where both *%.128s* variables were already replaced by the object names). Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

In this example, a printer of the type HP-Laserjet is added to the spooler database. Specifying the name of the printer and the physical device address is optional; default values are used for the other values.

```
#include "stdio.h"
#include "sys/types.h"
#include <xprint.h>
#include <client.h>
/*
 * Add an HP LaserJet device to the spooler database
 * xpadd -dev <device name> -da <device address> -pc HP-LASERJET
 * -ca <comment>
 * The name of the device and the physical device address are
 * optional. Default values are used for the other options
 */
main(argc,argv)
    int argc;
    char **argv;
{
    union Xp_object *obj;          /* object structure used as add structure */
    struct struct_ret_code *rc_tab; /* result array */
    short rc_num,i;
    /* Collect arguments : device name - device address */
    /* ----- */
    if (argc < 3) {
        printf("Usage %s <device name> <device address>\n",argv[0]);
        exit (1);
    }
    /* Allocate an add structure and fill it */
    /* ----- */
    obj = (union Xp_object *) Xp_alloc(DV,ADDITION);
    obj->dv.dv_name = argv[1];
    obj->dv.dv_pcl = "HP-LASERJET";
    obj->dv.dv_add = argv[2];
    obj->dv.dv_comment[CM_ADMIN] = "hp printer on 1st floor
    /double side";
    /* Call Xprint for adding the device */
    /* ----- */
    rc_num = add_all_cli(DV,obj,NULL,&rc_tab);
    /* Display the results */
    /* ----- */
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
    }
}
```

```

    printf("corresponding return code: %d\n",
           rc_tab[i].ret_code);
}
/* Free allocated space */
/*-----*/
release_cli((char *) rc_tab);
Xp_free(obj);
}

```

6.2 add_job_cli - adding a job

This function submits one job to the spool queue. The concept is to 'add' the job object to the system from where Xprint will schedule printing. A job may specify more than one file. The job identifier returned by this function can be used when interrogating or altering the progress of the job.

Synopsis

```

#include <xprint.h>
#include <client.h>
short add_job_cli (job,params,rc_tab)
struct Xp_job *job;
struct add_job_params *params;
struct struct_ret_code **rc_tab;

```

where:

job is a pointer to an *Xp_job* structure, allocated as an add structure by *Xp_alloc()*, containing the description for the particular instance of the job to be added.

params is either a NULL pointer (meaning no job template) or a pointer to an *add_job_params* structure containing job template details. This structure must be allocated using the *Xp_alloc()* function.

rc_tab is used by *add_job_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains just one element, i.e. only one job can be added for each call of this function.



The calling code must free the *rc_tab* structure using the *release_cli()* function.

The *add_job_params* structure has the following definition:

```

Identifier:  ADD_JOB_PARAMS
struct add_job_params {
    string version;    /* filled by Xp_alloc() */
    string job_template;
};

```

The *job_template* field provides the name of a job template to be applied to the added job. From within API only the system templates can be used. Values supplied in the *job* data structure will take precedence over the template; i.e. the template values are only used where the corresponding fields in the *job* structure are left empty. Refer to the "Xprint V6.0 User and Administrator Guide" for more information on job templates.

Return values

The return value is the number of elements in **rc_tab*. Since only one job may be added at a time, this value is always 1.

The *rc_tab* structure shows success or failure of the function. A successfully added job will have *ret_code* set to the message identifier CL_JOADD and the *msg* field set to the corresponding message Job <%u> (<%.128s>) added (where %u and %.128s have already been substituted by the job identifiers allocated by the *add_job_cli()* function). Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

The structure *Xp_job* supplied by the calling code will have the substructure *jo_id* set to the job identifier if *add_job* was successful. The values of this substructure are not defined if the *add_job* was not successful.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

In this example, a print job is printed on a specified printer using the default values. The job may consist of a list of documents.

```
#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Submit a job to a specific device.
 * xpadd -de <device> [-dr <list of documents>]
 * The job may consist of a list of documents. Stdin is
 * considered when no document name is given.
 * The job is printed with all defaulted options.
 */
main (argc, argv)
□□□□int argc;
□□□□char **argv;
□□□□char *stdinptr = "";      /* stdin when no document */
□□□□struct Xp_job *job;      /* add structure */
□□□□struct struct_ret_code *rc_tab; /* result array */
□□□□short rc_num,i;
□□□□/* Collect arguments : use stdin when no document */
□□□□/*-----*/
□□□□if (argc < 2) {
□□□□□□printf("Usage %s <destination> [<list of
□□□□□□documents>]\n",argv[0]);
□□□□□□exit (1);
□□□□}
□□□□/* Allocate an add structure and fill it */
□□□□/*-----*/
□□□□job = (struct Xp_job *) Xp_alloc(JO,ADDITION);
□□□□/* destination */
□□□□/*-----*/
□□□□job->jo_schedul.sch_destin = argv[1];
□□□□/* list of documents */
□□□□/*-----*/
□□□□if (argc > 2) {
□□□□□□job->jo_document.doc_doc_ref.seq_card = argc - 2;
□□□□□□job->jo_document.doc_doc_ref.seq_ptr = (char*)&argv[2];
□□□□}
□□□□else {
□□□□□□job->jo_document.doc_doc_ref.seq_card = 1;
□□□□□□job->jo_document.doc_doc_ref.seq_ptr = (char*)&(stdinptr);
□□□□}
□□□□/* Call Xprint for adding the job */
□□□□/*-----*/
□□□□rc_num = add_job_cli(job,NULL,&rc_tab);
□□□□/* Display the results */
□□□□/*-----*/
```

```

for (i=0; i<rc_num; i++) {
    printf("%s",rc_tab[i].msg);
    printf("corresponding return code : %d\n",
    rc_tab[i].ret_code);
}
/* Free allocated space */
/* ----- */
release_cli((char *) rc_tab);
Xp_free(job);
}

```

6.3 chg_dev_cli - changing printer status

This function requests Xprint to change the status of one or more selected printers.

Synopsis

```

#include <xprint.h>
#include <client.h>
short chg_dev_cli (dev,params,rc_tab)
struct ECM_printer *dev;
struct chg_dev_params *params;
struct struct_ret_code **rc_tab;

```

where:

dev is a pointer to an *ECM_printer* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the set of devices whose status is to be changed. Refer to the [Section "Object structures" in Chapter "Preface"](#) for a description of the use of select structures.

params is a pointer to a *chg_dev_params* structure. This structure must be allocated by *Xp_alloc()* and set up to define the required change in status, which will be applied to all selected printers. See the explanation below for details on the use of this structure.

rc_tab is used by *chg_dev_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each object defined in the *dev* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

The definition of the structure *chg_dev_params* is reproduced below.

Identifier: CHG_DEV_PARAMS

```

struct chg_dev_params {
    string version; /* filled by Xp_alloc() */
    string cur_form; /* xpchange -dev -mf form_name */
    string justification; /* xpchange -dev -ju */
    u_short new_priority; /* xpchange -dev -pr */
    v_short change_type; /* (see manual) */
    v_short new_state; /* xpchange -dev -st */
    v_short new_spoolin; /* xpchange -dev -si */
    v_short new_spoolout; /* xpchange -dev -so */
    v_short when_tbd; /* xpchange -dev -wn */
    v_short form_manip; /* mount or unmount (see manual) */
    string dv_redirect; /* xpchange -dev -rd #2 */
};

```

This structure is used as follows:

Field	Set to	Comments
cur_form	Specifies the name	

	of a form to be mounted (manual method)	
justification	String	Explanatory comment for status change
new_priority	New priority value	
change_type		Controls the overall action of the requested change
	CH_STATE	Requests change of status. The following fields become significant: new_state; new_spoolout; justification and when_tbd
	CH_FORM	Requests action to be taken on form mounting. The following fields become significant: form_manip; and cur_form
	CH_STOP_AFP	Printer will stop after printing the first page
	CH_UNTIL_END	Resume printing (to end of job)
	CH_NEXT_PAGE	Print next page
	CH_AGAIN	Reprint current page
	CH_PRIORITY	Requests a change to priority. The field new_priority provides the new value
new_state	CH_ACTIVECH_N OT_ACTIVE	Sets printer(s) to ACTIVE Sets printer(s) to NOT ACTIVE
new_spoolin	CH_ON or CH_OFF	Sets spoolin to ON or OFF
new_spoolout	CH_ON or CH_OFF	Sets spoolout to ON or OFF
when_tbd		Defines point at which state change is to be effective. Only relevant in conjunction with new_state set to CH_ACTIVE and/or spoolout set to OFF
	WT_END	At end of current job
	WT_LAST	When all currently active and waiting jobs have been printed
	WT_NOW	Immediately
	WT_DELETE	Immediately and delete current job
form_manip	CH_MOUNT_FOR MCH_UNMOUNT_F ORM	Sets manual form mode. Used with "manual method" form loading Removes "manual method" form mode

Table 6: Use of the structure chg_dev_params

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding printer in the *dev* structure. A successfully modified printer object will have *ret_code* set to the message identifier CL_OBCHG and the *msg* field set to the corresponding message (<%*.128s*>) changed (where <%*.128s*> has already been substituted by the printer name. Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example changes the status of specific printers. Specifying the name of the host is optional.

```
#include "stdio.h"
#include "sys/types.h"
#include <xprint.h>
#include <client.h>
/*
 * Change the state of a set of controlled devices.
 * xpchange -dev dv\* dev\* -st <state> [-scl -ho <host>]
 * The new state is given as argument. The name of a host can be
 * specified (optional). The name of the devices are known by
 * the application.
 */
main(argc,argv)
    int argc;
    char **argv;
{
    struct ECM_printer *dev; /* select structure */
    struct chg_dev_params *params; /* parameter structure */
    struct struct_ret_code *rc_tab; /* result array */
    char state; /* requested device state */
    short rc_num,i;
    /* Collect arguments : new state [ - host ] */
    /* ----- */
    if (argc < 2) {
        printf("Usage %s <state {ACTIVE/NOT_ACTIVE}> [<host>] \n",
            argv[0]);
        exit(1);
    }
    state = CH_ACTIVE;
    if (strcmp (argv[1],"NOT_ACTIVE") == 0) {
        state = CH_NOT_ACTIVE;
    }
    /* Allocate a select structure and build the selection
     * criteria */
    /* ----- */
    /* ----- */
    dev = (struct ECM_printer *) Xp_alloc(DV,SEL_OR_MOD);
    dev->dv_name = "dv* dev*";
    if (argc > 2) {
        dev->dv_host = argv[2];
    }
    /* Allocate a parameter structure and fill it */
```

```

    /*-----*/
    params = (struct chg_dev_params *)
    Xp_alloc(CHG_DEV_PARAMS,0);
    params->change_type = CH_STATE;
    params->new_state = state;
    /* Call Xprint for changing the state of devices */
    /*-----*/
    rc_num = chg_dev_cli(dev,params,&rc_tab);
    /* Display the results */
    /*-----*/
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code : %d\n",
            rc_tab[i].ret_code);
    }
    /* Free allocated space */
    /*-----*/
    release_cli((char*)rc_tab);
    Xp_free(dev);
    Xp_free(params);
}

```

6.4 chg_dgr_cli - changing printer group status

This function requests Xprint to change the status of one or more selected printer groups.

Synopsis

```

#include <xprint.h>
#include <client.h>
short chg_dgr_cli (dgr,params,rc_tab)
struct Xp_dgroup *dgr;
struct chg_dgr_params *params;
struct struct_ret_code **rc_tab;

```

where:

dgr is a pointer to an *Xp_dgroup* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the set of device groups whose status is to be changed. Refer to the [Section "Object structures"](#) in ["Chapter Preface"](#) for a description of the use of select structures.

params is a pointer to a *chg_dgr_params* structure. This structure must be allocated by *Xp_alloc()* and set up to define the required change in status, which will be applied to all selected printer groups. See the explanation below for details on the use of this structure.

rc_tab is used by *chg_dgr_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each device group object selected by the *dgr* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

The structure *chg_dgr_params* is defined as follows:

```

Identifier:  CHG_DGR_PARAMS
struct chg_dgr_params {
    string version;      /* filled by Xp_alloc() */
    string justification; /* xpchange -dgr -ju */
    char new_spoolin;    /* xpchange -dgr -si */
};

```

This structure is used as follows:

Field	Set to	Comments
justification	string	Explanatory comment
new_spoolin	CH_ON or CH_OFF	Sets spoolin status to ON or OFF respectively

Table 7:

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding device group object selected by the *dgr* structure. A successfully changed printer group object will have the *ret_code* field set to the message identifier CL_OBCHG and the *msg* field set to the corresponding message <%.128s> changed (where %.128s has already been replaced by the device group name). Any other value in *ret_code* is an error message identifier, *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example enables spoolin for all the device groups containing at least the printers "dv1" and "dv2" in their device list.

```
#include <xprint.h>
#include <client.h>
main()
{
    struct Xp_dgroup *dgr; /* select structure */
    struct chg_dgr_params *params; /* parameter structure */
    struct struct_ret_code *rc_tab; /* result array */
    short rc_num i;
    static char* seq[3] = { "+", "dv1", "dv2" };
    /* Allocate a select structure and build the selection
    criteria */
    /* _____ */
    dgr = (struct Xp_dgroup *) Xp_alloc(DG_SEL_OR_MOD);
    dgr->dg_devlst.seq_card = 3;
    dgr->dg_devlst.seq_ptr = (char*)seq;
    /* Allocate a parameter structure and fill it */
    /* _____ */
    params = (struct chg_dgr_params *) Xp_alloc(CHG_DGR_PARAMS, 0);
    params->new_spoolin = CH_ON;
    /* Call Xprint for changing the spoolin of device groups */
    /* _____ */
    rc_num = chg_dgr_cli(dgr, params, &rc_tab);
    /* Display the results */
    /* _____ */
    for (i=0; i<rc_num; i++)
    {
        printf("%s", rc_tab[i].msg);
        printf("corresponding return code: %d\n",
            rc_tab[i].ret_code);
    }
    Xp_free(params);
}
```

```

Xp_free(dgr);
release_cli((char*) rc_tab);
}

```

6.5 chg_gtw_cli - changing gateway status

This function changes the status of one or more gateways.

Synopsis

```

#include <xprint.h>
#include <client.h>
short chg_gtw_cli (gtw,params,rc_tab)
struct Xp_gateway *gtw;
struct chg_gtw_params *params;
struct struct_ret_code **rc_tab;

```

where:

gtw is a pointer to an *XP_gateway* structure used to select the set of gateways whose status is to be changed. Objects of this structure must be allocated with the *XP_alloc* function. Replacement characters may be used in the fields of type *string*. Gateway names must be specified in the *gw_name* field and must be separated by at least one space or a tab.

params is a pointer to a structure.

```

struct chg_gtw_params
{ string version; /* filled by XP_alloc() */
  string justification; /* xpchange -gtw -ju */
  v_short new_state; /* xpchange -gtw -st (see
  below) */
  v_short new_in; /* xpchange -gtw -gi (see
  below) */
  v_short new_out; /* xpchange -gtw -st (see
  below) */
  v_short when_tbd; /* xpchange -gtw -wn */
  v_short activation; /* xpchange -gtw -sa */
}

```

The *new_state* field can contain the following values:

CH_NOT_ACTIVE (setting NOT ACTIVE)

CH_ACTIVE (setting ACTIVE)

The *new_in* field can contain the following values:

CH_OFF (receive off)

CH_ON (receive on)

The *new_out* field can contain the following values:

CH_OFF (send off)

CH_ON (send on)

rc_tab is an array of the structure *struct_ret_code*, whose address must be specified as the last argument of the function. The array contains the messages and return values indicating success or failure of the function:

- A successfully changed gateway object will have the *ret_code* field set to the message identifier CL_OBCHG and the *msg* field set to the message "%s is being changed" (where %s has already been replaced by the gateway name).
- In the case of errors or warnings the return values and messages described below in the "Error" section are used.

Return values

The function returns the number of elements in the array *rc_tab*.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

6.6 chg_job_cli - changing job status

This function requests Xprint to change the status of one or more selected jobs.

Synopsis

```
#include <xprint.h>
#include <client.h>
short chg_job_cli (job,params,rc_tab)
struct Xp_job *job;
struct chg_job_params *params;
struct struct_ret_code **rc_tab;
```

where:

job is a pointer to an *Xp_job* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the set of jobs whose status is to be changed. Refer to the [Section "Object structures" in Chapter "Preface"](#) for a description of the use of select structures.

params is a pointer to a *chg_job_params* structure. This structure must be allocated by *Xp_alloc()* and set up to define the required change in status, which will be applied to all selected jobs. See the explanation below for details on the use of this structure.

rc_tab is used by *chg_job_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each object selected by the *job* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

The structure *chg_job_params* has the following definition:

Identifier: CHG_JOB_PARAMS□

□

```
struct chg_job_params {
□□□□string version;    /* filled by Xp_alloc() */
□□□□string page_select; /* xpchange -job -ps */
□□□□char new_state;    /* xpchange -job -st */
□□□□};
```

This structure is used as follows:

Field	Set to	Comments
page_select	A string specifying the page selection criteria	Refer to the description of the <i>xpadd -job</i> command in the "Xprint V6.0 - Reference Manual" for the format of this string
new_state	CH_WAIT	Set job(s) to WAIT state
	CH_SUSPEND	Set job(s) to SUSPEND state
	CH_TOP	Set job(s) to top of spool queue

Table 8: Use of the structure *chg_job_params*

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the

corresponding job object selected by the *job* structure. A successfully changed job object will have the *ret_code* field set to the message identifier CL_JOCHG and the *msg* field set to the corresponding message Job *<u>* (*<%.128s>*) changed (where *u* (*<%.128s>*) has already been replaced by the global job identifier). Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example changes the state of print jobs from SUSPEND to WAIT. The jobs are controlled by the servers sv1 and sv2.

```
#include "stdio.h"
#include "sys/types.h"
#include <xprint.h>
#include <client.h>
/*
 * Change the state of jobs, controlled by an application, from
 * SUSPEND to WAIT
 * xpcchange <list of jobs> -st WAIT
 *      -scl " -jc application -st SUSPEND".
 * Jobs are controlled by both sv1 and sv2 servers. Jobs are
 * signed by the application that submitted them. The job comment
 * was used for that purpose
 */
main(argc,argv)
    int argc;
    char *argv;
{
    struct Xp_job *job;          /* select structure */
    struct chg_job_params *params; /* parameter structure */
    struct struct_ret_code *rc_tab; /* result array */
    short rc_num,i;
    /* Allocate a select structure and build the selection
     * criteria */
    /*-----*/
    /*-----*/
    job = (struct Xp_job *) Xp_alloc(JO,SEL_OR_MOD);
    job->jo_id.jid_jid = "sv1:* sv2:*";
    job->jo_info.j_comment = "application";
    job->jo_info.j_state = ST_SUSPEND;
    /* Allocate a parameter structure and fill it */
    /*-----*/
    params = (struct chg_job_params *)
        Xp_alloc(CHG_JOB_PARAMS,0);
    params->new_state = CH_WAIT;
    /* Call Xprint for changing the state of jobs */
    /*-----*/
    rc_num = chg_job_cli(job,params,&rc_tab);
    /*
     * Display the results */
    /*-----*/
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
    }
}
```

```

    printf("corresponding return code :
    %d\n",rc_tab[i].ret_code);
}
/* Free allocated space */
/* ----- */
release_cli((char *) rc_tab);
Xp_free(job);
Xp_free(params);
}

```

6.7 chg_spv_cli - changing supervisor status

This function requests Xprint to change the status of one or more supervisors.

Synopsis

```

#include <xprint.h>
#include <client.h>
short chg_spv_cli (spv,params,rc_tab)
struct Xp_superv *spv;
struct chg_spv_params *params;
struct struct_ret_code **rc_tab;

```

where:

spv is a pointer to an *Xp_superv* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the set of supervisors whose states are to be changed. Refer to the [Section "Object structures"](#) in [Chapter "Preface"](#) for a description of the use of select structures.

params is a pointer to a *chg_spv_params* structure. This structure must be allocated by *Xp_alloc()* and set up to define the required change in status, which will be applied to all selected supervisors. See the explanation below for details on the use of this structure.

rc_tab is used by *chg_spv_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each supervisor object selected by the *spv* structure.



The calling code must free the *rc_tab* structure using *release_cli()*. □

The structure *chg_spv_params* has the following definition:

Identifier: CHG_SPV_PARAMS

```

struct chg_spv_params {
    string version; /* filled by Xp_alloc() */
    char new_state; /* xpchange -spv -st */
    char when_tbd; /* not used */
};

```

This structure is used as follows:

Field	Set to	Comments
new_state	CH_ACTIVECH_NOT_ACTIVE	Set supervisor(s) active.Set supervisor(s) not active.

Table 9: Use of the structure *chg_spv_params*

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for each of the corresponding supervisor objects selected by the *spv* structure. A successfully changed supervisor object will have the *ret_code* field set to the message identifier CL_OBCHG and the *msg* field set to the corresponding message <%.128s> changed (where %.128s has already been replaced by the supervisor name). Any other

value in *ret_code* is an error message identifier, *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

6.8 chg_svr_cli - change server status

This function requests Xprint to change the status of one or more servers.

Synopsis

```
#include <xprint.h>
#include <client.h>
short chg_svr_cli (srv,params,rc_tab)
struct Xp_server *srv;
struct chg_srv_params *params;
struct struct_ret_code **rc_tab;
```

where:

srv is a pointer to an *Xp_server* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the set of servers whose status is to be changed. Refer to the [Section "Object structures"](#) in [Chapter "Preface"](#) for a description of the use of select structures.

params is a pointer to a *chg_srv_params* structure. This structure must be allocated by *Xp_alloc()* and set up to define the required change in status, which will be applied to all selected servers. See the explanation below for details on the use of this structure.

rc_tab is used by *chg_svr_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each server object selected by the *srv* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

The structure *chg_srv_params* has the following definition:

Identifier: CHG_SVR_PARAMS

```
struct chg_srv_params {
    □□□□string version; /* filled by Xp_alloc() */
    □□□□char new_state; /* xpchange -srv -st */
    □□□□char new_spoolin; /* xpchange -srv -si */
    □□□□char new_spoolout; /* xpchange -srv -so */
    □□□□char when_tbd; /* not used */
    □□□□char activation; /* xpchange -srv -sa */
    □□□□□□};
```

This structure is used as follows:

Field	Set to	Comments
new_state	CH_ACTIVECH _NOT_ACTIVE	Set server(s) active Set server(s) not active
new_spoolin	CH_ON orCH_OFF	Sets spoolin to ON or OFF respectively
new_spoolout t	CH_ON orCH_OFF	Sets spoolout to ON or OFF respectively
activation	XP_COLD orXP_WARM	Selects COLD or WARM activation respectively for the server(s). If this field is not set, the activation type of the local system is assumed. Activation type is only

		relevant when the server(s) are being activated by this function
--	--	--

Table 10: Use of the structure `chg_svr_params`**Return values**

The return value is the number of elements in the `rc_tab` array.

The `rc_tab` structure has each element of its array set to show success or failure of the function for the corresponding server object selected by the `srv` structure. A successfully changed server object will have this field set to the message identifier `CL_OBCHG` and the `msg` field set to the corresponding message `<%.128s>` changed (where `%.128s` has already been replaced by the server name). Any other value in `ret_code` is an error message identifier, `msg` will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the `rc_tab` structure.

6.9 chg_usr_cli - changing user status

This function requests Xprint to set new values for the consumed quotas of pages and jobs. These quotas are not automatically reset by Xprint, therefore this function must be used to reinitialize the quotas.

Synopsis

```
#include <xprint.h>
#include <client.h>
short chg_usr_cli (usr,params,rc_tab)
struct ECM_security *usr;
struct chg_usr_params *params;
struct struct_ret_code **rc_tab;
```

where:

`usr` is a pointer to an `ECM_security` structure, allocated as a select structure by `Xp_alloc()`. This structure is used to select the set of users whose states are to be changed. Refer to [Section "Object structures" in Chapter "Preface"](#), for a description on the use of select structures.

`params` is a pointer to a `chg_usr_params` structure. This structure must be allocated by `Xp_alloc()` and set up to define the required change in status, which will be applied to all selected users. See the explanation below for details on the use of this structure.

`rc_tab` is used by `chg_usr_cli()` to return the address of the structure `struct_ret_code`. This array structure is allocated by Xprint and contains one element for each user object selected by the `usr` structure.



The calling code must free the `rc_tab` structure using `release_cli()`.

The structure `chg_usr_params` has the following definition:

Identifier: `CHG_USR_PARAMS`

```
struct chg_usr_params {
    □□□□string version;      /* filled by Xp_alloc() */
    □□□□u_long cons_pages;   /* xpchange -usr -cp */
    □□□□u_long cons_jobs;    /* xpchange -usr -cj */
    □□□□};
```

This structure is used as follows:

Field	Set to	Comments
<code>cons_pages</code>	<code><numeric></code>	Provides a new quota value for number of consumed pages.

cons_jobs	<numeric>	Provides a new quota value for ☐ number of consumed jobs.
-----------	-----------	--

Table 11: Use of the structure `chg_usr_params`**Return values**

The return value is the number of elements in the `rc_tab` array.

The `rc_tab` structure has each element of its array set to show success or failure of the function for the corresponding user object selected by the `usr` structure. Any other value in `ret_code` is an error message identifier, `msg` will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the `rc_tab` structure.

6.10 cpy_all_cli - copying an object

This function requests Xprint to create a new object description in the main database by copying an existing description. The function may request Xprint to modify values for the created object during the copy process.

Synopsis

```
#include <xprint.h>
#include <client.h>
short cpy_all_cli (typ,from,to,mod,params,rc_tab)
unsigned short typ;
char *from;
char *to;
union Xp_object *mod;
struct cpy_all_params *params;
struct struct_ret_code **rc_tab;
```

where:

typ is the identifier for the object type to be copied. From the full list of object types listed in the [Table ""](#), one of the following may be specified for this function: DV (printer), DG (printer group), FL (filter), FT (font), ☐ FM (form), GW (gateway), JT (job template), LR (loadable resource), ☐ PN (permission list), SP (supervisor), SV (server), and US (user).

from is a string value giving the name of the object to be copied. This name is compared for an exact match with the name field of the structure of the object type specified by *typ*.

to is a string value specifying the name of the object being created as a result of the copy. No two objects of the same type can have the same name in the database. Therefore it is a minimum requirement that a new name is specified to be given to the object being created.

mod is a pointer to a structure allocated as a modify structure by `Xp_alloc()`. The structure type must correspond to the type defined by *typ*.

params is not used in this release of Xprint and should be set to the NULL pointer.

rc_tab is used by `cpy_all_cli()` to return the address of the structure `struct_ret_code`. This array structure is allocated by Xprint and contains one element.



The calling code must free the `rc_tab` structure using `release_cli()`.

Return values

The return value is the number of elements in the `rc_tab` array, i.e. 1.

The `rc_tab` structure has the only element of its array set to show success or failure of the function. A successfully copied object will have this field set to the message identifier `CL_OBCPY` and the `msg` field set to the corresponding message `<%.128s>` copied into `<%.128s>` (where both `%.128s` tokens have already been

replaced by the object names). Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example creates a device definition on the basis of an existing device but changing the name and the address.

```
#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Duplicate a device definition, changing its name and device
 * address
 * xpcopy <device1> <device2> -da <device address>
 * Both device names and the new physical device address are
 * optional.
 */
main (argc, argv)
    int argc;
    char **argv;
{
    union Xp_object *obj;          /* Xprint object
    structure
    used as modification
    structure */
    struct struct_ret_code *rc_tab; /* result array */
    short rc_num,i;
    /* Collect arguments : use stdin when no document */
    /*-----*/
    if (argc < 4) {
        printf("Usage %s <device1><device2><deviceaddress>\n",
            argv[0]);
        exit (1);
    }
    /* Allocate a modify structure and fill it */
    /*-----*/
    obj = (union Xp_object *) Xp_alloc(DV,SEL_OR_MOD);
    obj->dv.dv_add = argv[3];
    /* Call Xprint for a device copy */
    /*-----*/
    rc_num = cpy_all_cli(DV,argv[1],argv[2],obj,NULL,&rc_tab);
    /* Display the results */
    /*-----*/
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code : %d\n",
            rc_tab[i].ret_code);
    }
    /* Free allocated space */
    /*-----*/
}
```

```

□□□□release_cli((char *) rc_tab);
□□□□Xp_free(obj);
}

```

6.11 del_all_cli - deleting objects

This function requests Xprint to delete one or more selected objects. Note that Xprint will not permit the font object with the name "GENERIC" or the form object with the name "GENERIC" or the pcl object with the name "GENERIC" to be deleted.

Synopsis

```

#include <xprint.h>
#include <client.h>
short del_all_cli (typ,obj,params,rc_tab)
unsigned short typ;
union Xp_object *obj;
struct del_all_params *params;
struct struct_ret_code **rc_tab;

```

where:

typ is the identifier for the object type to be deleted. From the full list of object types listed in [Table ""](#), one of the following may be specified for this function: DV (printer), DG (printer group), FL (filter), FT (font), □ FM (form), GW (gateway), JT (job template), LR (loadable resource), □ PL (PCL), PN (permission list), RR (recovery rules), SP (supervisor), □ SV (server), and US (user).

obj is a pointer to the structure, allocated as a select structure by *Xp_alloc()*, containing the selection criteria by which the objects will be selected. Only objects of the type *typ* will be searched for and *obj* must be the corresponding structure. The appropriate structure for each object type is listed in [Table ""](#) (in [Chapter "Preface"](#)).

params is not used in this release of Xprint and should be set to the NULL pointer.

rc_tab is used by *del_all_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each object selected by the *obj* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding object selected by the *obj* structure. Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example deletes all the device groups defined with at least dv1 and dv2.

```

#include "stdio.h"
#include "sys/types.h"
#include <xprint.h>
#include <client.h>
/*
□ * Delete all device groups defined with at least dv1 and dv2
□ * xpdcl -dgr -scl -dl "dv1 dv2"

```

```

/*
main(argc,argv)
    int argc;
    char **argv;
{
    union Xp_object *obj;          /* object structure used as
    select structure */
    struct struct_ret_code *rc_tab; /* result array */
    short rc_num,i;
    static char * seq[3] = { "+","dv1","dv2" };
    /* Allocate a select structure and build the select
    criteria */
    /*-----*/
    /*-----*/
    obj = (union Xp_object *) Xp_alloc(DG,SEL_OR_MOD);
    obj->dg.dg_devlst.seq_card = sizeof(seq) / sizeof(char *);
    obj->dg.dg_devlst.seq_ptr = (char *)seq;
    /* Call Xprint for deleting the device groups */
    /*-----*/
    rc_num = del_all_cli(DG,obj,NULL,&rc_tab);
    /* Display the results */
    /*-----*/
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code : %d\n",
        rc_tab[i].ret_code);
    }
    /* Free allocated space */
    /*-----*/
    release_cli((char *)rc_tab);
    Xp_free(obj);
}

```

6.12 del_job_cli - deleting jobs

This function requests Xprint to delete one or more selected jobs from one or more spool queues.

Synopsis

```

#include <xprint.h>
#include <client.h>
short del_job_cli (job,params,rc_tab)
struct Xp_job *job;
struct del_job_params *params;
struct struct_ret_code **rc_tab;

```

where:

job is a pointer to an *Xp_job* structure, allocated as a select structure by *Xp_alloc()*, containing the selection criteria by which the jobs will be selected. Refer to the [Section "Object structures"](#) in [Chapter "Preface"](#) for a description of the use of select structures.

params is not used in this release of Xprint and should be set to the NULL pointer.

rc_tab is used by *del_job_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each job selected by the *job* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding job selected by the *job* structure. A successfully deleted job will have the *ret_code* field set to the message identifier CL_JODEL and the *msg* field set to the corresponding message Job <%.128s> (<%.128s>) deleted (where <%.128s> (<%.128s>) has already been replaced by the global job identifier). Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example deletes all jobs controlled by an application and which have the status SUSPEND.□

All jobs (meeting the requirements above) of the desired server are deleted; if you do not specify a server name, all jobs meeting the requirements above are deleted.

```
#include "stdio.h"
#include "sys/types.h"
#include <xprint.h>
#include <client.h>
/*
□ * Delete all jobs controlled by an application and that are in
□ * SUSPEND state
□ * xpdel <list of jobs> -scl " -st SUSPEND -jc application "
□ * All jobs for a given server are considered. When no server is
□ * specified, all servers are considered.
□ * Jobs are signed with the name of an application
□ */
main(argc,argv)
□□□□int argc;
□□□□char *argv;
{
□□□□struct Xp_job *job;          /* select structure */
□□□□struct struct_ret_code *rc_tab; /* result array */
□□□□char jids[128];             /* list of jids */
□□□□short rc_num,i;
□□□□/* Collect arguments : <server> */
□□□□/* ----- */
□□□□if (argc < 2) {
□□□□□□□□printf(jids, ".*");
□□□□}
□□□□else {
□□□□□□□□printf(jids, "%s:.*",argv[1]);
□□□□}
□□□□/* Allocate a select structure and build the selection
□□□□□□□□criteria */
□□□□/*-----*/
□□□□□□□□-----*/
□□□□job = (struct Xp_job *) Xp_alloc(JO,SEL_OR_MOD);
□□□□job->jo_id.jid_jid = jids;
□□□□job->jo_info.j_comment = "application";
□□□□job->jo_info.j_state = ST_SUSPEND;
□□□□/* Call Xprint for deleting the jobs */
```

```

    /*-----*/
    rc_num = del_job_cli(job,NULL,&rc_tab);
    /* Display the results */
    /*-----*/
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code : %d\n",
            rc_tab[i].ret_code);
    }
    /* Free allocated space */
    /*-----*/
    release_cli((char *) rc_tab);
    Xp_free(job);
}

```

6.13 mod_all_cli - modifying objects

This function requests Xprint to modify one or more object descriptions in the main database. This function does not apply to the job object.

Where the modified object is of type PL (Printer Capability List) or RR (Recovery Rules) and the statements file field is modified, then the statements file will be compiled within the *mod_all_cli()* function.

Synopsis

```

#include <xprint.h>
#include <client.h>
short mod_all_cli (typ,obj,mod,params,rc_tab)
unsigned short typ;
union Xp_object *obj;
union Xp_object *mod;
struct mod_all_params *params;
struct struct_ret_code **rc_tab;

```

where:

typ is the identifier for the object type to be modified. All object types except job objects are allowed; a full list of object types can be found in [Table ""](#). Here, object types specified for this function may be the following: DV (printer), DG (printer group), FL (filter), FT (font), FM (form), GW (gateway), HO (host), JT (job template), LR (loadable resource), PL (PCL), PN (permission list), RR (recovery rules), SP (supervisor), SV (server), and US (user) or SY (system).

obj is a pointer to the structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the objects in the main database to be modified. The structure must be compatible with the object type defined by *typ*.

mod is a pointer to a structure of the same type as *obj*, allocated as a modify structure by *Xp_alloc()*. This structure is used to define the modifications to be made to the selected objects. Refer to the [Section "Object structures"](#) in [Chapter "Preface"](#) for a description of the use of modify structures.

params is only defined when the object type is US (i.e. user). For all other types, *params* should be set to the NULL pointer. For type US, *params* can be NULL, or a pointer to a structure *mod_all_params*. If *params* is different from NULL for the user object, *mod* is ignored.

rc_tab is used by *mod_all_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each object selected by the *obj* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

The structure *mod_all_params* has the following definition:

Identifier: MOD_ALL_PARAMS

```
struct mod_all_params {
    string version; /* filled by Xp_alloc()*/
    struct /*string*/ sequence host_lst; /* xpmmod -usr -cl */
};
```

This structure is used as follows:

Field	Set to	Comments
host_lst	struct /*string*/ sequence	A sequence of strings, identifying the hosts on which the user definition has to be checked.

Table 12: Use of the structure mod_all_params

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding object selected by the *obj* structure. A successfully modified object will have this field set to the message identifier CL_OBMOD and the *msg* field set to the corresponding message <%.128s> modified (where %.128s has already been replaced by the object name). Any other value in *ret_code* is an error message identifier, *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example enables polling for all printers specified with a PCL as an argument. If no PCL is specified all printers are modified accordingly:

```
#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Enable the polling for all devices of a specified pcl. Select
 * the ETX/ACK protocol.
 * xpmmod -dev -scl " -pc <pcl> " -ep -poETX
 * When no pcl is specified, all devices are considered
 */
main (argc, argv)
    int argc;
    char **argv;
{
    union Xp_object *obj; /* object structure used
    as select structure */
    union Xp_object *mod; /* modify structure */
    struct struct_ret_code *rc_tab; /* result array */
    short rc_num,i;
    /* Allocate a select structure and build the selection
    criteria */
    /*-----*/
    /*-----*/
    obj = (union Xp_object *) Xp_alloc(DV,SEL_OR_MOD);
    /* Collect arguments : pcl of devices */
```

```

    /*-----*/
    if (argc > 1) {
        obj->dv.dv_pcl = argv[1];
    }
    /* Allocate a modify structure and fill it */
    /*-----*/
    mod = (union Xp_object *) Xp_alloc(DV,SEL_OR_MOD);
    mod->dv.dv_enable_poll = TRUE;
    mod->dv.dv_proto.cn_protocol = XP_ETX;
    /* Call Xprint for a device modification */
    /*-----*/
    rc_num = mod_all_cli(DV,obj,mod,NULL,&rc_tab);
    /* Display the results */
    /*-----*/
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code : %d\n",
            rc_tab[i].ret_code);
    }
    /* Free allocated space */
    /* ----- */
    release_cli( (char *) rc_tab);
    Xp_free(obj);
    Xp_free(mod);
}

```

6.14 mod_job_cli - modifying jobs

This function requests Xprint to modify the details of one or more jobs currently in the spool queues. Job(s) currently being printed cannot be modified.

Synopsis

```

#include <xprint.h>
#include <client.h>
short mod_job_cli (job,mod,params,rc_tab)
struct Xp_job *job;
struct Xp_job *mod;
struct mod_job_params *params;
struct struct_ret_code **rc_tab;

```

where:

job is a pointer to an *Xp_job* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the set of jobs whose details are to be modified. Refer to [Section "Object structures" in Chapter "Preface"](#) for a description of the use of select structures.

mod is a pointer to an *Xp_job* structure, allocated as a modify structure by *Xp_alloc()*. This structure is used to define the modifications to be made to the selected jobs. Refer to [Section "Object structures"](#) earlier in this guide for a description of the use of modify structures.

params is not used in this release of Xprint and should be set to the NULL pointer.

rc_tab is used by *mod_job_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each object selected by the *job* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding job selected by the *job* structure. A successfully modified *job* structure will have the *ret_code* field set to the message identifier CL_JOMOD and the *msg* field set to the corresponding message Job <%.128s> (<%.128s>) modified (where <%.128s>(<%.128s>) has already been replaced by the global job identifier). Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example redirects all print jobs with the status WAITING from dv1 to dv2.

```
#include "stdio.h"
#include "sys/types.h"
#include <xprint.h>
#include <client.h>
/*
 * Modify the destination of all jobs in state WAITING on dv1 to
 * dv2
 * xpmmod -scl " -de <dv1>" -de <dv2>
 */
main(argc,argv)
    int argc;
    char **argv;
{
    struct Xp_job *job;          /* select structure */
    struct Xp_job *mod;          /* modify structure */
    struct struct_ret_code *rc_tab; /* result array */
    short rc_num,i;
    /* Collect arguments : <dv1> <dv2> */
    /* ----- */
    if (argc < 3) {
        printf("Usage %s <dv1> <dv2>\n",argv[0]);
        exit (1);
    }
    /* Allocate a select structure and build the selection criteria */
    /* ----- */
    job = (struct Xp_job *) Xp_alloc(JO,SEL_OR_MOD);
    job->jo_schedul.sch_destin = argv[1];
    job->jo_info.j_state = ST_WAIT;
    /* Allocate a modify structure and fill it */
    /* ----- */
    mod = (struct Xp_job*) Xp_alloc(JO,SEL_OR_MOD);
    mod->jo_schedul.sch_destin = argv[2];
    /* Call Xprint for modifying the jobs */
    /* ----- */
    rc_num = mod_job_cli(job,mod,NULL,&rc_tab);
    /* Display the results */
    /* ----- */
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code : %d\n",
```

```

    rc_tab[i].ret_code);
}
/* Free allocated space */
/* ----- */
release_cli( (char *) rc_tab);
Xp_free(job);
Xp_free(mod);
}

```

6.15 release_cli - releasing memory

This function requests the release of memory implicitly allocated through function calls to Xprint . Memory allocated by Xprint is used either for the *struct_ret_code* or the *disp_mdl* structure. Both of these structures are array structures. The *disp_mdl* structure contains pointers to further structures for which Xprint would also have allocated memory. The *release_cli()* function will analyse the structure being freed and ensure that all memory is also freed (see also [Chapter "Preface"](#) for more information).

Synopsis

```

#include <xprint.h>
#include <client.h>
void release_cli (ptr)
char *ptr;

```

where:

ptr is a pointer to the memory to be released. This pointer contains the address provided by Xprint in the *rc_tab* parameter of functions that return this structure.

Return values

There are no return values provided by *release_cli()*.

Errors

There are no errors returned by *release_cli()*.



An incorrect value for *ptr* will result in undefined behavior.

6.16 shw_all_cli - showing object information

This function requests the return of information for selected objects in the database.

Synopsis

```

#include <xprint.h>
#include <client.h>
short shw_all_cli (typ,obj,params,rc_tab)
unsigned short typ;
union Xp_object *obj;
struct shw_all_params *params;
struct disp_mdl **rc_tab;

```

where:

typ is the identifier for the object type for which information is being requested. One of the following object types may be specified for this function: DV (printer), DG (printer group), FL (filter), FT (font), FM (form), GW (gateway), HO (host), JT (job template), LR (loadable resource), ☐ PL (PCL), PN (permission list), RR (recovery rules), SP (supervisor), ☐ SV (server), and US (user).

obj is a pointer to a structure allocated as a select structure by *Xp_alloc()*. This structure is used to select the objects in the main database for which information is to be provided. The structure must be compatible with

the object type defined by *typ*.

params is not used in this release of Xprint and should be set to the NULL pointer.

rc_tab is used by *shw_all_cli()* to return the address of the structure *disp_mdl*. This array structure is allocated by Xprint and contains one element for each object selected by the *obj* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding object selected by the *obj* structure. Where the function is successful, the field *disp_obj* is a pointer to a structure containing the information for the corresponding selected object. The structure type is dependent on the object and will be the same as the structure for *obj*. The definitions for these structures are reproduced in the appendix.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example displays a list of definitions of selected devices.

```
#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Display the definitions of a set of controlled devices
 * xpshow -dev "dv* dev*" -di "-pc -ho -su -da"
 * A specific comment is attached to the devices controlled by the
 * application.
 * Devices have the same prefix, either dv or dev
 */
main (argc, argv)
    int argc;
    char **argv;
{
    struct disp_mdl *ptr; /* result array */
    union Xp_object *obj; /* object structure used as
    select structure */
    short i,size;
    /* Allocate a select structure and build the selection
    criteria */
    /*-----*/
    /*-----*/
    obj = (union Xp_object *) Xp_alloc(DV,SEL_OR_MOD);
    obj->dv.dv_name = "dv* dev*";
    obj->dv.dv_comment[CM_ADMIN] = "application";
    /* Call Xprint to get device definitions */
    /*-----*/
    size = shw_all_cli(DV,obj,NULL,&ptr);
    /* Display the results */
    /*-----*/
    printf("Name   pcl   Host   Supervisor Device address\n");
```

```

printf("----  ---  ---  ----- \n");
for (i=0; i<size; i++) {
    if ((ptr[i].diag.ret_code) == OKAY) {
printf("%-9.9s %-9.9s %-9.9s %-9.9s  %s\n",
    ((struct ECM_printer*)(ptr[i].disp_obj))->dv_name,
    ((struct ECM_printer*)(ptr[i].disp_obj))->dv_pcl,
    ((struct ECM_printer*)(ptr[i].disp_obj))->dv_host,
    ((struct ECM_printer*)(ptr[i].disp_obj))
->dv_supervisor,
    ((struct ECM_printer*)(ptr[i].disp_obj))->dv_add);
    }
    else printf("%s",ptr[i].diag.msg);
}
/* Release the allocated memory */
/*-----*/
release_cli((char *) ptr);
Xp_free(obj);
}

```

6.17 shw_job_cli - showing job information

This function requests the return of information for selected jobs in the Xprint queues.

Synopsis

```

#include <xprint.h>
#include <client.h>
short shw_job_cli (job,params,rc_tab)
struct Xp_job *job;
struct shw_job_params *params;
struct disp_mdl **rc_tab;

```

where:

job is a pointer to an *Xp_job* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the jobs in the spool queue for which information is to be provided. Refer to the [Section "Object structures"](#) in [Chapter "Preface"](#) for a description of the use of select structures.

params is not used in this release of Xprint and should be set to the NULL pointer.

rc_tab is used by *shw_job_cli()* to return the address of the structure *disp_mdl*. This array structure is allocated by Xprint and contains one element for each object selected by the *job* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding job selected by the *job* structure. A successful request for a job object will result in the field *ret_code* being set to the message identifier OKAY. Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message. Where the function is successful, the *disp_obj* field is a pointer to a structure containing the information for the corresponding selected job. This structure is the job object data structure *Xp_job*. The definition of this structure is reproduced in the appendix.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example displays the definition of selected print jobs. The print jobs are displayed with their global and local job identifiers, their job title, the job size and a list of possible printers.

```
#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Display the definitions of a set of jobs
 * xps show -job <list of jobs> -di "-jt" <total size>
 * potential devices>
 * All jobs are considered
 * Jobs are shown with their global and local job-id, the known
 * title, the total size of the documents to be printed, and the
 * list of printer candidates (after job validation)
 */
main (argc, argv)
int argc;
char **argv;
{
    struct disp_mdl *ptr; /* result array */
    struct Xp_job *job; /* select structure */
    short i, size;
    short n, j;
    char **destin; /* potential devices */
    /* Allocate a select structure and build the selection
     * criteria */
    /* ----- */
    /* ----- */
    job = (struct Xp_job *) Xp_alloc(JO, SEL_OR_MOD);
    job->jo_id.jid_jid = ".*";
    /* Call Xprint to get job definitions */
    /* ----- */
    size = shw_job_cli(job, NULL, &ptr);
    /* Display the results */
    /* ----- */
    printf("global jid jid title size destination\n");
    printf("----- -- --- ----\n");
    for (i=0; i<size; i++)
    {
        if ((ptr[i].diag.ret_code) == OKAY) {
            printf("%-8.8s: %-4.4s %-4.4d %-9.9s %-6.6d ",
                ((struct Xp_job*)(ptr[i].disp_obj))
                ->jo_id.jid_server,
                ((struct Xp_job*)(ptr[i].disp_obj))
                ->jo_id.jid_jid,
                ((struct Xp_job*)(ptr[i].disp_obj))
                ->jo_id.jid_local_jid,
                ((struct Xp_job*)(ptr[i].disp_obj))
                ->jo_id.jid_title,
                ((struct Xp_job*)(ptr[i].disp_obj))
                ->jo_document.doc_total_size);
            n = ((struct Xp_job*)(ptr[i].disp_obj))
```

```

XXXXXXXXXXXXXXXXXXXX->jo_schedul.sch_competitors.seq_card;
XXXXXXXXXXXXXXXXXXXXdestin =
XXXXXXXXXXXXXXXXXXXX(char **) ((struct Xp_job*)(ptr[i].disp_obj))
XXXXXXXXXXXXXXXXXXXX->jo_schedul.sch_competitors.seq_ptr;
XXXXXXXXXXXXXXXXXXXXfor (j = 0; j < n; j++) {
XXXXXXXXXXXXXXXXXXXXprintf("%-9.9s ",*destin++);
XXXXXXXXXXXX}
XXXXXXXXXXXXprintf("\n");
XXXXXXXXXXXXXXXXXXXX}
XXXXXXXXXXXXelse printf("%s",ptr[i].diag.msg);
XXXX}
XXXX/* Release the allocated memory */
XXXX/*-----*/
XXXXrelease_cli((char *) ptr);
XXXXXp_free(job);
}

```

6.18 shw_ref_cli - showing references to an object

This function displays existing references to an object.

Synopsis

```

#include <xprint.h>
#include <client.h>
short shw_ref_cli (typ,obj,params,rc_tab)
unsigned short typ;
char *obj;
struct struct_ret_code **rc_tab;

```

where:

typ is the identifier for the object type for which information is being requested. One of the following object types may be specified for this function: DV (printer), DG (printer group), FL (filter), FT (font), FM (form), GW (gateway), HO (host), JT (job template), LR (loadable resource), PL (PCL), PN (permission list), RR (recovery rules), SP (supervisor), SV (server), and US (user).

obj is a string identifying the object for which references are to be displayed.

rc_tab is used by *shw_ref_cli()* to return the identifier and types of the objects referring to the selected object. This array structure is allocated by Xprint and contains one element for each object selected by the *obj* structure.

Return values

The return value is the number of elements in the *rc_tab* array.

Example

This example displays a list of all the objects (name and type) with a reference to the font *my_font*. Only objects with a direct reference are displayed and not recursive objects.

```

#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Display the list of objects (name and type) referenced by the font
 * "my_font"
 * xpshow -fnt "my_font" -diR
 * ! Only the objects directly referencing "my_font" are displayed
 * (no recursive display).

```

```

    */
main (argc, argv)
    int argc;
    char **argv;
{
    struct shw_ref_by *return_list; /* returned list*/
    short i;
    struct struct_ret_code *ptr_diag; /* pointer to return code structure */
    struct ref_by_obj *ptr_ref; /* pointer to a reference definition */
    char *type; /* type of object*/
    struct ref_by_obj **ptr;
}

/* Call Xprint to get the list of objects */
/*-----*/
shw_ref_cli(FT,"my_font",NULL,&return_list);
ptr_diag = return_list->diag;
if ( ptr_diag->ret_code != OKAY )
    /* Print error message */
    printf ("%s", ptr_diag->msg);
else {
    /* Display title */
    /*-----*/
    printf("List of objects referencing \"my_font\" \n");
    printf("----- \n");
    /* Display the results */
    /*-----*/
    for (i=0; i< return_list->ref_list.seq_card; i++)
    {
        ptr = ((struct ref_by_obj **)return_list
        &return_list->ref_list.seq_ptr);
        ptr_ref = ptr[i];
        switch (ptr_ref->obj_type) {
            case DV: type = "device";
                break;
            case DG: type = "device group";
                break;
            case FL: type = "filter";
                break;
            case FM: type = "form";
                break;
            case FT: type = "font";
                break;
            case GW: type = "gateway";
                break;
            case HO: type = "host";
                break;
            case JT: type = "job template";
                break;
            case PL: type = "PCL";
                break;
            case PN: type = "permission";
                break;
            case RR: type = "recovery rule";
                break;
        }
    }
}

```

```

case SP: type = "supervisor";
break;
case SV: type = "server";
break;
case US: type = "user";
break;
}
printf("%s of type  %s \n", ptr_ref->obj_name, type);
}
}
/* Release the allocated memory */
/*-----*/
release_cli((char *)return_list);
}

```

6.19 sta_all_cli - status of object

This function provides the status for one or more selected objects. It is applicable only for device (DV), device group (DG), host (HO), supervisor (SP), server (SV) and user (US) objects.

Synopsis

```

#include <xprint.h>
#include <client.h>
short sta_all_cli (typ,obj,params,rc_tab)
unsigned short typ;
union Xp_object *obj;
struct sta_all_params *params;
struct disp_mdl **rc_tab;
where:

```

typ is the identifier for the object type whose status is being requested. Valid values are DV (printer), DG (printer group), HO (host), SP (supervisor), SV (server) and US (user).

obj is a pointer to a structure allocated as a select structure by *Xp_alloc()*. This structure is used to select the objects in the main database whose status is to be provided. The structure must be compatible with the object type defined by *typ*.

params is not used by this version of Xprint and should be set to the NULL pointer.

rc_tab is used by *sta_all_cli()* to return the address of the structure *disp_mdl*. This array structure is allocated by Xprint and contains one element for each object selected by the *obj* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding object selected by the *obj* structure. A successful request for an object will result in the field *ret_code* being set to the message identifier OKAY. Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message. Where the function is successful, the field *disp_obj* is a pointer to a structure containing the status for the corresponding selected object. The structure type is dependent on the object and will be the same as used for *obj*. The definitions for these structures are reproduced in the appendix.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example displays the status of all devices (in SHORT format).

```
#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Display the status of all devices (display short)
 * xpstat -dev
 */
main (argc, argv)
    int argc;
    char **argv;
{
    struct disp_mdl *ptr; /* result array */
    union Xp_object *obj; /* select structure */
    short i,size;
    char state[80];

    /* Allocate a select structure and build the selection
     * criteria */
    /* ----- */
    obj = (union Xp_object *) Xp_alloc(DV,SEL_OR_MOD);
    /* Call Xprint to get device status */
    /* ----- */
    size = sta_all_cli(DV,obj,NULL,&ptr);
    /* Display the results */
    /* ----- */
    printf("Name      Waiting jobs Active job   State\n");
    printf("----      -\n");
    for (i=0; i<size; i++) {
        if ((ptr[i].diag.ret_code) == OKAY) {
            switch (((struct sta_dv_ext*)(ptr[i].disp_obj))
                    ->dv_state) {
                case ST_ACTIVE:
                    sprintf(state,"ACTIVE");
                    break;
                case ST_DEVICE_ERROR:
                    sprintf(state,"DEVICE_ERROR (%s)",
                        ((struct sta_dv_ext*)(ptr[i].disp_obj))
                        ->error_msg);
                    break;
                default:
                    sprintf(state,"NOT_ACTIVE");
                    break;
            }
            printf("%-14.14s %4.4d      %-14.14s %s\n",
                ((struct sta_dv_ext*)(ptr[i].disp_obj))>dv_name,
                ((struct sta_dv_ext*)(ptr[i].disp_obj))
                ->dv_waiting_jobs,
                ((struct sta_dv_ext*)(ptr[i].disp_obj))
                ->dv_act_jid,
                state);
        }
    }
}
```

```

}
□□□□□□□□else printf("%s",ptr[i].diag.msg);
□□□□}
□□□□/* Release the allocated memory */
□□□□/*-----*/
□□□□release_cli( (char *) ptr);
□□□□Xp_free(obj);□
}

```

6.20 sta_job_cli - status of job

This function provides the status for one or more selected jobs.

Synopsis

```

#include <xprint.h>
#include <client.h>
short sta_job_cli (job,params,rc_tab)
struct Xp_job *job;
struct sta_job_params *params;
struct disp_mdl **rc_tab;

```

where:

job is a pointer to an *Xp_job* structure, allocated as a select structure by *Xp_alloc()*. This structure is used to select the jobs in the spool queue whose status is to be provided. Refer to the [Section "Object structures"](#) in [Chapter "Preface"](#) for a description of select structures.

params is not used by this version of *Xprint* and should be set to the NULL pointer.

rc_tab is used by *sta_job_cli()* to return the address of the structure *disp_mdl*. This array structure is allocated by *Xprint* and contains one element for each job selected by the *job* structure.



The calling code must free the *rc_tab* structure using *release_cli()*.

Return values

The return value is the number of elements in the *rc_tab* array.

The *rc_tab* structure has each element of its array set to show success or failure of the function for the corresponding job selected by the *job* structure. A successful request for a job will result in the field *ret_code* being set to the message identifier OKAY. Any other value in *ret_code* is an error message identifier; *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

This example displays the status of selected jobs.

```

#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * Display the status of a set of jobs generated by 'application'
 */
□ * xpstat -job
□ */
main (argc, argv)
□□□□int argc;

```

```

char **argv;
{
    struct disp_mdl *ptr;    /* result array */
    struct Xp_job *job;     /* select structure */
    struct sta_job_params *mod; /* parameters -di */
    short i,size;
}

/* Allocate a select structure and build the selection criteria */
/*-----*/
job = (struct Xp_job *) Xp_alloc(JO,SEL_OR_MOD);
job->jo_info.j_comment = "application";
/* Allocate a structure for the parameters of xpstat -job */
/*-----*/
mod = (struct sta_job_params *) Xp_alloc(STA_JOB_PARAMS,NULL);
/* Call Xprint to get job status */
/*-----*/
size = sta_job_cli(job,mod,&ptr);
/* Display the results */
/*-----*/
for (i=0; i<size; i++)
{
    if ((ptr[i].diag.ret_code) == OKAY) {
        printf("LOCAL JOB ID.: %d\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))->lcl_jid);
        printf("Global job id.: %s\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))->glb_jid);
        printf("State: %d\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))->state);
        printf("State information: %s\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))
            ->state_info_msg);
        printf("Destination: %s\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))->dv_name);
        printf("Rank: %d\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))->rank);
        printf("Requested copies: %d\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))
            ->rq_copies);
        printf("Copies printed: %d\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))
            ->pr_copies);
        printf("%% already printed: %d\n",
            ((struct sta_job_ext*)(ptr[i].disp_obj))
            ->pr_percent);
        printf("\n");
    }
    else printf("%s",ptr[i].diag.msg);
}
/* Release the allocated memory */
/*-----*/
release_cli((char *)ptr);
Xp_free(job);
Xp_free(mod);
}

```

6.21 Xp_alloc - structure allocation

This function allocates memory for structures that must be passed as function parameter(s) to Xprint by the application. All such structures must be allocated by this function. The structures allocated will be preset by Xprint as empty or unmodified (depending on their intended use).



All memory allocated by *Xp_alloc()* must be freed using the *Xp_free()* function.

Synopsis

```
#include <xprint.h>
#include <client.h>
void *Xp_alloc(typ,act)
unsigned short typ;
unsigned short act;
where:
```

typ is the identifier for the structure or object type for which memory is to be allocated. Valid types are DV (printer), DG (printer group), FL (filter), FT (font), FM (form), GW (gateway), HO (host), JO (job), JT (job template), LR (loadable resource), PL (PCL), PN (permission list), RR (recovery rules), SP (supervisor), SV (server), and US (user). All the structure types listed in the table on [Table ""](#) and the structure types that are used for the *param* parameter in functions defined in this chapter can be used. The identifiers for the *param* structures are listed within the description for each function in this chapter.

act defines the use of the structure. Xprint will initialize the structure in accordance with the intended use. Valid values for *act* are ADDITION, to indicate an add structure, or SEL_OR_MOD for either a select or modify structure. Refer to the [Section "Object structures" in Chapter "Preface"](#) for a description of the use of add, select and modify structures.



This parameter is ignored when the structure is a *params* structure.

Return values

The *Xp_alloc()* function returns the address of the memory allocated for the structure. The allocated structure is initialized to *empty* for add structures and *not modified* for select and modify structures. *params* structures are always initialized to all binary zeros.

Errors

Xp_alloc() returns a NULL pointer if:

- the requested *typ* is invalid.
- Xprint is unable to allocate memory for the structure.

6.22 Xp_clean - cleaning structure

This function will clean a structure previously allocated by the *Xp_alloc()* function.

Synopsis

```
#include <xprint.h>
#include <client.h>
short Xp_clean(typ,act,ptr)
unsigned short typ;
unsigned short act;
void *ptr;
where:
```

typ is the identifier for the structure type being cleaned and must be the same type as when the structure was originally allocated. Valid types are DV (printer), DG (printer group), FL (filter), FT (font), FM (form), GW

(gateway), HO (host), JO (job), JT (job template), LR (loadable resource), PL (PCL), PN (permission list), RR (recovery rules), SP (supervisor), SV (server), SY (system), and US (user). Refer to the *Xp_alloc()* function for more information on structure types.

act defines the intended use of the structure. Xprint will initialize the structure in accordance with the intended use. Valid values for *act* are ADDITION, to indicate an add structure, or SEL_OR_MOD for either a select or modify structure.

ptr is the pointer to the structure being *cleaned*. This pointer must originally have been supplied by the *Xp_alloc()* function.

Return values

Xp_clean() returns the pointer to the structure if no errors occur.

The allocated structure is initialized to *empty* for add structures and *not modified* for select and modify structures.

Errors

Xp_clean() returns 0 if:

The pointer has the value NULL or the *typ* parameter is invalid.



The memory for the structure still remains allocated.

Example

This example first adds a form to the configuration, then cleans the structure and finally adds a second form.

```
#include <xprint.h>
#include <client.h>
main()
{
    struct Xp_formular *frm; /* add structure */
    struct struct_ret_code *rc_tab; /* result array*/
    short rc_num,i;
    /* allocate add structure */
    /*-----*/
    frm = (struct Xp_formular *) Xp_alloc(FM,ADDITION);
    frm->fm_name = "f1";
    frm->fm_dft_font = "font2";
    /* Call Xprint for adding first form. Display result*/
    /*-----*/
    rc_num = add_all_cli(FM,frm,NULL,&rc_tab);
    for (i=0; i<rc_num; i++)
    {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code : %d\n",
            rc_tab[i].ret_code);
    }
    /* Release allocated memory */
    /*-----*/
    release_cli(rc_tab);
    /* Clean and reuse add structure */
    /*-----*/
    Xp_clean(FM,ADDITION,frm);
    frm->fm_name = "f2";
    frm->fm_comment[1] = "my special form";
    /* Call Xprint for adding a second form. Display result*/
    /*-----*/
}
```

```

rc_num = add_all_cli(FM,frm,NULL,&rc_tab);
for (i=0; i<rc_num; i++)
{
    printf("%s",rc_tab[i].msg);
    printf("corresponding return code :
    %d\n",rc_tab[i].ret_code);
}
/* Release allocated memory */
/*-----*/
release_cli(rc_tab);
Xp_free(frm);
}

```

6.23 Xp_free - freeing memory for structure

This function will free the memory previously allocated by the *Xp_alloc()* to hold a structure.

Synopsis

```

#include <xprint.h>
void Xp_free(ptr)
void *ptr;

```

where:

ptr is a pointer to a structure (memory area) previously allocated by the *Xp_alloc()* function.

Return values

There are no returned values from *Xp_free()*.

Errors

There are no errors returned by *Xp_free()*.



An invalid value for *ptr* will result in unpredictable behavior.

6.24 xpstart_cli - starting Xprint

This function starts Xprint on the local host.

Synopsis

```

#include <xprint.h>
#include <client.h>
short xpstart_cli (params,rc_tab)
struct xpstart_params *params;
struct struct_ret_code **rc_tab;

```

where:

params is a pointer to the structure *xpstart_params*, which is used to select the method of system activation as either cold or warm start.

rc_tab is used by *xpstart_cli()* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains just one element.



The calling code must free the *rc_tab* structure using the *release_cli()* function.

The structure *xpstart_params* has the following definition:

Identifier: XPSTART_PARAMS

```

struct xpstart_params {
    string version;    /* filled by Xp_alloc() */

```

```

□□□□boolean force_version /* xpstart -fv */
□□□□v_start activation; /* xpstart -sa */
□□□□};

```

This structure is used as follows:

Field	Set to	Comments
activation	XP_COLD or XP_WARM	Selects COLD or WARM start respectively

Table 13: Use of the structure xpstart_params

Return values

The return value is the number of elements in the *rc_tab* array. In the case of automatic activation of printers, supervisors and servers the *rc_tab* array may contain several elements.

The *rc_tab* structure shows success or failure. Successful execution of the function will result in *ret_code* being set to the message identifier CL_SETAVAIL and the *msg* field set to the message stating that Xprint was activated or remains active. This array element may be followed by other elements stating the automatic activation of servers (CL_SVACT, "Server <server> set ACTIVE"), supervisors (CL_SPACT, "S"upervisor <supervisor> set ACTIVE), printers (CL_DVACT, "Printer <printer> set active") and gateways (CL_GWACT, "Gateway <gateway> set ACTIVE"). Any other value in *ret_code* is an error message identifier, *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

Xprint is started on the local system; startup type is WARM.

```

#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * start up the spooler system on the local host. The type of
 * startup is WARM.
 * xpstart
 */
main (argc, argv)
□□□□int argc;
□□□□char **argv;
{
□□□□static struct xpstart_params params;
□□□□struct struct_ret_code *rc_tab;
□□□□short i,rc_num;
□□□□params.activation = XP_WARM;
□□□□rc_num = xpstart_cli(&params,&rc_tab);
□□□□for (i=0; i<rc_num; i++) {
□□□□□□□□printf("%s",rc_tab[i].msg);
□□□□□□□□printf("corresponding return code : %d\n",
□□□□□□□□rc_tab[i].ret_code);
□□□□}
□□□□release_cli( (char *) rc_tab);
}

```

6.25 xpstop_cli - stopping Xprint

This function requests that Xprint be deactivated on a set of hosts.

Synopsis

```
#include <xprint.h>
#include <client.h>
short xpstop_cli (params,rc_tab)
struct xpstop_params *params;
struct struct_ret_code **rc_tab;
where:
```

params is a pointer to the structure *xpstop_params*, which is used to select the hosts on which Xprint is to be deactivated.

rc_tab is used by *xpstop()_cli* to return the address of the structure *struct_ret_code*. This array structure is allocated by Xprint and contains one element for each host listed in *params*.



The calling code must free the *rc_tab* structure using the *release_cli()* function.

The structure *xpstop_params* is to be set up as follows:

Identifier: XPSTOP_PARAMS

Field	Set to	Comments
host_list	struct/*string*/sequence	A sequence of strings identifying hosts where Xprint is to be deactivated.

Table 14: Use of the structure *xpstop_params*

Return values

The return value is the number of elements in the *rc_tab* array. This will be equal to the number of hosts listed by the *params* parameter.

The *rc_tab* structure shows success or failure for each host specified by setting the fields *ret_code* and *msg* in the corresponding element (first host in *params* list corresponds to first element of the *rc_tab* array). Successful execution of the function will result in field *ret_code* being set to the message identifier CL_SETUNAVAIL and the *msg* field set to the corresponding message Xprint set or left NOT ACTIVE on <%.128s> (where %.128s is already substituted by the host name). Any other value in *ret_code* is an error message identifier, *msg* will be set to the corresponding error message.

Errors

There are no errors returned other than the message identifier and the error message inserted into the *rc_tab* structure.

Example

Xprint is stopped on the local host.

```
#include "sys/types.h"
#include "stdio.h"
#include <xprint.h>
#include <client.h>
/*
 * stop the spooler on the local host
 * xpstop
 */
main (argc, argv)
□□□□int argc;
□□□□char **argv;
```

```
{
    static struct xpstop_params params;
    struct struct_ret_code *rc_tab;
    short i,rc_num;
    rc_num = xpstop_cli(&params,&rc_tab);
    for (i=0; i<rc_num; i++) {
        printf("%s",rc_tab[i].msg);
        printf("corresponding return code :
        %d\n",rc_tab[i].ret_code);
    }
}
```

7 Printer Access Serialisation

Printer access serialisation is a way to prevent conflict between Xprint and applications which may wish to use the same printer. The `acquire_dev` function allows Xprint to open and 'claim' or 'lock' a physical device port, which will prevent other applications from using the port until Xprint releases it. The `release_dev` function will close the port, and make it available for other applications.

Applications can also use these functions. So if an application claims a port using `acquire_dev`, Xprint cannot use the port until the application has made the port available by using `release_dev`.

This 'first come, first served' procedure will prevent multiple print jobs from being submitted to a printer at the same time.

7.1 `acquire_dev( )` - open device port

This function opens and reserves a physical device port.

Synopsis

```
#include <xprint.h>
int acquire_dev (dev, lpid, timeout)
struct ECM_printer *dev;
int *lpid;
long timeout;
where:
```

dev is a pointer to the *ECM_printer* structure whose contents are used to access the specified device port. This structure *must* be allocated with the *Xp_alloc* function. The following fields have to be set:

```
string dv_add; /* dev -da */
u_short dv_connect_type; /* dev -co XP_SERIAL */
/* XP_PARALLEL */
/* XP_SCSI */
/* XP_LAN */
string dv_tac_add; /* lan address, only with XP_LAN */
u_short dv_tac_port; /* port number, only with XP_LAN */
```

lpid is a pointer to an integer that contains the identifier for the process locking the device, if it was already locked.

timeout is the maximum time allowed for opening the device. The connection with the device will be established without any timeout control if this parameter is omitted.

Return values

The function will return the file descriptor of the device port if it is successful in opening the port. Otherwise, it will return a negative value which identifies the system call that failed. See the following table .

value	system call
-1	open
-2	close
-3	fcntl
-4	gethostbyname
-5	socket
-6	setsockopt
-7	connect
-8	alarm

Table 15: Return values and system calls

The reason that the system call failed can also be found in the `errno` system variable.

If the function returns the value `-3` (`fcntl`) and `errno` is set to `EACCES` (permission denied), then the variable pointed to by `lpid` will contain the identifier of the locking process. Otherwise it will contain `0`. This situation is only possible with printers that are directly attached. That is, *not* with printers connected by a LAN.

Errors

The `acquire_dev( )` function will fail if any system call fails.

Examples

For a complete example, see [Section "release_dev\(\) - release device port"](#).

Application Usage

This function performs a one shot operation, without any recovery in the event of failure.

For printers that are directly attached, (that is *not* LAN printers), Xprint ensures that this function will not last more than two seconds. Xprint cannot give the same guarantee for LAN printers because this would interfere with the networking routines.

This function may be called even if Xprint is `NOT_ACTIVE`.

When a printer is connected directly, (`SERIAL`, `PARALLEL` or `SCSI`), then this function offers a full protection against concurrent accesses by Xprint. On the other hand, for printers connected via a LAN, the situation is not completely secure: if the connection fails (network problem, a reset of the terminal server etc.), then the lock on the port is lost and the function must be called again.

This function can not be used for the IOCS printer ports. This is because the open/lock system calls to the printer port are not made by Xprint, but by the IOCS component itself. However, IOCS also offers similar functions.

If Xprint is active, the `ECM_printer` structure could also be initialized by using Xprint APIs - see the `shw_all_cli( )` Xprint API.

7.2 release_dev() - release device port

This function releases and then closes a physical device port.

Synopsis

```
int release_dev (dev, file_desc, timeout)
struct ECM_printer *dev;
int file_desc;
long timeout;
```

where:

`release_dev( )` releases and closes a physical device port previously acquired by `acquire_dev()`.

`dev` is a pointer to the `ECM_printer` structure . This structure has been filled before calling the `acquire_dev( )` function.

`file_desc` is an integer containing the file descriptor of the device port, returned by the previous call to `acquire_dev( )`.

`timeout` is the maximum allowed time for releasing the device. The connection with the device will be released without any timeout control if this parameter is omitted.

Return Value

The function returns `0` if successful. Otherwise it returns a negative value in case of error. This negative value identifies the failing system call. See the following table. The reason that the system call failed can be found in the `errno` system variable.:

value	system call
-1	open

-2	close
-3	fcntl
-4	gethostbyname
-5	socket
-6	setsockopt
-7	connect
-8	alarm

Table 16: return values and system calls

Errors

The release_dev() function will fail if any system call fails.

Examples

```
#include <xprint.h>
struct ECM_printer *dev;
int file_desc;
int rel_code;
int lpid;
dev = (struct ECM_printer *) Xp_alloc (DV, ADDITION);
dev->dv_connect_type = XP_SERIAL; OR dev->dv_connect_type = XP_LAN;
dev->dv_add = "/dev/term/ser0"; OR dev->dv_add = "d241w010 9100";
dev->dv_tac_add = "d241w010";
dev->dv_tac_port = 9100;
file_desc = acquire_dev (dev, &lpid, 0L);
if (file_desc < 0) {
    printf ("Error: system call %d, errno %d, locking pid %d.\n",
    file_desc, errno, lpid);
}
else {
    write (file_desc, ..., ...);
    read (file_desc, ..., ...);
    .
    .
    rel_code = release_dev (dev, file_desc, 0L);
    if (rel_code < 0) {
        printf ("Error: system call %d, errno %d.\n", rel_code, errno);
    }
    else {
        printf ("Normal end.\n");
    }
}
Xp_free (dev);
```

Application Usage

This function may be called even if Xprint is NOT_ACTIVE.

8 Appendix

This appendix comprises several main sections. The first two sections, "[The client.h file](#)" and "[The xprint.h file](#)", reproduce definitions of macros (preprocessor constants), data structures from the header files *client.h* and *xprint.h* that are used directly by application programs. This is followed by the [Section "xp commands and corresponding structures"](#), which lists the possible options of the individual Xprint commands and states which structures or structure elements must be specified with the corresponding API functions. The last [Section "List of messages"](#) gives a full listing of Xprint V6.0 error, warning and information messages. The messages are also listed in the "Xprint V6.0 - Reference Manual", where additional information on corrective action for error situations is also provided.

8.1 The client.h file

The file */var/spool/Xprint/config/include/client.h* contains the definitions of various preprocessor constants and Xprint data structures.

8.1.1 Definitions

```
/* constants for XPSHOW #2 and XPSTAT #2 */
/*-----*/
#define SHORT_DISPLAY 1
#define LONG_DISPLAY 2
#define HIERARCHY 3   □□□□□□/* only for xpshow -hos */
#define REFERENCED_BY 4□□□□□□/* only for xpshow #2 */
/* constants for Xp_alloc and Xp_clean */
/*-----*/
#define ADDITION 1
#define SEL_OR_MOD 2
/* message codes for the client */
/*-----*/
/**/ Special values for result of show and stat **/
#define OKAY □□□□□□0
#define NOT_OKAY □□-1
```

This is followed by further definitions representing the different results of the API functions (see also [Section "List of messages"](#)).

8.1.2 Result structures

ret_code

```
struct struct_ret_code {      /* #2          */
short ret_code;              /* #2          */
short reserved;              /* #2          */
string msg;                  /* #2          */
};
```

disp_mdl

```
struct disp_mdl {            /* #2          */
struct struct_ret_code diag;
char *disp_obj;              /* ptr to union Xp_shw or */
/* Xp_sta                    */
};
```

8.1.3 Structures for displaying references

shw_ref_by

```
struct shw_ref_by {          /* #2          */
struct struct_ret_code *diag;
```

```

struct sequence ref_list; /* ref_by_obj */
};

ref_by_obj
struct ref_by_obj { /* #2 */
string obj_name;
struct shw_ref_by *sub_ref_list;
u_short obj_type;
u_short level;
};

```

8.1.4 Result structures for Xpstat

sta_dv_ext

This structure is used to return the status information for a selected printer (object type: DV).

```

struct sta_dv_ext { /* #2 */
string dv_name; /* device's name */
xp_v_short dv_state; /* chg -dev -st */
xp_v_short dv_spoolin; /* chg -dev -si */
xp_v_short dv_spoolout; /* chg -dev -so */
xp_v_short dv_when_tbd; /* chg -dev -wn */
string dv_sta_cm; /* chg -dev -ju */
string dv_redirect; /* #2 chg -dev -rd */
string error_msg; /* error msg if state = */
/* DEVICE_ERROR */
string dv_sta_date; /* date of last change */
long dv_waiting_jobs; /* number of waiting jobs */
string dv_cur_form; /* mounted manual form */
string dv_act_jid; /* active job id */
u_short dv_priority; /* chg -dev -pr */
PADDING(2);
};

```

Additional information:

Field	Set to
dv_state	ST_ACTIVE, ST_NOT_ACTIVE or ST_DEVICE_ERROR
dv_spoolin and dv_spoolout	ST_ON or ST_OFF

Table 17: Additional information about sta_dv_ext

sta_dg_ext

This structure is used to return the status information for a selected printer (object type: DV).

```

struct sta_dg_ext { /* #2 */
string dg_name; /* dgroup's name */
string dg_sta_cm; /* chg -dgr -ju */
string dg_sta_date; /* date of last change -dgr */
boolean dg_spoolin; /* chg -dgr -si */
PADDING(2);
};

```

sta_gw_ext

This structure is used to return the status information for a specific gateway (object type: GW).

```

struct sta_gw_ext { /* #2 */

```

```

string  gw_name;      /* gateway's name      */
string  gw_sta_cm;    /* chg gtw -ju      */
string  gw_sta_date;  /* date of last change */
u_long  gw_sta_ndate; /* date of last change num */
xp_v_short gw_state;  /* state -st        */
xp_v_short gw_in;     /* gtw -gi          */
xp_v_short gw_out;    /* gtw -go          */
PADDING(1);
};

```

Additional information:

Field	Set to
gw_state	ST_ACTIVE or ST_NOT_ACTIVE
gw_inandgw_out	ST_ON or ST_OFF

Table 18: Additional information about sta_gw_ext

sta_jo_ext

This structure is used to return the status information for a selected job (object type: JO).

```

struct sta_jo_ext {          /* #2          */
string  glb_jid;             /* global jid(<SV>:<number>)* */
string  glb_title;          /* global jid      */
/* (<SV>:<job_title>)      */
string  dv_name;             /* device the job is on */
string  error_msg;          /* error msg if state = */
/*      DEVICE_ERROR*/
string  state_info_key;     /* #2 key of state info msg */
string  state_info_msg;     /* #2 self explanatory msg */
u_short rank;              /* job rank in the queue */
u_short rq_copies;         /* add -job -cp      */
u_short pr_copies;         /* already printed copies */
u_short pr_percent;        /* percentage printed */
short   lcl_jid;           /* local jid          */
xp_v_short state;          /* chg -job -st      */
PADDING(1);
string  last_pr_doc_name;   /* #3 last printed doc */
u_long  last_pr_page;      /* #3 last printed page */
u_long  last_pr_doc_position; /* #3 position of */
/* last document          */
xp_v_short substate;        /* #3 job substate (IN_DEL) */
PADDING_3(3);              /* #3          */
};

```

Additional information:

Field	Set to
jo_state	ST_ACTIVE, ST_WAIT, ST_SUSPEND, ST_TOP, ST_DATA_FILTERING, ST_FILE_TRANSF, ST_DEVICE_ERROR, ST_TERMINATED, ST_INTERRUPTED, ST_WAITING_FOR_NP or

	ST_SCHEDULED
--	--------------

Table 19: Additional information about sta_jo_ext

The local and global job identifiers are discussed in the chapter "Printing with the command line interface" of the "Xprint V6.0 - User and Administrator Guide".

sta_sv_ext

This structure is used to return the status information for a selected server (object type: SV).

```
struct sta_sv_ext {          /* #2          */
string  sv_name;           /* server's name      */
xp_v_short sv_state;       /* chg -srv -st       */
xp_v_short sv_spoolin;     /* chg -srv -si       */
xp_v_short sv_spoolout;    /* chg -srv -so       */
xp_v_short sv_wnn_tbd;     /* unused             */
u_short  sv_load_level;    /* load level [0-100] */
PADDING(2);
long     sv_jbmax;         /* max. nbr. of jobs  */
long     sv_waiting_jobs; /* number of waiting jobs */
string   sv_sta_date;     /* #4 date of last change */
string   sv_sta_cm;       /* #4 justification */
u_long   sv_sta_ndate;    /* #4 date of last change (num) */
};
```

Additional information:

Field	Set to
sv_state	ST_ACTIVE or ST_NOT_ACTIVE
sv_spoolinandsv_spoolout	ST_ON or ST_OFF

Table 20: Additional information about sta_sv_ext

sta_sp_ext

This structure is used to return the status information for a selected supervisor (object type: SP).

```
struct sta_sp_ext {          /* #2          */
string  sp_name;           /* supervisor's name   */
xp_v_short sp_state;       /* chg -spv -st       */
PADDING(3);
string  sp_sta_date;       /* #4 date of last change */
string  sp_sta_cm          /* #4 justification     */
u_long  sp_sta_ndate       /* date of last change (num) */
};
```

Additional information:

Field	Set to
sp_state	ST_ACTIVE, ST_NOT_ACTIVE or ST_SHUTDOWN

Table 21: Additional information about sta_sp_ext

sta_ho_ext

This structure is used to return the status information for a selected host (object type: HO).

```
struct sta_ho_ext {          /* #2          */
string  ho_name;           /* host's name         */
string  ho_product;        /* #2 spooler product  */
string  ho_spooler_version; /* #2 spooler version  */
};
```

```

xp_v_short ho_state;    /* host's state      */
PADDING(3);
string  ho_sta_date;    /* #4 date of last change */
string  ho_sta_cm       /* #4 justification      */
u_long  ho_sta_ndate    /* date of last change (num)*/
};

```

Additional information:

Field	Set to
ho_state	ST_ACTIVE, ST_NOT_ACTIVE or ST_ACTIVE_MASTER

Table 22: Additional information about sta_ho_ext

sta_us_ext

This structure is used to return the status information for a selected user (object type: US).

```

struct sta_us_ext {      /* #2      */
string  sc_usr_name;    /* user's name      */
long    sc_pages_allowed; /* usr -ap          */
long    sc_cons_pages;  /* already consumed pages */
long    sc_jobs_allowed; /* usr -aj          */
long    sc_cons_jobs;   /* already consumed jobs  */
};

```

8.1.5 Structures for parameter definition

add_all_params

```

struct add_all_params {  /* #2      */
string  version;        /* filled by Xp_alloc() */
};

```

add_job_params

```

struct add_job_params {  /* #2      */
string  version;        /* filled by Xp_alloc() */
string  job_template;
};

```

chg_dev_params

```

struct chg_dev_params {  /* #2      */
string  version;        /* filled by Xp_alloc() */
string  cur_form;       /* xpchange -dev -mf frm_nam*/
string  justification;  /* xpchange -dev -ju      */
u_short new_priority;   /* xpchange -dev -pr      */
xp_v_short change_type; /* (see manual)          */
xp_v_short new_state;   /* xpchange -dev -st      */
xp_v_short new_spoolin; /* xpchange -dev -si      */
xp_v_short new_spoolout; /* xpchange -dev -so      */
xp_v_short when_tbd;    /* xpchange -dev -wn      */
xp_v_short form_manip;  /* mount or unmount      */
/* (see manual)          */
string  dv_redirect;    /* xpchange -dev -rd #2 */
xp_boolean no_redir_jobs /* xpchange -dev -is      */
PADDING_4(2);
};
/* constants for field change_type of struct chg_dev_params */

```

```

/*-----*/
#define CH_STATE 100
#define CH_FORM 101
/* #defines for other values can be found in xprint.h */
/* include file */

chg_dgr_params
struct chg_dgr_params { /* #2 */
string version; /* filled by Xp_alloc() */
string justification; /* xpchange -dgr -ju */
xp_v_short new_spoolin; /* xpchange -dgr -si */
PADDING(3);
};

chg_gtw_params
struct chg_gtw_params { /* #2 */
string version; /* filled by Xp_alloc() */
string justification; /* xpchange -gtw -ju */
xp_v_short new_state; /* xpchange -gtw -st */
xp_v_short new_in; /* xpchange -gtw -gi */
xp_v_short new_out; /* xpchange -gtw -go */
xp_v_short when_tbd; /* xpchange -gtw -wn */
xp_v_short activation; /* xpchange -gtw -sa */
PADDING(3);
};

chg_job_params
struct chg_job_params { /* #2 */
string version; /* filled by Xp_alloc() */
string page_select; /* xpchange -job -ps */
string recover_page; /* #3 xpchange -job -rf */
xp_v_short new_state; /* xpchange -job -st */
PADDING(3);
};

chg_spv_params
struct chg_spv_params { /* #2 */
string version; /* filled by Xp_alloc() */
string justification /* #4 xpchange -spv -ju */
xp_v_short new_state; /* xpchange -spv -st */
xp_v_short when_tbd; /* unused */
PADDING(2);
};

chg_srv_params
struct chg_srv_params { /* #2 */
string version; /* filled by Xp_alloc() */
string justification /* #4 xpchange -spv -ju */
xp_v_short new_state; /* xpchange -srv -st */
xp_v_short new_spoolin; /* xpchange -srv -si */
xp_v_short new_spoolout; /* xpchange -srv -so */
xp_v_short when_tbd; /* unused */
xp_v_short activation; /* xpchange -srv -sa */
PADDING(3);
};

```

chg_usr_params

```
struct chg_usr_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
u_long  cons_pages;        /* xpchange -usr -cp    */
u_long  cons_jobs;         /* xpchange -usr -cj    */
};
```

cpy_all_params

```
struct cpy_all_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
};
```

del_all_params

```
struct del_all_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
};
```

del_job_params

```
struct del_job_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
};
```

mod_all_params

```
struct mod_all_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
struct /*string*/ sequence host_lst; /* xpmmod -usr -cl */
};
```

mod_job_params

```
struct mod_job_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
};
```

shw_all_params

```
struct shw_all_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
u_short disp_mode;         /* xpshow -di          */
PADDING (2);
};
```

shw_job_params

```
struct shw_job_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
u_short disp_mode;         /* xpshow -job -di     */
PADDING (2);
};
```

shw_ref_params

```
struct shw_ref_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
};
```

sta_all_params

```
struct sta_all_params {      /* #2          */
string  version;           /* filled by Xp_alloc() */
};
```

```

u_short  disp_mode;    /* xpsat -di .2      */
xp_v_short dv_poll;    /* xpsat -dev -ep      */
PADDING (1);
};

sta_job_params
struct sta_job_params {    /* #2          */
string  version;          /* filled by Xp_alloc() */
u_short disp_mode;        /* xpsat -job -di .2    */
PADDING (2);
};

xpsat_params
struct xpsat_params {      /* #2          */
string  version;          /* filled by Xp_alloc() */
boolean force_version;     /* xpsat -fv          */
xp_v_short activation;     /* xpsat -sa          */
PADDING (1);
};

xpsat_params
struct xpsat_params {      /* #2          */
string  version;          /* filled by Xp_alloc() */
struct /*string*/ sequence host_list; /* xpsat -ho          */
xp_v_short when_tbd;       /* unused          */
PADDING (3);
};

```

8.1.6 Unions

Xp_sta

```

union Xp_sta {             /* #2          */
struct sta_dv_ext dv;
struct sta_dg_ext dg;
struct sta_gw_ext gw;
struct sta_jo_ext jo;
struct sta_sp_ext sp;
struct sta_sv_ext sv;
struct sta_ho_ext ho;
struct sta_us_ext us;
};

```

Xp_params

```

union Xp_params {          /* #2          */
struct add_all_params add_all;
struct add_job_params add_job;
struct chg_dev_params chg_dev;
struct chg_dgr_params chg_dgr;
struct chg_gtw_params chg_gtw;
struct chg_job_params chg_job;
struct chg_spv_params chg_spv;
struct chg_srv_params chg_srv;
struct xpsat_params xpsat;
struct xpsat_params xpsat;
struct chg_usr_params chg_usr;
struct cpy_all_params cpy_all;
};

```

```

struct del_all_params del_all;
struct del_job_params del_job;
struct mod_all_params mod_all;
struct mod_job_params mod_job;
struct shw_all_params shw_all;
struct shw_job_params shw_job;
struct shw_ref_params shw_ref; /* #2 */
struct sta_all_params sta_all;
struct sta_job_params sta_job;
};

```

8.2 The xprint.h file

The file `/var/spool/Xprint/config/include/xprint.h` contains the constant and type definitions of the different Xprint objects.

8.2.1 General constants and data types

```

/* Boolean values */
#define TRUE 1
#define FALSE 0
#define DEF_PCL "GENERIC"
#define DEF_FONT "GENERIC"
#define DEF_FORM "GENERIC"
#define DEF_TYPE_JO "COMPATIBLE"
#define DEF_TYPE_DV "SIMPLE"
#define SYSTEM_OBJECT_NAME "System"
#ifndef MAXHOSTNAMELEN
#define MAXHOSTNAMELEN 64
#endif
#ifndef MAXPATHLEN
#define MAXPATHLEN 256
#endif
#ifndef MAXJOBLEN
#define MAXJOBLEN 6
#endif
#ifndef MAXLSTRLEN
#define MAXLSTRLEN 12
#endif
#ifndef PADDING
#define PADDING(x) xp_v_short unused[x]
#endif
#ifndef PADDING_3 /* padding for internal version #3 */
#define PADDING_3(x) xp_v_short unused_3[x]
#endif
#ifndef PADDING_4 /* padding for internal version #4 */
#define PADDING_4(x) xp_v_short unused_4[x]
#endif
#define XP_VERSION "1.2"
/* */
/*
* Define XP_CODE_COMPAT to ensure source code compatibility.
* Compile with -DXP_CODE_COMPAT in order to have the old definitions,
* for instance the boolean definition.
*/

```

```

#ifdef XP_CODE_COMPAT
#define boolean xp_boolean
#endif
typedef char* string;
typedef char* version;
typedef char* wcstring;
typedef short xp_enum_t
/* enum value #4 */
typedef long xp_time_t;
/* time value */
typedef char xp_v_short;
/* binary value */
typedef u_short boolean;
typedef long r_long;
/* relative long: when update of DB, */
/* there is accumulation of value */
/* instead of replacement */
typedef struct sequence strsq;
/* sequence of strings */
typedef struct sequence wstrsq;
/*sequence of strings (wild allowed) */
typedef struct sequence vshsq;
/* sequence of bytes (binary) */
typedef struct sequence wvshsq;
/* sequence of bytes (binary with wild) */
typedef struct sequence chrsq;
/* sequence of chars (character) */
/*
*/

```

8.2.2 Various Definitions

```

/*****
/* The following syntax HAS TO BE USED : */
/*      define CC <number in ascending order> ... */
/*      ... <start of comment> struct <ident>*/
/*      ... <end of comment> */
/* Note : no space allowed when number is relative number */
/*      e.g. : define .... NEW+3 and not NEW + 3 */
*****/
#define NOOBJ 0
/* not valid object type */
#define SV 1
/* struct Xp_server server */
#define SP 2
/* struct Xp_superv supervisor */
#define SY 3
/* struct Xp_sys system */
#define DV 4
/* struct ECM_printer device */
#define DG 5
/* struct Xp_dgroup device group */
#define FT 6
/* struct Xp_font font */
#define FM 7

```

```

/* struct Xp_form          form          */
#define PN 8
/* struct Xp_perm          permission    */
#define US 9
/* struct ECM_security      user          */
#define FL 10
/* struct Xp_filter         filter        */
#define DA 11
/* struct Xp_daemon Xprint  daemon        */
#define PL 12
/* struct Xp_pcl            prt control lang*/
#define RR 13
/* struct Xp_rec_rule       recovery rule */
#define HO 14
/* struct Xp_host           host          */
#define JT 15
/* struct Xp_job_temp       job template  */
#define CL 16
/* struct Xp_class          class         */
#define LR 17
/* struct Xp_soft_load      soft loadable */
#define DB_MAXT 18
/* Lowest impossible value for DB type    */
#define DB_EXT 100
/* DB : extension for new objects          */
#define DB_NMAX 110
/* DB : new maximum                       */
#define JO 18
/* struct Xp_job #mercator   job          */
/* start of P/L (entity_call)          */
#define OBJ_MAXT 19
/* Lowest impossible value for obj type    */
/* new objects (Spool 4.2)              */
#define GW DB_EXT
/* struct Xp_gateway         gateway       */
#define OD DB_EXT+1
/* struct Xp_objdesc         tbd          */
#define UM DB_EXT+2
/* struct usr_map            */
#define CO DB_EXT+3
/* struct job_options        */
#define DB_EXT_MAX (DB_EXT+4)
/* Lowest impossible value for extension */
/*
*/
/*****/
/* definitions for object states */
/*****/
/* system, device, supervisor, server and job state */
/* ----- */
#define ST_NOT_ACTIVE      '\000'
#define ST_ACTIVE          '\001'
/* additional state for system, supervisor and server */
/* ----- */

```

```

#define ST_SHUTDOWN      '\002'
/* */
/* additional state for system */
/* ----- */
#define ST_ACTIVE_MASTER  '\003'
/* additional states for device and server */
/* ----- */
#define ST_OFF      ST_NOT_ACTIVE
#define ST_ON       ST_ACTIVE
/* additional states for job */
/* ----- */
#define ST_WAIT      '\002'
#define ST_SUSPEND   '\003'
#define ST_TOP       '\004'
#define ST_DATA_FILTERING  '\005'
/* job waiting filter process end */
#define ST_FILE_TRANSF  '\010'
#define ST_DEVICE_ERROR  '\011'
#define ST_TERMINATED   '\012'
/* job with retention period */
#define ST_INTERRUPTED   '\013'
/* stopped by disabling spoolout */
#define ST_DELETE        '\006'
/* INTERNAL USE ONLY */
#define ST_INTERNAL_SUSP  '\007'
/* " " " */
#define ST_FREE          '\014'
/* " " " */
/* CAUTION: ST_FREE does not mean that this value is free */
#define ST_WAITING_FOR_NP '\015'
/* waiting for next page */
#define ST_SCHEDULED      '\016'
/* job substates #3 */
/* ----- */
#define ST_IN_DELETE      '\001'
#define ST_IN_SUSPEND     '\002'
/* disable spoolout: "when to be done" types */
/* ----- */
#define WT_END            '\001'
#define WT_NOW            '\002'
#define WT_DELETE         '\003'
#define WT_LAST           '\004'
#define WT_DONE           '\005'
#define WT_MAXVALUE       '\005'
/* disable gateway: "when to be done" types */
/* ----- */
#define WT_GW_LAST_TRANS  '\001'
#define WT_GW_NOW         WT_NOW
/* change status actions */
/* ----- */
#define CH_NOT_ACTIVE     '\001'
/* -st for DV, SP, SV and SY */
#define CH_ACTIVE         '\002'
/* -st for SP and SY */

```

```

#define CH_SHUTDOWN      '\003'
/* for SP (warn server) */
#define CH_OFF           CH_NOT_ACTIVE
/* -si -so for DV, DG and SV */
#define CH_ON            CH_ACTIVE
/* -si -so for DV, DG and SV */
#define CH_STOP_AFP      '\010'
/* xpchange -dev -sf */
#define CH_UNTIL_END      '\011'
/* xpchange -dev -np END */
#define CH_NEXT_PAGE     '\012'
/* xpchange -dev -np NEXT */
#define CH_AGAIN          '\013'
/* xpchange -dev -np AGAIN */
#define CH_PRIORITY      '\014'
/* xpchange -dev -pr */
#define CH_MAX_CHANGE     '\015'
/*-----*/
#define CH_WAIT          ST_WAIT
/* xpchange -job -st */
#define CH_TOP           ST_TOP
/* xpchange -job -st */
#define CH_SUSPEND       ST_SUSPEND
/* xpchange -job -st */
#define CH_SCHED_ACTIVE ST_ACTIVE
/* xpchange -job -st */
/* mount form actions */
/*-----*/
#define CH_MOUNT_FORM    '\015' /* xpchange -dev -mf */
#define CH_UNMOUNT_FORM  '\016' /* xpchange -dev -mf- */
#define CH_MAX_MOUNT     '\017' /*-----*/
/* redirection actions #2 */
/*-----*/
#define CH_REDIR          '\020' /* xpchange -dev -rd */
#define CH_STOP_REDIR     '\021' /* xpchange -dev -rd- */
/* user privileges */
/* ----- */
#define ALL_PRIVIL        '\001'
#define NONE_PRIVIL       '\002'
#define TOP_PRIVIL        '\003'
#define ANY_DEVICE        '\004'
/* definitions for notification event types */
/* ----- */
#define EV_ALL            '\001'
#define EV_STARTED        '\002'
#define EV_COMPLETED      '\003'
#define EV_SUSPEND        '\004'
#define EV_ABORTED        '\005'
#define EV_DEVICE_ERROR   '\006'
#define EV_USER           '\007'
#define EV_MODIFIED       '\010'
#define EV_RESUMED        '\011'
#define EV_MOUNTING       '\012'
#define EV_NONE_NO        '\013'

```

```

#define EV_REJECT_FR_REM_DOMAIN '\014'
/* rejected from remote domain */
#define EV_SENT_TO_REM_DOMAIN '\015'
/* sent to remote domain */
#define EV_INTERRUPTED '\016'
#define EV_MAX_EVENTS '\017'
/* definitions for security levels */
/* ----- */
#define UNCLASSIFIED 1
#define SENSITIVE 2
#define CONFIDENTIAL 3
#define SECRET 4
#define TOP_SECRET 5
/* comment indexes -ca -cu */
/* ----- */
#define CM_ADMIN 0
#define CM_USER 1
/* Paper type defs: -dev -ps */
/* ----- */
#define UNDEF_PAPER_FEED -1
#define LISTING_PAPER 1
#define SINGLE_SHEET_AUTO 2
#define SINGLE_SHEET_MANUAL 3
/* sides: -frm -sd */
/* ----- */
#define atXP_SINGLE 1
#define XP_DOUBLE_WIDE 2
#define XP_DOUBLE_NARROW 3
/* paper size indexes */
/* ----- */
#define XP_HEIGHT 0
#define XP_WIDTH 1
/* default paper size values in hundredth of inch */
/* ----- */
#ifdef PRODUCT_PM
/* 11 x 8.5 inches */
#define XP_PAGE_HEIGHT 1100
#define XP_PAGE_WIDTH 850
#else
/* format A4 */
#define XP_PAGE_HEIGHT 1169
#define XP_PAGE_WIDTH 826
#endif
/* connection type -dev -co */
/* ----- */
#define XP_SERIAL 1
#define XP_PARALLEL 2
#define XP_LAN 3
#define XP_SCSI 4
/* Gateway protocol definitions */
/* ----- */
#define DEF_GW_PP "GIP2"
#define DEF_GW_PL "TCP"
/* Gateway arguments keywords */

```

```

/* ----- */
#define AG_ALL      "ALL"
/* Transmission Protocol defs */
/* ----- */
#define XP_XON      1
#define XP_ETX      2
#define XP_RXON     3
/* Possible parity settings */
/* ----- */
#define XP_EVEN_PAR 1
#define XP_ODD_PAR  2
#define XP_NONE_PAR 3
/* Possible stop bits */
/* ----- */
#define XP_STOP_1  1
#define XP_STOP_2  2
/* initialisation information for xpadd -form */
/* ----- */
#define XP_PORTRAIT  '\001'
#define XP_LANDSCAPE '\002'
#define XP_HIGH      '\001'
#define XP_NORMAL     '\002'
#define XP_DRAFT      '\003'
#define XP_ITALICS    '\001'
#define XP_STRAIGHT   '\002'
#define XP_BRIEF_SHEET 1
#define XP_FULL_SHEET  2
#define XP_NONE_SHEET  3
#define XP_FREE_SHEET  4 /* device level only */
/* xpadd -job/frm/dev -no (-nm) */
/* ----- */
#define XP_MAIL      '\001'
#define XP_WRITE     '\002'
#define XP_COMMAND   '\003'
#define XP_FILE       '\004'
/* scheduling policy */
/* ----- */
#define XP_BIG        '\001'
#define XP_SMALL      '\002'
#define XP_PRIORITY   '\003'
#define XP_SECURITY   '\004'
#define XP_MAX_SCHEDPOL '\005'
#define XP_RAW        '\006'
#define XP_NO_RAW     '\007'
/* copies */
/* ----- */
#define XP_SFT_CPY    0
#define XP_HRD_CPY    1
/* Operating system types */
/* ----- */
#define XP_ATT        1
#define XP_BSD        2
/* device keep connection */
/* ----- */

```

```

#define XP_CO_TEMP  '\000'
#define XP_CO_PERM  '\001'
/* filter types */
/* ----- */
#define XP_FAST      1
#define XP_SLOW      2
/* form mounting type: -frm -mm */
/* ----- */
#define XP_AUTOMATIC 1
#define XP_MANUAL    2
/* start name */
/* ----- */
#define SINIX_START_NAME  ".SINIX"
#define T_SPOOL_START_NAME ".T-SPOOL"
/* emulation types */
/* ----- */
#define UNDEF_EMUL  0
#define XPRINT_NATIVE 1
#define SINIX_EMUL  2
#define T_SPOOL_EMUL 3
#define TO_GTW_EMUL 4
#define FROM_GTW_EMUL 5
/* lengths */
/* ----- */
#define XP_HUNDREDTH_OF_INCH      'i'
#define XP_HUNDREDTH_OF_CENTIMETER 'c'
#define XP_LINE                    'L'
#define XP_COLUMN                  'C'
/* form feed processing */
/* ----- */
#define FF_NONE      '\001'
#define FF_BEFORE    '\002'
#define FF_AFTER     '\003'
#define FF_BETWEEN   '\004'
#define FF_ALL       '\005'
/* wrap mode */
/* ----- */
#define WR_CUT        1
#define WR_WRAP       2
#define WR_NONE       3
/* host responsibility */
/* ----- */
#define XP_POT_MASTER 1
#define XP_SLAVE      2
#define XP_PARASITE   3
/* host position */
/* ----- */
#define XP_POS_FIRST  1
#define XP_POS_LAST   2
/* file transfer initiator */
/* ----- */
#define FT_SUPERV      1 /* xpadd -dev -if */
#define FT_CLIENT      2
/* New file transfer type */

```

```

/* ----- */
#define FT_TRANS      1      /* new file transfer type */
/* fast filter synchronisaion */
/* ----- */
#define FL_SYNCHRO     1      /* xpadd -flt -df */
#define FL_ASYNCHRO    2
/* types of gateway */
/* ----- */
#define XP_LOCAL       1
#define XP_PARTNER     2
#define XP_REMOTE      3
#define XP_GW_ALL      4
/* call resource modes of loadable ressource #3 */
/* ----- */
#define XP_LR_ALL      '\001'
#define XP_LR_NONE     '\002'
#define XP_LR_FIRST    '\003'
/* various definitions */
/* ----- */
#define XP_UNDEFINED   -1
/* no value for -usr -pr */
#define RP_INFINITE    -1
/* infinite retention period */
#define OR_INFINITE    0
/* infinite open retry time */
#define XP_NO_LIMIT    -1
/* no limit for -usr -aj, -ap and -nc */
#define DP_JOB_ID      '@'
/* direct printing job identification */
#define PAPER_TYPE     "PAPER_TYPE"
/* special form for T_Spool */
#define STTY_NO_SETTINGS "NO_SETTINGS"
/* special value for -dev -sy */
/* Values meaning "no value at all" (set by the client in */
/* the DB when a value is not specified by the requester and */
/* no default exists) */
/* ----- */
#define UNSP_NUM 0
#define UNSP_US (unsigned short)0x0000
#define UNSP_S (short)0x0000
#define UNSP_UL (unsigned long)0x00000000
#define UNSP_L (long)0x00000000
#define UNSP_C '\000'
#define UNSP_STR ""
/*
*/
/* UNSPECIFIED CHAR SEQUENCE */
/* seq_card : 1 */
/* seq_ptr : "" */
/*
*/
/* UNSPECIFIED STRING SEQUENCE */
/* seq_card : 1 */
/* seq_ptr : &("") */
/*
*/
#define ANY 0xfe      /* PCL wildcard */

```

```

/* Daemon and networking */
/* ----- */
#define UNRECOV_ERROR -1
#define XP_TCP_IP 1
#define XP_ISO 2
#define XP_WARM 'W' /* warm startup */
#define XP_COLD 'C' /* cold startup */
#define XP_RECOVERY 'R' /* recovery startup */
#define XP_NET 1
#define XP_HOST 2
#define XP_DAEMON_REQ 1
/* requests for Xp_daemon */
#define DB_MANAGER_REQ 2
/* requests for DB_manager */
#define SERVER_REQ 3
/* requests for Server */
#define SUPERVISOR_REQ 4
/* requests for Supervisor */
#define GATEWAY_REQ 5
/* requests for Gateway */
#define XP_ACK '\000'
/* positive ack to the request */
#define XP_NACK '\377'
/* negative ack to the request */
/* Gateway */
/* ----- */
#define XP_GW_WARM XP_WARM /* warm startup */
#define XP_GW_COLD XP_COLD /* cold startup */
#define XP_GW_RECOVERY XP_RECOVERY /* recovery startup */
/* signals */
/* ----- */
#define SIG_SF SIGFPE /* stop after first page :
xpchange -dev -sf */
#define SIG_NP SIGSYS /* next page : xpchange -dev -np
AGAIN/NEXT/END */
/* abbreviations, filenames and directories */
/* ----- */
#define RR_PATH "RR/"
#define TMP_PATH "tmp/"
#define PCL_PATH "PCL/"
#define SV_PATH "SV/"
#define SP_PATH "SP/"
#define DOWN_PATH "resources/"
#define MAP_PATH "maps/"
#define FILTER_PATH "filters/"
#define BACKEND_PATH "backends/"
#define GW_PATH "GW/"
#define GTW_PATH "gateways/"
#define BIN_DIR "bin/"
#define LOC_DIR "loc/"
#define CMD_DIR "cmd/"
/*
*/
#endif PRODUCT_SPOOL

```

```

#define CL_SERVICE    "Spool_cl"
#define DA_SERVICE    "Spool_da"
#define DB_SERVICE    "Spool_db"
#define SP_DB_SERVICE "Spool_db"
#define XP_DB_SERVICE "Xprint_db"
#define DB_NAM1       "Spool_db1"
#define DB_NAM2       "Spool_db2"
#define MSG_FILE      "Spool.msg"
#define HLP_FILE      "Spool.hlp"
#define LOG_FILE      "Spool.log"
#define AUD_FILE      "Spool.aud"
#define DBG_FILE      "Spool.dbg"
#define spooler_name   "Spool"
#define spooler_abb1   "SP"
#define spooler_abb2   "Sp"
#define spooler_vers   " 5.0"
#define spooler_rel    " A00"
#endif
#ifdef PRODUCT_PM
#define CL_SERVICE    "PM_cl"
#define DA_SERVICE    "PM_da"
#define DB_SERVICE    "PM_db"
#define SP_DB_SERVICE "Spool_db"
#define XP_DB_SERVICE "Xprint_db"
#define DB_NAM1       "PM_db1"
#define DB_NAM2       "PM_db2"
#define MSG_FILE      "PM.msg"
#define HLP_FILE      "PM.hlp"
#define LOG_FILE      "PM.log"
#define AUD_FILE      "PM.aud"
#define DBG_FILE      "PM.dbg"
#define spooler_name   "DM/Print Manager"
#define template_name  "print_manager"
#define spooler_abb1   "PM"
#define spooler_abb2   "PM"
#define spooler_vers   " 5.0"
#define spooler_rel    " A00"
#endif
/* */
#ifdef PRODUCT_XPRINT
#define CL_SERVICE    "Xprint_cl"
#define DA_SERVICE    "Xprint_da"
#define DB_SERVICE    "Xprint_db"
#define SP_DB_SERVICE "Spool_db"
#define XP_DB_SERVICE "Xprint_db"
#define DB_NAM1       "Xprint_db1"
#define DB_NAM2       "Xprint_db2"
#define MSG_FILE      "Xprint.msg"
#define HLP_FILE      "Xprint.hlp"
#define LOG_FILE      "Xprint.log"
#define AUD_FILE      "Xprint.aud"
#define DBG_FILE      "Xprint.dbg"
#define spooler_name   "Xprint"
#define spooler_abb1   "XP"

```

```

#define spooler_abb2  "Xp"
#define spooler_vers  " 5.0"
#define spooler_rel   " A00"
#endif
/* */
/*
* Internal conventions :
*   Conventions are used by the tool generating abstract data
*   structures for all handled objects.
*
* #n : used as comment identifies the version where the
*      corresponding field was introduced
*      e.g struct ECM_printer {
*          u_long dv_timeout;  #2 : this field was introduced
*                               when INT_VERS was 2
*      };
*
* ^value : specifies how to initialise NEW fields when
*          expanding structures
*          e.g struct ECM_printer {
*              xp_v_short dv_template;  ^FALSE #2
*          };
*
* when comments can not hold on the same line as a field
* definition, they should be stored on the FOLLOWING
* line and NOT before.
* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
* For a new internal version, do not forget to
* update SPOOLER_INT_VERS and XP_LATEST_V.
* Ex: for Xprint V50a, #4 is used and SPOOLER_INT_VERS is 4.
* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*/
/*
* Other conventions :
*
* PADDING_n : macro used for padding related to internal
* version #n
*   Ex: PADDING_3 is used with version #3 (04.2B00)
*
*/
/*
===== History start =====
#define SPOOLER_INT_VERS    1    < Spool 4.2
===== History end =====
*/
#define SPOOLER_INT_VERS  4 /* identified hereafter with #4 */
#define XP_LATEST_V      "4" /* latest version of objects */
/*
* The following sections can be found :
* 1) misc. structures used within other structures
* 2) typical Xprint structures standing for each type of object
*    supported by Xprint except job and job template.
*    The 'xpadd' alphabetic order is used.

```

```

* 3) Ecma structures corresponding to Xprint objects. These are
*   limited to :
*       ECM_printer : Xprint's device object
*       ECM_security: Xprint's user object
* 4) Ecma structures : those structures are referenced by Xprint
*   structures
* 5) Structures related to the job object
*       Xp_jid, Xp_job, Xp_job_temp
* 6) Union of objects handled by the Xprint's DB
*
*/
/*
*/

```

8.2.3 Descriptive structures

sequence

```

struct sequence {
    u_long  seq_card; /* number of elements in the sequence */
    char *  seq_ptr;  /* pointer to an array of objects... */
    /* ...of dimension seq_card */
};

```

scaled_value

```

struct scaled_value {
    u_short value;
    u_short unit;
};

```

pcl_trans

```

struct pcl_trans {
    vshsq  plt_value; /* replacement expression */
    xp_v_short plt_type; /* type of expression */
    PADDING(3);
};

```

pcl_feedback

```

struct pcl_feedback {
    vshsq  plf_afdb;
    /* allowed feedback */
    string plf_prc;
    /* printer return code */
    short  plf_type;
    /* typeof feedback : constant/expression */
    PADDING(2);
    vshsq  plf_pre;
    /* #3 return code as expression */
};
/* identification and meaning of a native sequence */

```

pcl_natsq

```

struct pcl_natsq {
    string pln_string;
    /* native mask */
    vshsq  pln_expression;
    /* terminfo like expression for convers */
};

```

```

short pln_capability;
/* standard capability it stands for */
short pln_keep;
/* this sequence may have to be repeated*/
/* when printing non sequential pages */
};

pcl_language
struct pcl_language {
/* native sequ. understood by printer */
vshsq plp_natsq;
/* starting index for given leading esc */
struct /*pcl_natsq*/ sequence plp_natsq;
/* native sequences */
};

rec_rcd
struct rec_rcd {          /* recovery rule record      */
string rrd_key;          /* recovery rule key          */
u_long rrd_time;         /* allowed error time        */
u_long rrd_retry;        /* time bef.next status      */
string rrd_gencod;       /* sub rule identifier        */
strsq rrd_actions;       /* action list                */
};

job_options
struct job_options {      /* #2 : customised options   */
string pro_identifier;    /* option identifier          */
string pro_value;         /* option value               */
};

usr_map
struct usr_map {          /* #2                          */
string hou_glbld;         /* user identifier within cluster */
string hou_locid;         /* local identifier           */
};

lrs_spec
struct lrs_spec {         /* #2 : loadable resource specifications*/
string lrs_pcl;           /* xpadd -lrs -pc            */
string lrs_file;          /*          -fi              */
string lrs_host;          /*          -ho              */
string lrs_int_id;        /*          -pi              */
boolean lrs_permanent;    /*          -pe              */
xp_v_short lrs_scope;     /*          -sc              */
PADDING(1);
};

Xp_net_info
struct Xp_net_info {      /* network info for active object */
string nt_name;           /* entity name                */
string nt_ent_type;       /* entity type: SP, SV...     */
string nt_host;
u_short nt_port_num;
PADDING(2);
};

```

```

};

Xp_con_protocol
struct Xp_con_protocol {
u_short  cn_protocol;
u_short  cn_speed;
u_short  cn_parity;
u_short  cn_bits;
u_short  cn_stopb;      /* #2 >new<      */
PADDING(2);             /* #2 >new<      */
};

check_point
struct check_point {      /* #3          */
struct /*u_long*/ sequence cp_pagnbr;
/* nb pages per doc      */
u_long  cp_bytes; /* number of printed bytes */
u_long  cp_pages; /* number of printed pages */
u_long  cp_stop;  /* stopped page      */
u_short cp_docnbr; /* active doc number */
u_short cp_cpynbr; /* active soft. copy */
};
/*
*/

Xp_soft_load
struct Xp_soft_load {      /* #2          */
string  lr_name;
strsq   lr_fwref; /* #2 forward reference */
strsq   lr_bckref; /* #2 backward reference */
vshsq   lr_strip; /* stripped data      */
version lr_version; /* loadable resource version */
string  lr_admin; /* lrs administrator */
string  lr_comment[2]; /* -lrs comment -ca -cu */
string  lr_dvperm; /* UNUSED - UNUSED - UNUSED */
string  lr_ad_host; /* -lrs -ah          */
string  lr_type; /* -lrs -tp          */
struct /*lrs_spec*/ sequence lr_lrs_specs; /* UNUSED */
string  lr_pcl; /* #3 -lrs -pc */
string  lr_file; /* #3 -lrs -fi */
string  lr_host; /* #3 UNUSED - UNUSED */
string  lr_int_id; /* #3 -lrs -pi */
string  lr_class; /* #3 -lrs -cn */
strsq   lr_devlst; /* #3 -lrs -dl */
xp_time_t lr_time_stamp; /* #3 INTERNAL USE ONLY */
boolean lr_encapsulate; /* #3 -lrs -ec */
boolean lr_permanent; /* #3 -lrs -pe */
xp_v_short lr_call_resource; /* #3 -lrs -cr */
PADDING_3(3); /* #3          */
};

ECM_notific
struct ECM_notific {
vshsq  no_events; /*ECM* job/frm/dev -no (-ev) */
vshsq  no_triggrs; /*ECM*          */

```

```

string  no_address; /*ECM* job/frm/dev -no (-na) */
string  no_descriptor; /* job/frm/dev -no (-nd) */
string  no_user_name; /* ----- */
string  no_current_host; /* DELIVERY */
string  no_current_dir; /* ENVIRONMENT */
strsq   no_env_var; /* ----- */
u_short no_threshold; /* frm -no (-nt) */
u_short no_frequency; /* frm/dev -no (-nr) */
/* ----- */
u_short no_user_id; /* DELIVERY */
u_short no_group_id; /* ENVIRONMENT */
/* ----- */
xp_v_short no_method; /*ECM* job/frm/dev -no (-nm) */
PADDING(3);
};
/* */

```

8.2.4 Primary Object Structures

The structures in this section are the primary object structures used by Xprint V6.0 (refer to [Chapter "Preface"](#) for more information).

```

/*
 * structure device : see ECM_printer
 */

```

Xp_dgroup

This structure defines the printer group object.

Object type: DG

```

struct Xp_dgroup {
string  dg_name; /* device group name */
strsq   dg_frwrap; /* #2 forward reference */
strsq   dg_bckref; /* #2 backward reference */
vshsq   dg_strip; /* #2 stripped data */
version dg_version; /* device group version */
string  dg_admin; /* administrator */
string  dg_ad_host; /* administration host -ah */
strsq   dg_devlst; /* devices list */
strsq   dg_rem_devlst /* #4 remote dev. list -rl */
string  dg_uperm; /* user permission */
string  dg_comment[2]; /* free comment */
string  dg_sta_cm; /* chg -dgr -ju */
string  dg_sta_date; /* date of change -dgr */
xp_v_short dg_spoolin; /* chg -dgr -si */
PADDING(3);
};

```

Xp_filter

This structure defines the filter object.

Object type: FL

```

struct Xp_filter {
string  fl_name; /* name -flt */
strsq   fl_frwrap; /* #2 forward reference */
strsq   fl_bckref; /* #2 backward reference */
vshsq   fl_strip; /* #2 stripped data */

```

```

version  fl_version; /* filter version */
string   fl_admin;   /* filter administrator */
string   fl_ad_host; /* administration host -ah */
string   fl_comment[2]; /* comment -ca -cu */
string   fl_command; /* activation command -cm */
strsq    fl_host_lst; /* supporting hosts -ho */
strsq    fl_inp_type; /* input types -it */
strsq    fl_out_type; /* output types -ot */
strsq    fl_options; /* options -op */
strsq    fl_sup_smod; /* special modes -sm */
xp_v_short fl_data_flow; /* ^FL_ASYNCHRO flt -df #2 */
xp_v_short fl_unused[3]; /* #2 >new< */
xp_v_short fl_type; /* type: slow or fast -tp */
PADDING(3);
};

```

Xp_font

This structure defines the font object.

Object type: FT

```

struct Xp_font {
string   ft_name;
strsq    ft_frwrap; /* #2 forward reference */
strsq    ft_bckref; /* #2 backward reference */
vshsq    ft_strip; /* #2 stripped data */
version  ft_version; /* font version */
string   ft_admin; /* font administrator */
string   ft_ad_host; /* administration host -ah */
string   ft_typeface; /* typeface -tp */
string   ft_language; /* font -la */
string   ft_comment[2]; /* font -ca -cu */
boolean  ft_type_man; /* typeface mandatory -tm */
xp_v_short ft_quality; /* font -pq */
xp_v_short ft_style; /* font -cs */
xp_v_short ft_vm; /* font -vm */
xp_v_short ft_hm; /* font -hm */
PADDING(2);
};

```

Xp_form

This structure defines the form object

Object type:FM

```

struct Xp_form {
string   fm_name;
strsq    fm_frwrap; /* #2 forward reference */
strsq    fm_bckref; /* #2 backward reference */
vshsq    fm_strip; /* #2 stripped data */
version  fm_version; /* form version */
string   fm_admin; /* form administrator */
string   fm_ad_host; /* administration host -ah */
string   fm_dft_font; /* form -ft */
string   fm_user_perm; /* user permission -up */
struct   ECM_notific fm_notify; /* notification profile */
string   fm_comment[2]; /* form -ca -cu */
};

```

```

struct scaled_value fm_paper_size[2]; /* form -ph -pw */
struct scaled_value fm_page; /* page length value -pl */
struct scaled_value fm_line; /* line length value -ll */
struct scaled_value fm_left; /* left margin value -lm */
struct scaled_value fm_top; /* top margin value -tm */
u_short fm_start; /* form -js */
u_short fm_end; /* form -je */
u_short fm_separator; /* form -sp */
boolean fm_ft_mandatory; /* default font mandatory -mf */
strsq fm_lrs; /* frm -lr #2 */
xp_v_short fm_layup; /* form -ls */
xp_v_short fm_sides; /* form -sd */
xp_v_short fm_mount_method; /* mounting method -mm */
PADDING(1);
};

```

Xp_gateway

This object defines the gateway object.

Object type: GW

```

struct Xp_gateway { /* #2 */
string gw_name;
strsq gw_frwrap; /* #2 forward reference */
strsq gw_bckref; /* #2 backward reference */
vshsq gw_strip; /* #2 stripped data */
version gw_version; /* gateway version */
string gw_admin; /* gateway administrator */
string gw_host; /* gateway */
string gw_uperm; /* gateway -up */
struct Xp_net_info gw_net; /* .nt_host (-ho option) */
string gw_proto; /* gateway -np */
string gw_print_proto; /* gateway -pp */
wcstring gw_orig_name; /* gateway -on */
wcstring gw_orig_host; /* gateway -oh */
strsq gw_allow_list; /* gateway -ag */
strsq gw_bkwd_lks; /* gateway -bl */
strsq gw_options; /* gateway -op */
string gw_sta_cm; /* last change comment */
string gw_sta_date; /* last change date */
u_long gw_sta_ndate; /* numeric last change date */
string gw_cluster; /* cluster name */
struct /*usr_map*/ sequence gw_usr_map; /* cluster : usr map */
vshsq gw_pub_key; /* public key */
vshsq gw_privat_key; /* private key */
u_long gw_distance;
string gw_comment; /* comment -ca */
string gw_unused; /* UNUSED - UNUSED */
string gw_add; /* gateway -ga */
u_short gw_timeout; /* gateway -ti */
boolean gw_aut_act; /* gateway -aa */
boolean gw_check; /* gateway -ck */
xp_v_short gw_type; /* gateway -tp */
xp_v_short gw_state; /* gtw -st */
xp_v_short gw_in; /* gtw -gi */
}

```

```

xp_v_short gw_out;          /* gtw -go          */
xp_v_short gw_loc_state; /* local server status UNUSED */
xp_v_short gw_rem_state; /* remote server status UNUSED */
};

```

Xp_host

This structure defines the host object.

Object type: HO

```

struct Xp_host {
string  ho_name;          /* name -hos          */
strsq   ho_frwrref;      /* #2 forward reference */
strsq   ho_bckref;       /* #2 backward reference */
vshsq   ho_strip;        /* #2 stripped data    */
version ho_version;      /* host version        */
string  ho_product;      /* #2 spooler product  */
string  ho_spooler_version; /* #2 spooler version  */
string  ho_comment[2];    /* comment -ca -cu     */
string  ho_infile;       /* script file with startup cmds*/
string  ho_deffont;      /* default font        */
string  ho_defform;      /* default form        */
string  ho_defdest;      /* default destination  */
string  ho_def_class;    /* default class        */
string  ho_predecessor;   /* predecessor -pd      */
string  ho_supp_host;    /* supporting host -sh   */
string  ho_sta_cm;       /* #4 state justification */
string  ho_sta_date;     /* #4 date of state change */
u_long  ho_sta_ndate;    /* #4 num date of state change */
u_long  ho_nature;       /* #4 nature of the local host */
strsq   ho_sub_root;     /* sub_root list        */
u_long  ho_int_vers;     /* internal version number */
u_short ho_def_max_pri;  /* default maximum priority */
u_short ho_def_job_pri;  /* default job priority    */
u_short ho_def_dev_pri;  /* default device priority  */
short   ho_protocol;     /* network protocol       */
xp_v_short ho_responsibility; /* responsibility -re    */
xp_v_short ho_position;    /* position -po           */
xp_v_short ho_OS;          /* operating system type -os */
padding(1);
};
/*
* struct Xp_soft_load : see above
*/

```

Xp_pcl

This object defines the PCL (Printer Capability List) object.

Object type: PL

```

struct Xp_pcl {
string  pl_name;          /* pcl name          */
strsq   pl_frwrref;      /* #2 forward reference */
strsq   pl_bckref;       /* #2 backward reference */
vshsq   pl_strip;        /* #2 stripped data    */
version pl_version;      /* pcl version        */
version pl_capvers;      /* #2 pcl version(capability) */

```

```

string  pl_admin;      /* user who has created the pcl */
string  pl_host;       /* pcl host -ho      */
string  pl_file;       /* origin file of pcl -fi */
string  pl_filter;     /* permanent fast filter */
string  pl_comment[2]; /* comment -ca -cu      */
string  pl_ldgcc;      /* listing of leading chars*/
string  pl_ldgvp;      /* #3 listing of leading chars */
struct pcl_planguage pl_pnatsq; /* native sequences */
struct pcl_planguage pl_vp;    /* #3 vp native sequences */
struct scaled_value pl_flinl; /* top margin after FF */
struct scaled_value pl_nppl;
/* non printable line area lgth */
struct scaled_value pl_nppi;
/* non printable page area lgth */
vshsq   pl_sfrm1;      /* form lenght sequence */
vshsq   pl_topm;       /* top margin value */
vshsq   pl_leftm;      /* left margin value */
vshsq   pl_settabs;    /* set tabs sequence */
wvshsq  pl_hopper;     /* hopper support */
wvshsq  pl_obin;       /* output bin support */
wvshsq  pl_hrd_copy;   /* harware copies */
wvshsq  pl_dside;
wvshsq  pl_layup;
wvshsq  pl_quality;
wvshsq  pl_style;
wstrsq  pl_typeface;
wvshsq  pl_vm;
wvshsq  pl_hm;
wstrsq  pl_language;
wvshsq  pl_ext1;       /* reserved */
wvshsq  pl_ext2;       /* reserved */
wvshsq  pl_ext3;       /* reserved */
wvshsq  pl_ext4;       /* reserved */
wstrsq  pl_ext5;       /* reserved */
wstrsq  pl_ext6;       /* reserved */
wstrsq  pl_tp_lrs;     /* #3 supported types of lrs */
strsq   pl_options;   /* order for option translation */
/* supported values */
struct /*pcl_trans*/ sequence pl_v_hopper;
struct /*pcl_trans*/ sequence pl_v_obin;
struct /*pcl_trans*/ sequence pl_v_hrd_copy;
struct /*pcl_trans*/ sequence pl_v_dside;
struct /*pcl_trans*/ sequence pl_v_layup;
struct /*pcl_trans*/ sequence pl_v_quality;
struct /*pcl_trans*/ sequence pl_v_style;
struct /*pcl_trans*/ sequence pl_v_typeface;
struct /*pcl_trans*/ sequence pl_v_vm;
struct /*pcl_trans*/ sequence pl_v_hm;
struct /*pcl_trans*/ sequence pl_v_language;
struct /*pcl_trans*/ sequence pl_v_ext1; /* reserved */
struct /*pcl_trans*/ sequence pl_v_ext2; /* reserved */
struct /*pcl_trans*/ sequence pl_v_ext3; /* reserved */
struct /*pcl_trans*/ sequence pl_v_ext4; /* reserved */
struct /*pcl_trans*/ sequence pl_v_ext5; /* reserved */

```

```

struct /*pcl_trans*/ sequence pl_v_ext6; /* reserved */
struct /*pcl_trans*/ sequence pl_v_boirs; /* #3 */
vshsq pl_LF; /* line feed support */
vshsq pl_FF; /* form feed support */
vshsq pl_CR; /* carriage return support */
vshsq pl_BS; /* backspace support */
vshsq pl_HT; /* horizontal tab support */
vshsq pl_cancel; /* cancel support */
vshsq pl_r_boj; /* reset beginning of job */
vshsq pl_r_eoj; /* end of job */
vshsq pl_r_bod; /* beginning of doc */
vshsq pl_r_eod; /* end of doc */
vshsq pl_r_bol; /* beginning of line */
vshsq pl_r_eol; /* end of line */
vshsq pl_r_bop; /* beginning of page */
vshsq pl_r_eop; /* end of page */
vshsq pl_ext7; /* reserved */
vshsq pl_ext8; /* reserved */
vshsq pl_ext9; /* reserved */
string pl_map; /* map table */
string pl_inpro; /* init program */
string pl_expro; /* exit program */
string pl_backend; /* backend to be used */
strsq pl_be_opt; /* backend options */
long pl_bufsiz; /* # bytes buf.before prt */
long pl_speed; /* printing rate */
vshsq pl_stareq; /* printer status */
vshsq pl_handsh; /* handshaking sequence */
struct /*pcl_feedback*/ sequence pl_fdb;
/* allowed feedback */
vshsq pl_eolrs; /* #3 end encapsulate LRS */
vshsq pl_rmlrs; /* #3 rm resources from printer */
vshsq pl_gclrs; /* #3 global resource call */
vshsq pl_lclrs; /* #3 local resource call */
long pl_mem; /* #3 available memory by def. */
vshsq pl_config; /* #3 collect hardware config. */
struct /*pcl_feedback*/ sequence pl_dconf;
/* #3 LRS feedback */
vshsq pl_pboj; /* #3 PjL beginning of job */
vshsq pl_pbod; /* #3 PjL beginning of document */
vshsq pl_pbop; /* #3 PjL beginning of page */
vshsq pl_pbol; /* #3 PjL beginning of line */
vshsq pl_pbob; /* #3 PjL beginning of buffer */
vshsq pl_peoj; /* #3 PjL end of job */
vshsq pl_peod; /* #3 PjL end of document */
vshsq pl_peop; /* #3 PjL end of page */
vshsq pl_peol; /* #3 PjL end of line */
vshsq pl_peob; /* #3 PjL end of buffer */
struct /*pcl_feedback*/ sequence pl_status;
/* #3 PjL feedback */
boolean pl_etx; /* etx allowed */
boolean pl_polling; /* polling allowed */
boolean pl_mix; /* native seq; lower=upper */
short pl_rpi; /* lpi after reset */

```

```

short   pl_rcpi;      /* cpi after reset          */
short   pl_sfrm;      /* form length unit l,i      */
short   pl_topu;      /* top margin unit l,c,i     */
short   pl_leftu;     /* left margin unit l,c,i    */
short   pl_toh;       /* type of handshaking :     */
/*      eop, eod or eoj      */
xp_v_short pl_coml;    /* common part chained esc sequ */
PADDING(1);
};

```

The PCL statements are located in the file referenced by the field *pl_file*. The PCL statements and formats are described in the "Xprint V6.0 Reference Manual".

Xp_perm

This structure defines the permission list object.

Object type: PN

```

struct Xp_perm {
string   pn_name;      /* permission name          */
strsq    pn_frwwref;   /* #2 forward reference     */
strsq    pn_bckref;    /* #2 backward reference    */
vshsq    pn_strip;     /* #2 stripped data         */
version  pn_version;   /* permission version       */
string   pn_admin;     /* administrator name       */
string   pn_ad_host;   /* administration host -ah */
strsq    pn_allow,     /* allowed list             */
pn_deny;   /* denied list             */
string   pn_comment;   /* administrator comment    */
u_short  pn_type;     /* permission type          */
PADDING(2);
};

```

Xp_rec_rule

This structure defines the recovery rules object.

Object type: RR

```

struct Xp_rec_rule {
string   rr_name;      /* recovery rule name       */
strsq    rr_frwwref;   /* #2 forward reference     */
strsq    rr_bckref;    /* #2 backward reference    */
vshsq    rr_strip;     /* #2 stripped data         */
version  rr_version;   /* recovery rule version    */
string   rr_admin;     /* user who has created the rr */
string   rr_host;      /* recovery rule host       */
string   rr_comment;   /* comment -ca             */
string   rr_file;      /* origin file -fi         */
struct /*rec_rcd*/ sequence rr_lst;
/* list of recovery rules */
};

```

The RR statements are located in the file referenced by the field *rr_file*. The RR statements and formats are described in the "Xprint V6.0 Reference Manual".

Xp_superv

This structure defines the supervisor object.

Object type: SP

```

struct Xp_superv {
string   sp_name;      /* supervisor name      */
strsq    sp_frwwref;   /* #2 forward reference */
strsq    sp_bckref;    /* #2 backward reference */
vshsq    sp_strip;     /* #2 stripped data     */
version  sp_version;   /* supervisor version   */
string   sp_admin;     /* admin who added the serv. */
struct Xp_net_info sp_net; /* network info        */
string   sp_servname;  /* server name          */
string   sp_comment;   /* free comment         */
string   sp_sta_cm;    /* #4 state justification */
string   sp_sta_date;  /* #4 date of state change */
u_long   sp_sta_ndate  /* #4 num date of state change */
u_long   sp_int_vers;  /* internal version number */
xp_v_short sp_state;   /* current state        */
PADDING(3);
};

```

Xp_server

This structure defines the server object.

Object type: SV

```

struct Xp_server {
string   sv_name;      /* server name          */
strsq    sv_frwwref;   /* #2 forward reference */
strsq    sv_bckref;    /* #2 backward reference */
vshsq    sv_strip;     /* #2 stripped data     */
version  sv_version;   /* server version       */
string   sv_admin;     /* admin. who adds the server */
struct Xp_net_info sv_net; /* network info        */
string   sv_comment;   /* free comment         */
vshsq    sv_schedpol;  /* scheduling policy     */
string   sv_perm;      /* perm. list of allowed partner*/
string   sv_sta_cm;    /* #4 state justification */
string   sv_sta_date;  /* #4 date of state change */
u_long   sv_sta_ndate  /* #4 num date of state change */
u_long   sv_int_vers;  /* internal version number */
r_long   sv_waiting_jobs; /* number of waiting jobs */
long     sv_jbmax;     /* max. nbr. of jobs     */
u_short  sv_load_level; /* load level            */
xp_v_short sv_wn_tbd;  /* when state will be effective */
xp_v_short sv_state;   /* srv -st              */
xp_v_short sv_spoolin; /* srv -si              */
xp_v_short sv_spoolout; /* srv -so              */
PADDING(2);
};

```

Xp_sys

This structure defines the sytem object.

Object type: SY

```

struct Xp_sys {
string   sy_name;      /* System              */
strsq    sy_frwwref;   /* #2 forward reference */
strsq    sy_bckref;    /* #2 backward reference */

```

```

vshsq  sy_strip;    /* #2 stripped data      */
version sy_version; /* system version          */
string  sy_admin;   /* The Xprint administrator */
string  sy_comment[2]; /* comment -ca -cu        */
string  sy_account_file; /* #3 -af accounting file */
boolean sy_logging; /* #3 -lg for centralized log */
boolean sy_auditing; /* #3 -au for auditing     */
};
/*
 * structure user : see ECM_security
 */

Xp_class
struct Xp_class {
string  cl_name;    /* name                */
version cl_version; /* class version        */
string  cl_admin;   /* class administrator  */
string  cl_ad_host; /* administration host  */
string  cl_alias;   /* alias name           */
string  cl_jt;      /* referenced job template */
string  cl_test_file; /* print test file      */
strsq   cl_dev_lst; /* existing devices      */
strsq   cl_pcl_lst; /* existing pcl's        */
u_long  cl_job_size; /* maximum job size      */
u_long  cl_copies;  /* maximum number of copies */
boolean cl_mapping; /* use mapping table     */
boolean cl_preact;  /* use preaction program */
boolean cl_postact; /* use postaction prog   */
boolean cl_filter;  /* use filter            */
};

```

Xp_soft_char_set

```

struct Xp_soft_char_set { /* soft loadable character set */
string  sl_name;    /* name -slc              */
version sl_version; /* s_l_c_s version        */
string  sl_admin;   /* s_l_c_s administrator  */
string  sl_ad_host; /* administration host -ah */
strsq   sl_loc_lst; /* location list -ll       */
};

```

Xp_objdesc

```

struct Xp_objdesc {
vshsq  ob_def;
};
/*****
/* XPRINT INTERNAL OBJECTS */
*****/
/*
*/

```

8.2.5 ECMA-specific structures

ECM_printer

This structure defines the printer object.

Object type: DV

```

struct ECM_printer {
string   dv_name;           /*ECM*/
strsq    dv_frwrap;        /* #2 forward reference */
strsq    dv_bckref;        /* #2 backward reference */
vshsq    dv_strip;         /* #2 stripped data */
version  dv_version;       /* version */
string   dv_location;      /*ECM*/
string   dv_host;          /*ECM*/ /* host name */
string   dv_model;         /*ECM*/
string   dv_type;          /*ECM*/
string   dv_admin;
string   dv_pcl;           /* dev -pc */
string   dv_filter;        /* dev -fl */
string   dv_usr_perm;      /* dev -up */
string   dv_dft_form;      /* dev -fm */
string   dv_j_title;       /*ECM*/
string   dv_rec_rule;      /* dev -rr */
string   dv_comment[2];    /* dev -ca -cu */
string   dv_add;           /* dev -da */
string   dv_tac_add;       /* internal use; tacan address */
/* significant only when */
/* dv_connect_type = XP_LAN */
string   dv_supervisor;    /* dev -su */
struct Xp_con_protocol dv_proto; /* dev -po -sp -pb -sb */
struct ECM_notific dv_notify; /* dev -no */
strsq    dv_all_type;      /* dev -ct */
r_long   dv_waiting_jobs;
string   dv_stty;          /* dev -sy */
string   dv_cur_form;      /* current form */
string   dv_fo_perm;       /* dev -fp */
string   dv_sta_cm;        /* chg -dev -ju */
string   dv_sta_date;      /* date of chg -dev -st */
/* u_long   dv_sta_ndate;    sta_date in num form */
boolean  dv_first_page;    /* chg -dev -sf */
u_short  dv_start_sheet;   /* dev -ms */
u_short  dv_tac_port;      /* internal use; tacan port nbr*/
/* significant only when */
/* dv_connect_type = XP_LAN */
boolean  dv_aut_act;       /* dev -aa */
u_short  dv_security;      /*ECM*/ /* dev -sl */
u_short  dv_speed;         /*ECM*/
u_short  dv_resolution;    /*ECM*/
u_short  dv_priority;      /* dev -pr */
u_short  dv_paper_feed;    /* dev -ps */
u_short  dv_connect_type;  /* dev -co */
boolean  dv_enable_poll;   /* dev -ep */
xp_v_short dv_state;       /* dev -st */
xp_v_short dv_spoolin;     /* dev -si */
xp_v_short dv_spoolout;    /* dev -so */
xp_v_short dv_keep_connection; /* dev -kc */
u_short  dv_open_retry_time; /* dev -or */
string   dv_redirect;      /* xpchange -dev -rd #2 */
strsq    dv_rdhistry;      /* internal history #2 */
strsq    dv_lrs;           /* dev -lr #2 */

```

```

u_long   dv_timeout;      /*      dev -ti #2 */
boolean  dv_template;     /* ^FALSE   dev -te #2 */
boolean  dv_resource_mgr; /* ^FALSE   dev -me #2 */
xp_v_short dv_ft_initiator; /* ^FT_SUPERV dev -if #2 */
PADDING (3);              /* #2 >new<      */
};

```

ECM_security

This structure defines the Xprint V6.0 user object.

Object type: US

```

struct ECM_security {
string   sc_usr_name;      /*ECM*/
strsq    sc_frwrap;       /* #2 forward reference */
strsq    sc_bckref;       /* #2 backward reference */
vshsq    sc_strip;        /* #2 stripped data     */
version  sc_version;      /* version              */
string   sc_authentic;    /*ECM*/
string   sc_account;      /*ECM*/
strsq    sc_host;
string   sc_home;
vshsq    sc_privileges;   /* user -pv            */
string   sc_comment;
r_long   sc_cons_pages;
r_long   sc_cons_jobs;
long     sc_max_priority; /* user -pr            */
long     sc_max_copies;  /* user -nc            */
long     sc_jobs_allowed; /* user -aj            */
long     sc_pages_allowed; /* user -ap            */
boolean  sc_usr_account; /* user -ua            */
u_short  sc_uid;
u_short  sc_gid;
u_short  sc_securit_lvl; /* user -sl            */
};
/*
*/

```

ECM_j_info

```

struct ECM_j_info {
string   j_title;         /*ECM*/ /* job -jt      */
string   j_originator;    /*ECM*/ /* -scl(-jo)   */
string   j_state_key;     /* READ ONLY - DO NOT INITIALIZE*/
string   j_state_msg;     /* READ ONLY - DO NOT INITIALIZE*/
string   j_language;      /* user language      */
string   j_comment;       /*ECM*/ /* job comment -jc */
u_short  j_client_id;     /* client identifier   */
u_short  j_start_sheet;   /*ECM                */
u_short  j_separ_sheet;   /*ECM                */
u_short  j_end_sheet;     /*ECM                */
xp_v_short j_nbr[6];      /* server job id      */
xp_v_short j_state;       /* job status -st     */
PADDING(1);
};

```

ECM_j_res

```

struct ECM_j_res {
string  dlivry_method; /*ECM*/
string  dlivry_name;   /*ECM*/
string  dlivry_add;    /*ECM*/
};

ECM_j_schedul
struct ECM_j_schedul {
strsq   sch_competitors; /*ECM*/
string  sch_destin;      /* job -de */
string  sch_j_state_ins; /*READ ONLY - DO NOT INITIALIZE*/
string  sch_time_after; /*ECM*/ /* job -at */
string  sch_magic; /* READ ONLY - DO NOT INITIALIZE*/
string  sch_time_retention; /*ECM*/ /* job -rt */
string  sch_sub_time; /* job submission time */
long    sch_retention; /* job -rp */
u_long  sch_t_after; /*ECM*/
u_long  sch_t_retention; /*ECM*/
u_long  sch_s_time;
u_long  sch_sv_dtime;
u_short sch_s_wait; /*ECM*/
u_short sch_e_wait; /*ECM*/
u_short sch_hold; /*ECM*/
u_short sch_priority; /*ECM*/ /* job -pr */
u_short sch_securit_lvl; /* job -sl */
boolean sch_wait_activation; /* job -wa */
};

ECM_font
struct ECM_font {
string  eft_name; /*ECM*/
string  eft_substituter; /*ECM*/
};

ECM_media
struct ECM_media {
string  eme_name; /*ECM*/
string  eme_substituter; /*ECM*/
};

ECM_ink
struct ECM_ink {
string  ein_name; /*ECM*/
string  ein_substituter; /*ECM*/
};

ECM_select
struct ECM_select {
string  sel_from; /* job -fr */
string  sel_to; /* job -to */
string  sel_page_select; /*ECM*/ /* job -ps */
string  sel_recover_page; /*#3 copy of xpchange -job -rf*/
struct check_point sel_check;
/* #3 return by SP - saved by SV*/
};

```

ECM_production

```

struct ECM_production {
    struct ECM_font prd_E_font; /*ECM*/
    struct ECM_media prd_media; /*ECM*/
    struct ECM_ink prd_ink; /*ECM*/
    struct ECM_select prd_select; /*ECM*/ /* ECMA +! */
    struct Xp_form prd_form; /* job -fm */
    struct Xp_font prd_font; /* job -fm(-ft) */
    string prd_finishing; /*ECM*/
    string prd_out; /*ECM*/
    vshsq prd_ff_mode; /* job -ff */
    u_short prd_sides; /*ECM*/
    u_short prd_copies[2]; /*ECMA +! */ /* job -fc -fp */
    u_short prd_wrap_mode; /* job -wm */
    u_short prd_tab_width; /* job -tw */
    u_short prd_hopper; /* job -hn */
    u_short prd_obin; /* job -ob */
    boolean prd_direct_prt; /* job -dp */
    boolean prd_first_page; /* job -sf */
    xp_v_short prd_layout; /*ECM*/ /* form -ls */
    xp_v_short prd_quality; /*ECM*/ /* font -pq */
    struct /* job_options */ sequence prd_options; /* job -op #2 */
    strsq prd_filters; /* job -fl #2 */
    boolean prd_novalidation; /* job -nv #2 */
    PADDING(2); /* #2 */
    strsq prd_lrs; /* #3 job -lr */
    struct /*Xp_soft_load*/ sequence prd_private_lrs; /* #3 */
    struct Xp_soft_load prd_overwrite_lrs; /* #3 job -cr */
};

```

ECM_profile

```

struct ECM_profile {
    strsq prf_files; /* job -ld */
    strsq prf_formats; /*ECM*/
    strsq prf_char_sets; /*ECM*/
    strsq prf_char_maps; /*ECM*/
    strsq prf_fonts; /*ECM*/
    strsq prf_char_rep; /*ECM*/
    strsq prf_medias; /*ECM*/
    strsq prf_inks; /*ECM*/
    strsq prf_hints; /*ECM*/
    string prf_map_table; /* job -mt */
    u_short prf_size[2]; /*ECM*/
    u_short prf_orientation; /*ECM*/
    u_short prf_page_order; /*ECM*/
    u_short prf_page_independent; /*ECM*/
    PADDING(2);
};

```

ECM_doc_uid

```

struct ECM_doc_uid {
    string did_title; /*ECM*/
    string did_author; /*ECM*/
    string did_comment; /*ECM*/
};

```

```

string  did_rev_date;    /*ECM*/
};

ECM_document
struct ECM_document {
struct ECM_profile doc_profile; /*ECM*/
struct ECM_doc_uid doc_uid;    /*ECM*/
string  doc_format;          /*ECM*/
string  doc_host;            /*ECM*/
strsq   doc_doc_ref;         /*ECM*/ /* job -dr (ECMA+) */
string  doc_description;     /*ECM*/
strsq   doc_fil_ref;         /* full path          */
strsq   doc_modes;           /* job -sm          */
string  doc_content_type;     /* job -ct          */
string  doc_result_type;      /* resulting type    */
string  doc_stotal_size;      /* #4 64 bits job size */
u_long  doc_total_size;       /* # bytes to print   */
boolean doc_raw_mode;         /* job -rw          */
boolean doc_no_filter;        /* job -nf          */
boolean doc_remove;          /*ECM*/ /* job -rm          */
boolean doc_copy;            /*ECM*/ /* job -cp          */
u_short doc_transfer;         /*ECM*/
u_short doc_trext;           /* extension for future */
};

```

ECM_processing

```

struct ECM_processing {
string  prc_service;         /*ECM*/
strsq   prc_fit_applied;     /* applied filter cascade */
};
/*
* ECM_notific ==> see in descriptive structures
*/

```

ECM_a_crit

```

struct ECM_a_crit {
vshsq   a_count;            /*ECM*/
u_short a_thresh;           /*ECM*/
PADDING(2);
};

```

ECM_w_crit

```

struct ECM_w_crit {
vshsq   w_count;            /*ECM*/
u_short w_thresh;           /*ECM*/
PADDING(2);
};

```

ECM_j_event

```

struct ECM_j_event {
struct ECM_notific ev_notify; /*ECM*/ /* job -no          */
struct ECM_a_crit  ev_abort;  /*ECM*/
struct ECM_w_crit  ev_warn;   /*ECM*/
};
/*

```

*/

8.2.6 Job Structures

Xp_jid

```
struct Xp_jid {
vshsq   jid_strip; /* #2 stripped data */
string  jid_gateway; /* gateway identifier #2 */
string  jid_server; /* server id (@<device> if -dp) */
string  jid_jid; /* job id [1..30000] (0 if -dp) */
string  jid_title; /* job title */
short   jid_local_jid; /* local job id */
PADDING(2);
};
```

Xp_job

This structure defines the Xprint V6.0 job object.

Object type: JO

```
struct Xp_job {
vshsq           jo_strip; /* #2 stripped data */
version         jo_version;
struct ECM_j_info   jo_info;
struct ECM_j_res    jo_result;
struct ECM_j_schedul jo_schedul;
struct ECM_printer  jo_dev;
struct ECM_production jo_product;
struct ECM_document jo_document;
struct ECM_processing jo_process;
struct ECM_security  jo_security;
struct ECM_j_event   jo_event;
struct Xp_jid        jo_id;
struct Xp_class      jo_T_class;
strsq jo_echo;
};
```

Xp_job_temp

```
struct Xp_job_temp {
string  jt_name; /* name -jtp */
strsq   jt_frwrref; /* #2 forward reference */
strsq   jt_bckref; /* #2 backward reference */
vshsq   jt_strip; /* #2 stripped data */
version jt_version; /* job template version */
string  jt_admin; /* job template administrator */
string  jt_ad_host; /* administration host -ah */
struct Xp_job jt_job; /* job parameters */
};
/*
*/
```

8.2.7 Possible Xprint objects

```
union Xp_object {
struct ECM_printer   dv;
struct Xp_dgroup     dg;
struct Xp_filter      fl;
struct Xp_font        ft;
```

```

struct Xp_form          fm;
struct Xp_gateway       gw;  /* #2      */
struct Xp_host          ho;
struct Xp_soft_load     lr;  /* #2      */
struct Xp_pcl           pl;
struct Xp_perm          pn;
struct Xp_rec_rule      rr;
struct Xp_superv        sp;
struct Xp_server        sv;
struct Xp_sys           sy;
struct ECM_security     us;
struct Xp_class         cl;
struct Xp_soft_char_set sl;
struct Xp_job           jo;
struct Xp_job_temp      jt;
};
#ifdef __cplusplus
};
#endif
#endif /* XPRINT_H */

```

8.3 xp commands and corresponding structures

This section describes for all Xprint commands which structures or structure elements must be specified for the available command options with the API functions corresponding to these commands.

8.3.1 The xpadd command

xpadd -dev/add_all_cli() (ECM_printer.dv_name)

Option	Structure (element)	Permissible values
-aa	ECM_printer.dv_aut_act	
-ca	ECM_printer.dv_comment [CM_ADMIN]	
-co	ECM_printer.dv_connect_type	XP_LAN, XP_SCSI, XP_PARALLEL, XP_SERIAL
-ct	ECM_printer.dv_all_type	
-cu	ECM_printer.dv_comment [CM_USER]	
-da	ECM_printer.dv_add	
-ep	ECM_printer.dv_enable_poll	
-ev	ECM_printer.dv_notify.no_events	EV_DEVICE_ERROR, EV_MOUNTING, EV_NONE_NO, EV_ALL
-fl	ECM_printer.dv_filter	
-fm	ECM_printer.dv_dft_form	
-fp	ECM_printer.dv_fo_perm	
-ho	ECM_printer.dv_host	
-kc	ECM_printer.dv_keep_connection	XP_CO_PERM, XP_CO_TEMP

-lr	ECM_printer.dv_lrs	
-me	ECM_printer.dv_resource_manager	
-ms	ECM_printer.dv_start_sheet	XP_FULL_SHEET, XP_NONE_SHEET, XP_FREE_SHEET, XP_BRIEF_SHEET
-na	ECM_printer.dv_notify.no_address	
-nd	ECM_printer.dv_notify.no_descriptor	
-nm	ECM_printer.dv_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-nr	ECM_printer.dv_notify.no_frequency	
-or	ECM_printer.dv_open_retry_time	OR_INFINITE
-pb	ECM_printer.dv_proto.cn_parity	XP_NONE_PAR, XP_EVEN_PAR, XP_ODD_PAR
-pc	ECM_printer.dv_pcl	
-po	ECM_printer.dv_proto.cn_protocol	XP_ETX, XP_RXON, XP_XON
-pr	ECM_printer.dv_priority	
-ps	ECM_printer.dv_paper_feed	SINGLE_SHEET_MANU AL, LISTING_PAPER, SINGLE_SHEET_AUTO
-rr	ECM_printer.dv_rec_rule	
-sb	ECM_printer.dv_proto.cn_bits	8, 7
-sl	ECM_printer.dv_security	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-sp	ECM_printer.dv_proto.cn_speed	38400, 19200, 9600, 4800, 2400, 1200
-su	ECM_printer.dv_supervisor	
-sy	ECM_printer.dv_stty	
-up	ECM_printer.dv_usr_perm	

Table 23: xpadd -dev/add_all_cli() (ECM_printer.dv_name)

xpadd -dgr/add_all_cli() (Xp_dgroup.dg_name)

Option	Structure (element)	Permissible values
-ah	Xp_dgroup.dg_ad_host	
-ca	Xp_dgroup.dg_comment [CM_ADMIN]	
-cu	Xp_dgroup.dg_comment [CM_USER]	
-dl	Xp_dgroup.dg_devlst	

-rl	Xp_dgroup.dg_rem_devlst	
-up	Xp_dgroup.dg_uperm	

Table 24: xpadd -dgr/add_all_cli() (Xp_dgroup.dg_name)

xpadd -flt (Xp_filter.fl_name)

Option	Structure (element)	Permissible values
-ah	Xp_filter.fl_ad_host	
-ca	Xp_filter.fl_comment [CM_ADMIN]	
-cm	Xp_filter.fl_command	
-cu	Xp_filter.fl_comment [CM_USER]	
-ho	Xp_filter.fl_host_lst	
-it	Xp_filter.fl_inp_type	
-op	Xp_filter.fl_options	
-ot	Xp_filter.fl_out_type	
-tp	Xp_filter.fl_type	XP_FAST, XP_SLOW

Table 25: xpadd -flt (Xp_filter.fl_name)

xpadd -fnt/add_all_cli() (Xp_font.ft_name)

Option	Structure (element)	Permissible values
-ah	Xp_font.ft_ad_host	
-ca	Xp_font.ft_comment [CM_ADMIN]	
-cs	Xp_font.ft_style	XP_ITALICS, XP_STRAIGHT
-cu	Xp_font.ft_comment [CM_USER]	
-hm	Xp_font.ft_hm	
-la	Xp_font.ft_language	
-pq	Xp_font.ft_quality	XP_HIGH, XP_NORMAL, XP_DRAFT
-tm	Xp_font.ft_type_man	
-tp	Xp_font.ft_typeface	
-vm	Xp_font.ft_vm	

Table 26: xpadd -fnt/add_all_cli() (Xp_font.ft_name)

xpadd -frm/add_all_cli (Xp_form.fm_name)

Option	Structure (element)	Permissible values
-ah	Xp_form.fm_ad_host	
-ca	Xp_form.fm_comment [CM_ADMIN]	
-cu	Xp_form.fm_comment [CM_USER]	
-ft	Xp_form.fm_dft_font	
-je	Xp_form.fm_end	XP_NONE_SHEET,

		XP_FULL_SHEET, XP_BRIEF_SHEET
-js	Xp_form.fm_start	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-ll	Xp_form.fm_line	
-lm	Xp_form.fm_left	
-lr	Xp_form.fm_lrs	
-ls	Xp_form.fm_layout	XP_PORTRAIT, XP_LANDSCAPE
-mf	Xp_form.fm_ft_mandatory	
-mm	Xp_form.fm_mount_method	XP_MANUAL, XP_AUTOMATIC
-nt	Xp_form.fm_notify.no_treshold	
-ph	Xp_form.fm_paper_size [XP_HEIGHT]	
-pl	Xp_form.fm_page	
-pw	Xp_form.fm_paper_size [XP_WIDTH]	
-sd	Xp_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sp	Xp_form.fm_separator	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-tm	Xp_form.fm_top	
-up	Xp_form.fm_user_perm	

Table 27: xpadd -frm/add_all_cli (Xp_form.fm_name)

xpadd -gtw/add_all_cli() (Xp_gateway.gw_name)

Option	Structure (element)	Permissible values
-aa	Xp_gateway.gw_aut_act	
-ag	Xp_gateway.gw_allow_list	
-bl	Xp_gateway.gw_bkwd_lks	
-ca	Xp_gateway.gw_comment	
-ga	Xp_gateway.gw_add	
-ho	Xp_gateway.gw_net.nt_host	
-np	Xp_gateway.gw_proto	
-oh	Xp_gateway.gw_orig_host	
-on	Xp_gateway.gw_orig_name	
-op	Xp_gateway.gw_options	
-pp	Xp_gateway.gw_print_proto	
-tp	Xp_gateway.gw_type	XP_PARTNER, XP_LOCAL

-up	Xp_gateway.gw_uperm	
-----	---------------------	--

Table 28: xpadd -gtw/add_all_cli() (Xp_gateway.gw_name)

Option	Structure (element)	Permissible values
-ah	Xp_soft_load.lr_ad_host	
-ca	Xp_soft_load.lr_comment[CM_ADMIN]	
-cn	Xp_soft_load.lr_class	
-cr	Xp_soft_load.lr_call_resource	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cu	Xp_soft_load.lr_comment[CM_USER]	
-dl	Xp_soft_load.lr_devlst	
-ec	Xp_soft_load.lr_encapsulate	
-fi	Xp_soft_load.lr_file	
-pc	Xp_soft_load.lr_pcl	
-pe	Xp_soft_load.lr_permanent	
-pi	Xp_soft_load.lr_int_id	
-tp	Xp_soft_load.lr_type	

Table 29: padd -lrs/add_all_cli() (Xp_soft_load.lr_name)

xpadd -hos/add_all_cli() (Xp_host.ho_name)

Option	Structure (element)	Permissible values
-ca	Xp_host.ho_comment [CM_ADMIN]	
-cu	Xp_host.ho_comment [CM_USER]	
-de	Xp_host.ho_defdest	
-dp	Xp_host.ho_def_dev_pri	
-fi	Xp_host.ho_infile	
-fm	Xp_host.ho_defform	
-ft	Xp_host.ho_deffont	
-mp	Xp_host.ho_def_max_pri	
-np	Xp_host.ho_protocol	XP_ISO, XP_TCP_IP
-os	Xp_host.ho_OS	XP_BSD, XP_ATT
-pd	Xp_host.ho_predecessor	
-po	Xp_host.ho_position	XP_POS_LAST, XP_POS_FIRST
-pr	Xp_host.ho_def_job_pri	
-re	Xp_host.ho_responsibility	XP_PARASITE, XP_SLAVE, XP_POT_MASTER
-sh	Xp_host.ho_supp_host	
-sr	Xp_host.ho_sub_root	

Table 30: xpadd -hos/add_all_cli() (Xp_host.ho_name)

xpadd -job/add_all_cli()

Option	Structure (element)	Permissible values
-at	Xp_job.job_schedul.sch_time_after	
-cp	Xp_job.job_document.doc_copy	
-cr	Xp_job.job_product.prd_over-write_l rs.lr_call_resource	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cs	Xp_job.job_product.prd_font.ft_style	XP_ITALICS, XP_STRAIGHT
-ct	Xp_job.job_document.doc_content_t ype	
-de	Xp_job.job_schedul.sch_destin	
-dp	Xp_job.job_product.prd_direct_prt	
-dr	Xp_job.job_document.doc_doc_ref	
-ev	Xp_job.job_event.ev_notify.no_even ts	EV_SENT_TO_REM_DOMAI N, EV_REJECT_FR_REM_D OMAIN, EV_RESUMED, EV_MODIFIED, EV_USER, EV_DEVICE_ERROR, EV_ABORTED, EV_SUSPEND, EV_COMPLETED, EV_STARTED, EV_NONE_NO, EV_ALL
-fc	Xp_job.job_product.prd_copies [XP_SFT_CPY]	
-ff	Xp_job.job_product.prd_ff_mode	FF_ALL, FF_BETWEEN, FF_AFTER, FF_BEFORE, FF_NONE
-fm	Xp_job.job_product.prd_form.fm_na me	
-fp	Xp_job.job_product.prd_copies [XP_HRD_CPY]	
-fr	Xp_job.job_product.prd_select.sel_fr om	
-ft	Xp_job.job_product.prd_font.ft_nam e	
-hm	Xp_job.job_product.prd_font.ft_hm	
-hn	Xp_job.job_product.prd_hopper	
-jc	Xp_job.job_info.j_comment	
-je	Xp_job.job_product.prd_form.fm_en d	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-js	Xp_job.job_product.prd_form.fm_sta	XP_NONE_SHEET,

	rt	XP_FULL_SHEET, XP_BRIEF_SHEET
-jt	Xp_job.jo_info.j_title	
-la	Xp_job.jo_product.prd_font.ft_language	
-ld	Xp_job.jo_document.doc_profile.prf_files	
-ll	Xp_job.jo_product.prd_form.fm_line	
-lm	Xp_job.jo_product.prd_form.fm_left	
-lr	Xp_job.jo_product.prd_lrs	
-ls	Xp_job.jo_product.prd_form.fm_layout	XP_PORTRAIT, XP_LANDSCAPE
-mt	Xp_job.document.doc_profile.prf_map_table	
-na	Xp_job.jo_event.ev_notify.no_address	
-nd	Xp_job.jo_event.ev_notify.no_descriptor	
-nf	Xp_job.jo_document.doc_no_filter	
-nm	Xp_job.jo_event.ev_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-nv	Xp_job.jo_product.prd_novalidation	
-ob	Xp_job.jo_product.prd_prd_obin	
-pc	Xp_job.jo_dev.dv_pcl	
-pl	Xp_job.jo_product.prd_form.fm_page	
-pq	Xp_job.jo_product.prd_font.ft_quality	XP_HIGH, XP_NORMAL, XP_DRAFT
-pr	Xp_job.jo_schedul.sch_priority	
-ps	Xp_job.jo_product.prd_select.sel_page_select	
-rm	Xp_job.jo_document.doc_remove	
-rp	Xp_job.jo_schedul.sch_retention	RP_INFINITE
-rt	Xp_job.jo_schedul.sch_time_retention	
-rw	Xp_job.jo_document.doc_raw_mode	
-sd	Xp_job.jo_product.prd_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sf	Xp_job.jo_product.prd_first_page	
-sl	Xp_job.jo_schedul.sch_security_lvl	TOP_SECRET, SECRET,

		CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-sm	Xp_job.job_document.doc_modes	
-sp	Xp_job.job_product.prd_form.fm_se pa-rator	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-st	Xp_job.job_info.j_state	ST_TOP, ST_SUSPEND, ST_WAIT
-te	add_job_params.job_template	
-tm	Xp_job.job_product.prd_form.fm_top	
-to	Xp_job.job_product.prd_select.sel_t o	
-tp	Xp_job.job_product.prd_font.ft_typef ace	
-tw	Xp_job.job_product.prd_tab_width	
-vm	Xp_job.job_product.prd_font.ft_vm	
-wa	Xp_job.job_schedul.sch_wait_activat ion	
-wm	Xp_job.job_product.prd_wrap_mode	WR_NONE, WR_WRAP, WR_CUT

Table 31: xpadd -job/add_all_cli()

xpadd -jtp/add_all_cli (Xp_job_temp.jt_name)

Optio n	Structure (element)	Permissible values
-ah	Xp_job_temp.jt_ad_host	
-at	Xp_job_temp.jt_job.job_schedul.sch_time _after	
-cp	Xp_job_temp.jt_job.job_document.doc_c opy	
-cr	Xp_job_temp.jt_job.job_pro-duct.prd_ove rwrite_lrs.lr_call_resource	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cs	Xp_job_temp.jt_job.job_pro-duct.prd_font .ft_style	XP_ITALICS, XP_STRAIGHT
-ct	Xp_job_temp.jt_job.job_docu- ment.doc_content_type	
-de	Xp_job_temp.jt_job.job_ schedul.sch_destin	
-dp	Xp_job_temp.jt_job.job_product.prd_dire ct_prt	
-dr	Xp_job_temp.jt_job.job_ document.doc_doc_ref	
-ev	Xp_job_temp.jt_job.job_	EV_SENT_TO_REM_DO

	event.ev_notify.no_events	MAIN, EV_REJECT_FR, REM_DO-MAIN, EV_RES UMED, EV_MODIFIED, EV_USER, EV_DEVICE_ERROR, EV_ABORTED, EV_SUSPEND, EV_COMPLETED, EV_STARTED, EV_NONE_NO, EV_ALL
-fc	Xp_job_temp.jt_job.jo_ product.prd_copies [XP_SFT_CPY]	
-ff	Xp_job_temp.jt_job.jo_ product.prd_ ff_mode	FF_ALL, FF_BETWEEN, FF_AFTER, FF_BEFORE, FF_NONE
-fm	Xp_job_temp.jt_job.jo_ product.prd_form.fm_name	
-fp	Xp_job_temp.jt_job.jo_ product.prd_copies [XP_HRD_CPY]	
-fr	Xp_job_temp.jt_job.jo_ product.prd_select.sel_from	
-ft	Xp_job_temp.jt_job.jo_ product.prd_font.ft_name	
-hm	Xp_job_temp.jt_job.jo_ product.prd_font.ft_hm	
-hn	Xp_job_temp.jt_job.jo_ product.prd_hopper	
-jc	Xp_job_temp.jt_job.jo_ info.j_comment	
-je	Xp_job_temp.jt_job.jo_ product.prd_form.fm_end	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-js	Xp_job_temp.jt_job.jo_ product.prd_form.fm_start	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-jt	Xp_job_temp.jt_job.jo_ info.j_title	
-la	Xp_job_temp.jt_job.□□□jo_ product.prd_font.ft_language	
-ld	Xp_job_temp.jt_job.□□□jo_ document.doc_profile.prf_files	
-ll	Xp_job_temp.jt_job.jo_ product.prd_form.fm_line	
-lm	Xp_job_temp.jt_job.jo_ product.prd_form.fm_left	
-lr	Xp_job_temp.jt_job.jo_ product.prd_lrs	
-ls	Xp_job_temp.jt_job.jo_ product.prd_form.fm_layup	XP_PORTRAIT, XP_LANDSCAPE

-mt	Xp_job.document.doc_profile.prf_map_table	
-na	Xp_job_temp.jt_job.job_event.ev_notify.no_address	
-nd	Xp_job_temp.jt_job.job_event.ev_notify.no_descriptor	
-nf	Xp_job_temp.jt_job.job_document.doc_no_filter	
-nm	Xp_job_temp.jt_job.job_event.ev_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-nv	Xp_job_temp.jt_job.job_product.prd_novalidation	
-ob	Xp_job_temp.jt_job.job_product.prd_prd_obin	
-pc	Xp_job_temp.jt_job.job_dev.dv_pcl	
-pl	Xp_job_temp.jt_job.job_product.prd_form.fm_page	
-pq	Xp_job_temp.jt_job.job_product.prd_font.ft_quality	XP_HIGH, XP_NORMAL, XP_DRAFT
-pr	Xp_job_temp.jt_job.job_schedul.sch_priority	
-ps	Xp_job_temp.jt_job.job_product.prd_select.sel_page_select	
-rm	Xp_job_temp.jt_job.job_document.doc_remove	
-rp	Xp_job_temp.jt_job.job_schedul.sch_retention	RP_INFINITE
-rt	Xp_job_temp.jt_job.job_schedul.sch_time_retention	
-rw	Xp_job_temp.jt_job.job_document.doc_raw_mode	
-sd	Xp_job_temp.jt_job.job_product.prd_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sf	Xp_job_temp.jt_job.job_product.prd_first_page	
-sl	Xp_job_temp.jt_job.job_schedul.sch_securit_lvl	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-sm	Xp_job_temp.jt_job.job_document.doc_modes	
-sp	Xp_job_temp.jt_job.job_	XP_NONE_SHEET,

	product.prd_form.fm_separator	XP_FULL_SHEET, XP_BRIEF_SHEET
-st	Xp_job_temp.jt_job.jo_info.j_state	ST_TOP, ST_SUSPEND, ST_WAIT
-tm	Xp_job_temp.jt_job.jo_product.prd_form.fm_top	
-to	Xp_job_temp.jt_job.jo_product.prd_select.sel_to	
-tp	Xp_job_temp.jt_job.jo_product.prd_font.ft_typeface	
-tw	Xp_job_temp.jt_job.jo_product.prd_tab_width	
-vm	Xp_job_temp.jt_job.jo_product.prd_font.ft_vm	
-wa	Xp_job_temp.jt_job.jo_schedul.sch_wait_activation	
-wm	Xp_job_temp.jt_job.jo_product.prd_wrap_mode	WR_NONE, WR_WRAP, WR_CUT

Table 32: xpadd -jtp/add_all_cli (Xp_job_temp.jt_name)

xpadd -pcl/add_all_cli() (Xp_pcl.pl_name)

Option	Structure (element)	Permissible values
-ca	Xp_pcl.pl_comment [CM_ADMIN]	
-cu	Xp_pcl.pl_comment [CM_USER]	
-fi	Xp_pcl.pl_file	
-ho	Xp_pcl.pl_host	

Table 33: xpadd -pcl/add_all_cli() (Xp_pcl.pl_name)

xpadd -prm (Xp_perm.pn_name)

Option	Structure (element)	Permissible values
-ah	Xp_perm.pn_ad_host	
-al	Xp_perm.pn_allow	
-ca	Xp_perm.pn_comment	
-dn	Xp_perm.pn_deny	
-pt	Xp_perm.pn_type	FM, US

Table 34: xpadd -prm (Xp_perm.pn_name)

xpadd -rec/add_all_cli() (Xp_rec_rule.rr_name)

Option	Structure (element)	Permissible values
-ca	Xp_rec_rule.rr_comment	
-fi	Xp_rec_rule.rr_file	
-ho	Xp_rec_rule.rr_host	

Table 35: xpadd -rec/add_all_cli() (Xp_rec_rule.rr_name)

xpadd -spv/add_all_cli() (Xp_superv.sp_name)

Option	Structure (element)	Permissible values
-ca	Xp_superv.sp_comment	
-ho	Xp_superv.sp_net.nt_host	
-se	Xp_superv.sp_servname	

Table 36: xpadd -spv/add_all_cli() (Xp_superv.sp_name)

xpadd -srv/add_all_cli() (Xp_server.sv_name)

Option	Structure (element)	Permissible values
-ca	Xp_server.sv_comment	
-ho	Xp_server.sv_net.nt_host	
-mj	Xp_server.sv_jbmax	XP_NO_LIMIT
-sp	Xp_server.sv_schedpol	XP_BIG, XP_SMALL, XP_PRIORITY, XP_SECURITY
-up	Xp_server.sv_perm	

Table 37: xpadd -srv/add_all_cli() (Xp_server.sv_name)

xpadd -usr/add_all_cli() (ECM_security.sc_usr_name)

Option	Structure (element)	Permissible values
-aj	ECM_security.sc_jobs_allowed	XP_NO_LIMIT
-ap	ECM_security.sc_pages_allowed	XP_NO_LIMIT
-ca	ECM_security.sc_comment	
-ho	ECM_security.sc_host	
-nc	ECM_security.sc_max_copies	XP_NO_LIMIT
-pr	ECM_security.sc_max_priority	XP_UNDEFINED
-pv	ECM_security.sc_privileges	ALL_PRIVIL, NONE_PRIVIL, ANY_DEVICE, TOP_PRIVIL
-sl	ECM_security.sc_securit_lvl	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-ua	ECM_security.sc_usr_account	

Table 38: xpadd -usr/add_all_cli() (ECM_security.sc_usr_name)

8.3.2 The xpchange command**xpchange -dev/chg_dev_cli() (ECM_printer.dv_name)**

Option	Structure (element)	Permissible values
-is	chg_dev_params.no_redir_jobs	

-ju	chg_dev_params.justification	
-mf	chg_dev_params.cur_form	
-np	chg_dev_params.change_type	CH_UNTIL_END, CH_NEXT_PAGE, CH_AGAIN
-pr	chg_dev_params.new_priority	
-rd	chg_dev_params.dv_redirect	
-sf	chg_dev_params.change_type	
-si	chg_dev_params.new_spoolin	CH_OFF, CH_ON
-so	chg_dev_params.new_spoolout	CH_OFF, CH_ON
-st	chg_dev_params.new_state	CH_NOT_ACTIVE, CH_ACTIVE
-uf	chg_dev_params.form_manip	
-wn	chg_dev_params.when_tbd	WT_DELETE, WT_NOW, WT_END, WT_LAST

Table 39: xpchange -dev/chg_dev_cli() (ECM_printer.dv_name)

xpchange -dgr/chg_dgr_cli() (Xp_dgroup.dg_name)

Option	Structure (element)	Permissible values
-ju	chg_dgr_params.justification	
-si	chg_dgr_params.new_spoolin	CH_OFF, CH_ON

Table 40: xpchange -dgr/chg_dgr_cli() (Xp_dgroup.dg_name)

xpchange -gtw/chg_gtw_cli() (Xp_gateway.gw_name)

Option	Structure (element)	Permissible values
-gi	chg_gtw_params.new_in	CH_OFF, CH_ON
-go	chg_gtw_params.new_out	CH_OFF, CH_ON
-ju	chg_gtw_params.justification	
-sa	chg_gtw_params.activation	XP_GW_COLD, XP_GW_WARM
-st	chg_gtw_params.new_state	CH_ACTIVE, CH_NOT_ACTIVE
-wn	chg_gtw_params.when_tbd	WT_GW_LAST_TRANS, WT_GW_NOW

Table 41: xpchange -gtw/chg_gtw_cli() (Xp_gateway.gw_name)

xpchange -job/chg_job_cli() (Xp_job.job_id.jid_jid)

Option	Structure (element)	Permissible values
-ps	chg_job_params.page_select	
-st	chg_job_params.new_state	CH_WAIT, CH_SCHED_ACTIVE, CH_SUSPEND,

	CH_TOP
--	--------

Table 42: xpchange -job/chg_job_cli() (Xp_job.job_id.job_id)

xpchange -spv/chg_spv_cli() (Xp_superv.sp_name)

Option	Structure (element)	Permissible values
-ju	chg_spv_params.justification	
-st	chg_spv_params.new_state	CH_ACTIVE, CH_NOT_ACTIVE

Table 43: xpchange -spv/chg_spv_cli() (Xp_superv.sp_name)

xpchange -srv/chg_srv_cli() (Xp_server.sv_name)

Option	Structure (element)	Permissible values
-ju	chg_srv_params.justification	
-sa	chg_srv_params.activation	XP_COLD, XP_WARM
-si	chg_srv_params.new_spoolin	CH_OFF, CH_ON
-so	chg_srv_params.new_spoolout	CH_OFF, CH_ON
-st	chg_srv_params.new_state	CH_ACTIVE, CH_NOT_ACTIVE

Table 44: xpchange -srv/chg_srv_cli() (Xp_server.sv_name)

xpchange -usr/chg_usr_cli() (ECM_security.sc_usr_name)

Option	Structure (element)	Permissible values
-cj	chg_usr_params.cons_jobs	
-cp	chg_usr_params.cons_pages	

Table 45: xpchange -usr/chg_usr_cli() (ECM_security.sc_usr_name)

8.3.3 The xpcopy command**xpcopy -dev/cpy_all_cli()**

The call xpcopy -dev d1 d2 is converted with the cpy_all_cli() function as follows: cpy_all_cli(d1,d2), where d1 and d2 are pointers to the two structures of the type ECM_printer.

Option	Structure (element)	Permissible values
-aa	ECM_printer.dv_aut_act	
-ad	ECM_printer.dv_admin	
-ca	ECM_printer.dv_comment [CM_ADMIN]	
-co	ECM_printer.dv_connect_type	XP_LAN, XP_SCSI, XP_PARALLEL, XP_SERIAL
-ct	ECM_printer.dv_all_type	
-cu	ECM_printer.dv_comment [CM_USER]	
-da	ECM_printer.dv_add	
-ep	ECM_printer.dv_enable_poll	

-ev	ECM_printer.dv_notify.no_events	EV_DEVICE_ERROR, EV_MOUNTING, EV_NONE_NO, EV_ALL
-fl	ECM_printer.dv_filter	
-fm	ECM_printer.dv_dft_form	
-fp	ECM_printer.dv_fo_perm	
-ho	ECM_printer.dv_host	
-kc	ECM_printer.dv_keep_connection	XP_CO_PERM, XP_CO_TEMP
-lr	ECM_printer.dv_lrs	
-me	ECM_printer.dv_resource_mngr	
-ms	ECM_printer.dv_start_sheet	XP_FULL_SHEET, XP_NONE_SHEET, XP_FREE_SHEET, XP_BRIEF_SHEET
-na	ECM_printer.dv_notify.no_address	
-nd	ECM_printer.dv_notify.no_descriptor	
-nm	ECM_printer.dv_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-nr	ECM_printer.dv_notify.no_frequency	
-or	ECM_printer.dv_open_retry_time	OR_INFINITE
-pb	ECM_printer.dv_proto.cn_parity	XP_NONE_PAR, XP_EVEN_PAR, XP_ODD_PAR
-pc	ECM_printer.dv_pcl	
-po	ECM_printer.dv_proto.cn_protocol	XP_ETX, XP_RXON, XP_XON
-pr	ECM_printer.dv_priority	
-ps	ECM_printer.dv_paper_feed	SINGLE_SHEET_MANUAL, LISTING_PAPER, SINGLE_SHEET_AUTO
-rr	ECM_printer.dv_rec_rule	
-sb	ECM_printer.dv_proto.cn_bits	8, 7
-sl	ECM_printer.dv_security	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-sp	ECM_printer.dv_proto.cn_speed	38400, 19200, 9600, 4800, 2400, 1200
-su	ECM_printer.dv_supervisor	
-sy	ECM_printer.dv_stty	
-up	ECM_printer.dv_usr_perm	

Table 46: xpcopy -dev/cpy_all_cli()

xpcopy -dgr/cpy_all_cli()

The call xpcopy -dgr d1 d2 is converted with the cpy_all_cli() function as follows: cpy_all_cli(d1,d2), where d1 and d2 are pointers to the two structures of the type Xp_dgroup.

Option	Structure (element)	Permissible values
-ad	Xp_dgroup.dg_admin	
-ah	Xp_dgroup.dg_ad_host	
-ca	Xp_dgroup.dg_comment [CM_ADMIN]	
-cu	Xp_dgroup.dg_comment [CM_USER]	
-dl	Xp_dgroup.dg_devlst	
-up	Xp_dgroup.dg_uperm	

Table 47: xpcopy -dgr/cpy_all_cli()

xpcopy -flt /cpy_all_cli()

The call xpcopy -flt f1 f2 is converted with the cpy_all_cli() function as follows: cpy_all_cli(f1,f2), where f1 and f2 are pointers to the two structures of the type Xp_filter.

Option	Structure (element)	Permissible values
-ad	Xp_filter.fl_admin	
-ah	Xp_filter.fl_ad_host	
-ca	Xp_filter.fl_comment [CM_ADMIN]	
-cm	Xp_filter.fl_command	
-cu	Xp_filter.fl_comment [CM_USER]	
-ho	Xp_filter.fl_host_lst	
-it	Xp_filter.fl_inp_type	
-op	Xp_filter.fl_options	
-ot	Xp_filter.fl_out_type	

Table 48: xpcopy -flt /cpy_all_cli()

xpcopy -fnt/cpy_all_cli()

The call xpcopy -fnt f1 f2 is converted with the cpy_all_cli() function as follows: cpy_all_cli(f1,f2), where f1 and f2 are pointers to the two structures of the type Xp_font.

Option	Structure (element)	Permissible values
-ad	Xp_font.ft_admin	
-ah	Xp_font.ft_ad_host	
-ca	Xp_font.ft_comment [CM_ADMIN]	
-cs	Xp_font.ft_style	XP_ITALICS, XP_STRAIGHT
-cu	Xp_font.ft_comment [CM_USER]	
-hm	Xp_font.ft_hm	
-la	Xp_font.ft_language	
-pq	Xp_font.ft_quality	XP_HIGH,

		XP_NORMAL, XP_DRAFT
-tm	Xp_font.ft_type_man	
-tp	Xp_font.ft_typeface	
-vm	Xp_font.ft_vm	

Table 49: xpcopy -fnt/cpy_all_cli()

xpcopy -frm/cpy_all_cli()

The call xpcopy -frm f1 f2 is converted with the cpy_all_cli() function as follows: cpy_all_cli(f1,f2), where f1 and f2 are pointers to the two structures of the type Xp_form.

Option	Structure (element)	Permissible values
-ad	Xp_form.fm_admin	
-ah	Xp_form.fm_ad_host	
-ca	Xp_form.fm_comment [CM_ADMIN]	
-cu	Xp_form.fm_comment [CM_USER]	
-ft	Xp_form.fm_dft_font	
-je	Xp_form.fm_end	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-js	Xp_form.fm_start	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-ll	Xp_form.fm_line	
-lm	Xp_form.fm_left	
-lr	Xp_form.fm_lrs	
-ls	Xp_form.fm_layup	XP_PORTRAIT, XP_LANDSCAPE
-mf	Xp_form.fm_ft_mandatory	
-mm	Xp_form.fm_mount_method	XP_MANUAL, XP_AUTOMATIC
-nt	Xp_form.fm_notify.no_treshold	
-ph	Xp_form.fm_paper_size [XP_HEIGHT]	
-pl	Xp_form.fm_page	
-pw	Xp_form.fm_paper_size [XP_WIDTH]	
-sd	Xp_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sp	Xp_form.fm_separator	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-tm	Xp_form.fm_top	
-up	Xp_form.fm_user_perm	

Table 50: xpcopy -frm/cpy_all_cli()

xpcopy -gtw/cpy_all_cli()

The call xpcopy -gtw g1 g2 is converted with the cpy_all_cli() function as follows: cpy_all_cli(g1,g2), where g1 and g2 are pointers to the two structures of the type Xp_gateway.

Option	Structure (element)	Permissible values
-aa	Xp_gateway.gw_aut_act	
-ad	Xp_gateway.gw_admin	
-ag	Xp_gateway.gw_allow_list	
-bl	Xp_gateway.gw_bkwd_lks	
-ca	Xp_gateway.gw_comment	
-ga	Xp_gateway.gw_add	
-ho	Xp_gateway.gw_net.nt_host	
-np	Xp_gateway.gw_proto	
-oh	Xp_gateway.gw_orig_host	
-on	Xp_gateway.gw_orig_name	
-op	Xp_gateway.gw_options	
-pp	Xp_gateway.gw_print_proto	
-tp	Xp_gateway.gw_type	XP_PARTNER, XP_LOCAL
-up	Xp_gateway.gw_uperm	

Table 51: xpcopy -gtw/cpy_all_cli()

xpcopy -jtp/cpy_all_cli()

The call xpcopy -jtp j1 j2 is converted with the cpy_all_cli() function as follows: cpy_all_cli(j1,j2), where j1 and j2 are pointers to the two structures of the type Xp_job.

Option	Structure (element)	Permissible values
-ad	Xp_job_temp.jt_admin	
-ah	Xp_job_temp.jt_ad_host	
-at	Xp_job_temp.jt_job.jo_sche- dul.sch_time_after	
-cp	Xp_job_temp.jt_job.jo_docu- ment.doc_copy	
-cr	Xp_job_temp.jt_job.jo_pro- duct.prd_overwrite_lrs.lr_call_reso- urce	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cs	Xp_job_temp.jt_job.jo_pro- duct.prd_font.ft_style	XP_ITALICS, XP_STRAIGHT
-ct	Xp_job_temp.jt_job.jo_docu- ment.doc_content_type	
-de	Xp_job_temp.jt_job.jo_ schedul.sch_destin	

-dp	Xp_job_temp.jt_job.jo_pro- duct.prd_direct_prt	
-dr	Xp_job_temp.jt_job.jo_ document.doc_doc_ref	
-ev	Xp_job_temp.jt_job.jo_ event.ev_notify.no_events	EV_SENT_TO_REM_DOMAI N, EV_REJECT_FR_REM_DOM AIN,EV_RESUMED, EV_MODIFIED, EV_USER, EV_DEVICE_ERROR, EV_ABORTED, EV_SUSPEND, EV_COMPLETED, EV_STARTED, EV_NONE_NO, EV_ALL
-fc	Xp_job_temp.jt_job.jo_ product.prd_copies [XP_SFT_CPY]	
-ff	Xp_job_temp.jt_job.jo_pro- duct.prd_ff_mode	FF_ALL, FF_BETWEEN, FF_AFTER, FF_BEFORE, FF_NONE
-fm	Xp_job_temp.jt_job.jo_ product.prd_form.fm_name	
-fp	Xp_job_temp.jt_job.jo_ product.prd_copies [XP_HRD_CPY]	
-fr	Xp_job_temp.jt_job.jo_ product.prd_select.sel_from	
-ft	Xp_job_temp.jt_job.jo_ product.prd_font.ft_name	
-hm	Xp_job_temp.jt_job.jo_ product.prd_font.ft_hm	
-hn	Xp_job_temp.jt_job.jo_ product.prd_hopper	
-jc	Xp_job_temp.jt_job.jo_ info.j_comment	
-je	Xp_job_temp.jt_job.jo_ product.prd_form.fm_end	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-js	Xp_job_temp.jt_job.jo_ product.prd_form.fm_start	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-jt	Xp_job_temp.jt_job.jo_info.j_title	
-la	Xp_job_temp.jt_job.jo_ product.prd_font.ft_language	
-ld	Xp_job_temp.jt_job.jo_ document.doc_profile.prf_files	

-ll	Xp_job_temp.jt_job.jo_ product.prd_form.fm_line	
-lm	Xp_job_temp.jt_job.jo_ product.prd_form.fm_left	
-lr	Xp_job_temp.jt_job.jo_product.prd_ _lrs	
-ls	Xp_job_temp.jt_job.jo_ product.prd_form.fm_layup	XP_PORTRAIT, XP_LANDSCAPE
-mt	Xp_job.document.doc_profile.prf_ map_table	
-na	Xp_job_temp.jt_job.jo_ event.ev_notify.no_address	
-nd	Xp_job_temp.jt_job.jo_ event.ev_notify.no_descriptor	
-nf	Xp_job_temp.jt_job.jo_ document.doc_no_filter	
-nm	Xp_job_temp.jt_job.jo_ event.ev_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-nv	Xp_job_temp.jt_job.jo_ product.prd_novalidation	
-ob	Xp_job_temp.jt_job.jo_ product.prd_prd_obin	
-pc	Xp_job_temp.jt_job.jo_dev.dv_pcl	
-pl	Xp_job_temp.jt_job.jo_ product.prd_form.fm_page	
-pq	Xp_job_temp.jt_job.jo_ product.prd_font.ft_quality	XP_HIGH, XP_NORMAL, XP_DRAFT
-pr	Xp_job_temp.jt_job.jo_ schedul.sch_priority	
-ps	Xp_job_temp.jt_job.jo_ product.prd_select.sel_page_selec t	
-rm	Xp_job_temp.jt_job.jo_ document.doc_remove	
-rp	Xp_job_temp.jt_job.jo_ schedul.sch_retention	RP_INFINITE
-rt	Xp_job_temp.jt_job.jo_ schedul.sch_time_retention	
-rw	Xp_job_temp.jt_job.jo_ document.doc_raw_mode	
-sd	Xp_job_temp.jt_job.jo_ product.prd_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sf	Xp_job_temp.jt_job.jo_ product.prd_first_page	

-sl	Xp_job_temp.jt_job.jo_schedul.sch_securit_lvl	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-sm	Xp_job_temp.jt_job.jo_document.doc_modes	
-sp	Xp_job_temp.jt_job.jo_product.prd_form.fm_separator	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-st	Xp_job_temp.jt_job.jo_info.j_state	ST_TOP, ST_SUSPEND, ST_WAIT
-tm	Xp_job_temp.jt_job.jo_product.prd_form.fm_top	
-to	Xp_job_temp.jt_job.jo_product.prd_select.sel_to	
-tp	Xp_job_temp.jt_job.jo_product.prd_font.ft_typeface	
-tw	Xp_job_temp.jt_job.jo_product.prd_tab_width	
-vm	Xp_job_temp.jt_job.jo_product.prd_font.ft_vm	
-wm	Xp_job_temp.jt_job.jo_product.prd_wrap_mode	WR_NONE, WR_WRAP, WR_CUT

Table 52: xpcopy -jtp/cpy_all_cli()

xpcopy -lrs/cpy_all_cli()

Option	Structure (element)	Permissible values
-ad	Xp_soft_load.lr_admin	
-ah	Xp_soft_load.lr_ad_host	
-ca	Xp_soft_load.lr_comment[CM_ADMIN]	
-cn	Xp_soft_load.lr_class	
-cr	Xp_soft_load.lr_call_resource	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cu	Xp_soft_load.lr_comment[CM_USER]	
-dl	Xp_soft_load.lr_devlst	
-ec	Xp_soft_load.lr_encapsulate	
-fi	Xp_soft_load.lr_file	
-pc	Xp_soft_load.lr_pcl	
-pe	Xp_soft_load.lr_permanent	
-pi	Xp_soft_load.lr_int_id	
-tp	Xp_soft_load.lr_type	

Table 53: xpcopy -lrs/cpy_all_cli()

xpcopy -prm/cpy_all_cli()

The call `xpcopy -prm p1 p2` is converted with the `cpy_all_cli()` function as follows: `cpy_all_cli(p1,p2)`, where `p1` and `p2` are pointers to the two structures of the type `Xp_perm`. □

Option	Structure (element)	Permissible values
-ad	Xp_perm.pn_admin	
-ah	Xp_perm.pn_ad_host	
-al	Xp_perm.pn_allow	
-ca	Xp_perm.pn_comment	
-dn	Xp_perm.pn_deny	
-pt	Xp_perm.pn_type	FM, US

Table 54: xpcopy -prm/cpy_all_cli()

xpcopy -srv/cpy_all_cli()

The call `xpcopy -srv s1 s2` is converted with the `cpy_all_cli()` function as follows: `cpy_all_cli(s1,s2)`, where `s1` and `s2` are pointers to the two structures of the type `Xp_server`. □

Option	Structure (element)	Permissible values
-ad	Xp_server.sv_admin	
-ca	Xp_server.sv_comment	
-ho	Xp_server.sv_net.nt_host	
-mj	Xp_server.sv_jbmax	XP_NO_LIMIT
-sp	Xp_server.sv_schedpol	XP_BIG, XP_SMALL, XP_PRIORITY, XP_SECURITY
-up	Xp_server.sv_perm	

Table 55: xpcopy -srv/cpy_all_cli()

xpcopy -spv/cpy_all_cli()

The call `xpcopy -spv s1 s2` is converted with the `cpy_all_cli()` function as follows: `cpy_all_cli(s1,s2)`, where `s1` and `s2` are pointers to the two structures of the type `Xp_superv`.

Option	Structure (element)	Permissible values
-ad	Xp_superv.sp_admin	
-ca	Xp_superv.sp_comment	
-ho	Xp_superv.sp_net.nt_host	
-se	Xp_superv.sp_servname	

Table 56: xpcopy -spv/cpy_all_cli()

xpcopy -usr/cpy_all_cli()

The call `xpcopy -usr u1 u2` is converted with the `cpy_all_cli()` function as follows: `cpy_all_cli(u1,u2)`, where `u1` and `u2` are pointers to the two structures of the type `ECM_security`.

Option	Structure (element)	Permissible values
-aj	ECM_security.sc_jobs_allowed	XP_NO_LIMIT
-ap	ECM_security.sc_pages_allowed	XP_NO_LIMIT

-ca	ECM_security.sc_comment	
-cl	ECM_security.sc_host	
-ho	ECM_security.sc_host	
-nc	ECM_security.sc_max_copies	XP_NO_LIMIT
-pr	ECM_security.sc_max_priority	XP_UNDEFINED
-pv	ECM_security.sc_privileges	ALL_PRIVIL, NONE_PRIVIL, ANY_DEVICE, TOP_PRIVIL
-sl	ECM_security.sc_securit_lvl	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-ua	ECM_security.sc_usr_account	

Table 57: xpcopy -usr/cpy_all_cli()

8.3.4 The xpdel command

Option	Structure
xpdel -dev/del_all_cli	ECM_printer.dv_name
xpdel -dgr/del_all_cli	Xp_dgroup.dg_name
xpdel -flt/del_all_cli	Xp_filter.fl_name
xpdel -fnt/del_all_cli	Xp_font.ft_name
xpdel -frm/del_all_cli	Xp_form.fm_name
xpdel -gtw/del_all_cli	Xp_gateway.gw_name
xpdel -hos/del_all_cli	Xp_host.ho_name
xpdel -job/del_job_cli	Xp_job.jo_id.jid_jid
xpdel -jtp/del_all_cli	Xp_job_temp.jt_name
xpdel -lrs/del_all_cli	Xp_soft_load.lr_name
xpdel -pcl/del_all_cli	Xp_pcl.pl_name
xpdel -prm/del_all_cli	Xp_perm.pn_name
xpdel -rec/del_all_cli	Xp_rec_rule.rr_name
xpdel -spv/del_all_cli	Xp_superv.sp_name
xpdel -srv/del_all_cli	Xp_server.sv_name
xpdel -usr/del_all_cli	ECM_security.sc_usr_name

Table 58: xpdel command

8.3.5 The xpmmod command

xpmmod -dev/mod_all_cli() (ECM.printer-dv_name)

Option	Structure (element)	Permissible values
-aa	ECM_printer.dv_aut_act	

-ad	ECM_printer.dv_admin	
-ca	ECM_printer.dv_comment [CM_ADMIN]	
-co	ECM_printer.dv_connect_type	XP_LAN, XP_SCSI, XP_PARALLEL, XP_SERIAL
-ct	ECM_printer.dv_all_type	
-cu	ECM_printer.dv_comment [CM_USER]	
-da	ECM_printer.dv_add	
-ep	ECM_printer.dv_enable_poll	
-ev	ECM_printer.dv_notify.no_events	EV_DEVICE_ERROR, EV_MOUNTING, EV_NONE_NO, EV_ALL
-fl	ECM_printer.dv_filter	
-fm	ECM_printer.dv_dft_form	
-fp	ECM_printer.dv_fo_perm	
-ho	ECM_printer.dv_host	
-kc	ECM_printer.dv_keep_connection	XP_CO_PERM, XP_CO_TEMP
-lr	ECM_printer.dv_lrs	
-me	ECM_printer.dv_resource_mngr	
-ms	ECM_printer.dv_start_sheet	XP_FULL_SHEET, XP_NONE_SHEET, XP_FREE_SHEET, XP_BRIEF_SHEET
-na	ECM_printer.dv_notify.no_address	
-nd	ECM_printer.dv_notify.no_descriptor	
-nm	ECM_printer.dv_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-nr	ECM_printer.dv_notify.no_frequency	
-or	ECM_printer.dv_open_retry_time	OR_INFINITE
-pb	ECM_printer.dv_proto.cn_parity	XP_NONE_PAR, XP_EVEN_PAR, XP_ODD_PAR
-pc	ECM_printer.dv_pcl	
-po	ECM_printer.dv_proto.cn_protocol	XP_ETX, XP_RXON, XP_XON
-pr	ECM_printer.dv_priority	
-ps	ECM_printer.dv_paper_feed	SINGLE_SHEET_MANUAL, LISTING_PAPER, SINGLE_SHEET_AUTO

-rr	ECM_printer.dv_rec_rule	
-sb	ECM_printer.dv_proto.cn_bits	8, 7
-sl	ECM_printer.dv_security	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-sp	ECM_printer.dv_proto.cn_speed	38400, 19200, 9600, 4800, 2400, 1200
-su	ECM_printer.dv_supervisor	
-sy	ECM_printer.dv_stty	
-up	ECM_printer.dv_usr_perm	

Table 59: xpmodev/mod_all_cli() (ECM_printer-dv_name)

xpmod -dgr/mod_all_cli() (Xp_dgroup.dg_name)

Option	Structure (element)	Permissible values
-ad	Xp_dgroup.dg_admin	
-ah	Xp_dgroup.dg_ad_host	
-ca	Xp_dgroup.dg_comment [CM_ADMIN]	
-cu	Xp_dgroup.dg_comment [CM_USER]	
-dl	Xp_dgroup.dg_devlst	
-up	Xp_dgroup.dg_uperm	

Table 60: xpmod -dgr/mod_all_cli() (Xp_dgroup.dg_name)

xpmod -flt/mod_all_cli() (Xp_filter.fl_name)

Option	Structure (element)	Permissible values
-ad	Xp_filter.fl_admin	
-ah	Xp_filter.fl_ad_host	
-ca	Xp_filter.fl_comment [CM_ADMIN]	
-cm	Xp_filter.fl_command	
-cu	Xp_filter.fl_comment [CM_USER]	
-ho	Xp_filter.fl_host_lst	
-it	Xp_filter.fl_inp_type	
-op	Xp_filter.fl_options	
-ot	Xp_filter.fl_out_type	XP_FAST, XP_SLOW

Table 61: xpmod -flt/mod_all_cli() (Xp_filter.fl_name)

xpmod -fnt/mod_all_cli() (Xp_font.ft_name)

Option	Structure (element)	Permissible values
-ad	Xp_font.ft_admin	
-ah	Xp_font.ft_ad_host	
-ca	Xp_font.ft_comment [CM_ADMIN]	

-cs	Xp_font.ft_style	XP_ITALICS, XP_STRAIGHT
-cu	Xp_font.ft_comment [CM_USER]	
-hm	Xp_font.ft_hm	
-la	Xp_font.ft_language	
-pq	Xp_font.ft_quality	XP_HIGH, XP_NORMAL, XP_DRAFT
-tm	Xp_font.ft_type_man	
-tp	Xp_font.ft_typeface	
-vm	Xp_font.ft_vm	

Table 62: xpmmod -fnt/mod_all_cli() (Xp_font.ft_name)

xpmmod -frm/mod_all_cli() (Xp_form.fm_name)

Option	Structure (element)	Permissible values
-ad	Xp_form.fm_admin	
-ah	Xp_form.fm_ad_host	
-ca	Xp_form.fm_comment [CM_ADMIN]	
-ft	Xp_form.fm_dft_font	
-je	Xp_form.fm_end	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-js	Xp_form.fm_start	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-ll	Xp_form.fm_line	
-lm	Xp_form.fm_left	
-lr	Xp_form.fm_lrs	
-ls	Xp_form.fm_layup	XP_PORTRAIT, XP_LANDSCAPE
-mf	Xp_form.fm_ft_mandatory	
-mm	Xp_form.fm_mount_method	XP_MANUAL, XP_AUTOMATIC
-nt	Xp_form.fm_notify.no_treshold	
-ph	Xp_form.fm_paper_size [XP_HEIGHT]	
-pl	Xp_form.fm_page	
-pw	Xp_form.fm_paper_size [XP_WIDTH]	
-sd	Xp_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sp	Xp_form.fm_separator	XP_NONE_SHEET, XP_FULL_SHEET,

		XP_BRIEF_SHEET
-tm	Xp_form.fm_top	
-up	Xp_form.fm_user_perm	

Table 63: xpmmod -frm/mod_all_cli() (Xp_form.fm_name)

xpmmod -gtw/mod_all_cli() (Xp_gateway.gw_name)

Option	Structure (element)	Permissible values
-aa	Xp_gateway.gw_aut_act	
-ad	Xp_gateway.gw_admin	
-ag	Xp_gateway.gw_allow_list	
-bl	Xp_gateway.gw_bkwd_lks	
-ca	Xp_gateway.gw_comment	
-ga	Xp_gateway.gw_add	
-ho	Xp_gateway.gw_net.nt_host	
-np	Xp_gateway.gw_proto	
-oh	Xp_gateway.gw_orig_host	
-on	Xp_gateway.gw_orig_name	
-op	Xp_gateway.gw_options	
-pp	Xp_gateway.gw_print_proto	
-tp	Xp_gateway.gw_type	XP_PARTNER, XP_LOCAL
-up	Xp_gateway.gw_uperm	

Table 64: xpmmod -gtw/mod_all_cli() (Xp_gateway.gw_name)

xpmmod -hos/mod_all_cli() (Xp_host.ho_name)

Option	Structure (element)	Permissible values
-ca	Xp_host.ho_comment [CM_ADMIN]	
-cu	Xp_host.ho_comment [CM_USER]	
-de	Xp_host.ho_defdest	
-dp	Xp_host.ho_def_dev_pri	
-fi	Xp_host.ho_infile	
-fm	Xp_host.ho_defform	
-ft	Xp_host.ho_deffont	
-mp	Xp_host.ho_def_max_pri	
-np	Xp_host.ho_protocol	XP_ISO, XP_TCP_IP
-pd	Xp_host.ho_predecessor	
-po	Xp_host.ho_position	XP_POS_LAST, XP_POS_FIRST
-pr	Xp_host.ho_def_job_pri	

-re	Xp_host.ho_responsibility	XP_PARASITE, XP_SLAVE, XP_POT_MASTER
-sh	Xp_host.ho_supp_host	
-sr	Xp_host.ho_sub_root	

Table 65: xpmmod -hos/mod_all_cli() (Xp_host.ho_name)

xpmmod -job/mod_all_cli() (Xp_job.jo.jid_jid)

Option	Structure (element)	Permissible values
-at	Xp_job.jo_schedul.sch_time_after	
-cr	Xp_job.jo_product.prd_overwrite_lrs.lr_call_resource	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cs	Xp_job.jo_product.prd_font.ft_style	XP_ITALICS, XP_STRAIGHT
-de	Xp_job.jo_schedul.sch_destin	
-ev	Xp_job.jo_event.ev_notify.no_events	EV_SENT_TO_REM_DOMAIN, EV_REJECT_FR_REM_DOMAIN, EV_RESUMED, EV_MODIFIED, EV_USER, EV_DEVICE_ERROR, EV_ABORTED, EV_SUSPEND, EV_COMPLETED, EV_STARTED, EV_NONE_NO, EV_ALL
-fc	Xp_job.jo_product.prd_copies [XP_SFT_CPY]	
-ff	Xp_job.jo_product.prd_ff_mode	FF_ALL, FF_BETWEEN, FF_AFTER, FF_BEFORE, FF_NONE
-fm	Xp_job.jo_product.prd_form.fm_name	
-fp	Xp_job.jo_product.prd_copies [XP_HRD_CPY]	
-fr	Xp_job.jo_product.prd_select.sel_from	
-ft	Xp_job.jo_product.prd_font.ft_name	
-hm	Xp_job.jo_product.prd_font.ft_hm	
-hn	Xp_job.jo_product.prd_hopper	
-jc	Xp_job.jo_info.j_comment	
-je	Xp_job.jo_product.prd_form.fm_end	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET

-js	Xp_job.job_product.prd_form.fm_start	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-la	Xp_job.job_product.prd_font.ft_language	
-ld	Xp_job.job_document.doc_pro-file.prf_files	
-ll	Xp_job.job_product.prd_form.fm_line	
-lm	Xp_job.job_product.prd_form.fm_left	
-lr	Xp_job.job_product.prd_lrs	
-ls	Xp_job.job_product.prd_form.fm_layup	XP_PORTRAIT, XP_LANDSCAPE
-mt	Xp_job.document.doc_pro-file.prf_map_table	
-na	Xp_job.job_event.ev_notify.no_address	
-nd	Xp_job.job_event.ev_notify.no_descryptor	
-nm	Xp_job.job_event.ev_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-ob	Xp_job.job_product.prd_prd_obin	
-pc	Xp_job.job_dev.dv_pcl	
-pl	Xp_job.job_product.prd_form.fm_page	
-pq	Xp_job.job_product.prd_font.ft_quality	XP_HIGH, XP_NORMAL, XP_DRAFT
-pr	Xp_job.job_schedul.sch_priority	
-ps	Xp_job.job_product.prd_select.select_page_select	
-rm	Xp_job.job_document.doc_remove	
-rp	Xp_job.job_schedul.sch_retention	RP_INFINITE
-rt	Xp_job.job_schedul.sch_time_retention	
-rw	Xp_job.job_document.doc_raw_mode	
-sd	Xp_job.job_product.prd_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sf	Xp_job.job_product.prd_first_page	
-sl	Xp_job.job_schedul.sch_security_level	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED

-sp	Xp_job.job_product.prd_form.fm_s epa-rator	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-tm	Xp_job.job_product.prd_form.fm_t op	
-to	Xp_job.job_product.prd_select.sel _to	
-tp	Xp_job.job_product.prd_font.ft_typ e-face	
-tw	Xp_job.job_product.prd_tab_width	
-vm	Xp_job.job_product.prd_font.ft_vm	
-wa	Xp_job.job_schedul.sch_wait_acti- vation	
-wm	Xp_job.job_product.prd_wrap_mo de	WR_NONE, WR_WRAP, WR_CUT

Table 66: xpmmod -job/mod_all_cli() (Xp_job.job.jid_jid)

xpmmod -jtp/mod_all_cli() (Xp_job.temp.jt_name)

Option	Structure (element)	Permissible values
-ad	Xp_job_temp.jt_admin	
-ah	Xp_job_temp.jt_ad_host	
-at	Xp_job_temp.jt_job.job_sche- dul.sch_time_after	
-cp	Xp_job_temp.jt_job.job_docu- ment.doc_copy	
-cr	Xp_job_temp.jt_job.job_pro-duct.pr d_overwrite_lrs.lr_call_resource	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cs	Xp_job_temp.jt_job.job_pro- duct.prd_font.ft_style	XP_ITALICS, XP_STRAIGHT
-ct	Xp_job_temp.jt_job.job_docu- ment.doc_content_type	
-de	Xp_job_temp.jt_job.job_ schedul.sch_destin	
-dp	Xp_job_temp.jt_job.job_pro- duct.prd_direct_prt	
-dr	Xp_job_temp.jt_job.job_ document.doc_doc_ref	
-ev	Xp_job_temp.jt_job.job_ event.ev_notify.no_events	EV_SENT_TO_REM_DOMAIN , EV_REJECT_FR_REM_DOM AIN, EV_RESUMED, EV_MODIFIED, EV_USER, EV_DEVICE_ERROR, EV_ABORTED, EV_SUSPEND,

		EV_COMPLETED, EV_STARTED, EV_NONE_NO, EV_ALL
-fc	Xp_job_temp.jt_job.jo_ product.prd_copies [XP_SFT_CPY]	
-ff	Xp_job_temp.jt_job.jo_pro- duct.prd_ff_mode	FF_ALL, FF_BETWEEN, FF_AFTER, FF_BEFORE, FF_NONE
-fm	Xp_job_temp.jt_job.jo_ product.prd_form.fm_name	
-fp	Xp_job_temp.jt_job.jo_ product.prd_copies [XP_HRD_CPY]	
-fr	Xp_job_temp.jt_job.jo_ product.prd_select.sel_from	
-ft	Xp_job_temp.jt_job.jo_ product.prd_font.ft_name	
-hm	Xp_job_temp.jt_job.jo_ product.prd_font.ft_hm	
-hn	Xp_job_temp.jt_job.jo_ product.prd_hopper	
-jc	Xp_job_temp.jt_job.jo_ info.j_comment	
-je	Xp_job_temp.jt_job.jo_ product.prd_form.fm_end	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-js	Xp_job_temp.jt_job.jo_ product.prd_form.fm_start	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-jt	Xp_job_temp.jt_job.jo_info.j_title	
-la	Xp_job_temp.jt_job.jo_ product.prd_font.ft_language	
-ld	Xp_job_temp.jt_job.jo_ document.doc_profile.prf_files	
-ll	Xp_job_temp.jt_job.jo_ product.prd_form.fm_line	
-lm	Xp_job_temp.jt_job.jo_ product.prd_form.fm_left	
-lr	Xp_job_temp.jt_job.jo_product.pr d_lrs	
-ls	Xp_job_temp.jt_job.jo_ product.prd_form.fm_layup	XP_PORTRAIT, XP_LANDSCAPE
-mt	Xp_job.document.doc_profile.prf_ map_table	

-na	Xp_job_temp.jt_job.jo_ event.ev_notify.no_address	
-nd	Xp_job_temp.jt_job.jo_ event.ev_notify.no_descriptor	
-nf	Xp_job_temp.jt_job.jo_ document.doc_no_filter	
-nm	Xp_job_temp.jt_job.jo_ event.ev_notify.no_method	XP_FILE, XP_MAIL, XP_WRITE, XP_COMMAND
-nv	Xp_job_temp.jt_job.jo_ product.prp_novalidation	
-ob	Xp_job_temp.jt_job.jo_ product.prp_prd_obin	
-pc	Xp_job_temp.jt_job.jo_ dev.dv_pcl	
-pl	Xp_job_temp.jt_job.jo_ product.prp_form.fm_page	
-pq	Xp_job_temp.jt_job.jo_ product.prp_font.ft_quality	XP_HIGH, XP_NORMAL, XP_DRAFT
-pr	Xp_job_temp.jt_job.jo_ schedul.sch_priority	
-ps	Xp_job_temp.jt_job.jo_ product.prp_select.sel_page_sele ct	
-rm	Xp_job_temp.jt_job.jo_ document.doc_remove	
-rp	Xp_job_temp.jt_job.jo_ schedul.sch_retention	RP_INFINITE
-rt	Xp_job_temp.jt_job.jo_ schedul.sch_time_retention	
-rw	Xp_job_temp.jt_job.jo_ document.doc_raw_mode	
-sd	Xp_job_temp.jt_job.jo_ product.prp_form.fm_sides	XP_DOUBLE_NARROW, XP_DOUBLE_WIDE, XP_SINGLE
-sf	Xp_job_temp.jt_job.jo_ product.prp_first_page	
-sl	Xp_job_temp.jt_job.jo_ schedul.sch_securit_lvl	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-sm	Xp_job_temp.jt_job.jo_ document.doc_modes	
-sp	Xp_job_temp.jt_job.jo_ product.prp_form.fm_separator	XP_NONE_SHEET, XP_FULL_SHEET, XP_BRIEF_SHEET
-st	Xp_job_temp.jt_job.jo_ event.ev_notify.no_address	ST_TOP, ST_SUSPEND,

	info.j_state	ST_WAIT
-tm	Xp_job_temp.jt_job.jo_product.prd_form.fm_top	
-to	Xp_job_temp.jt_job.jo_product.prd_select.sel_to	
-tp	Xp_job_temp.jt_job.jo_product.prd_font.ft_typeface	
-tw	Xp_job_temp.jt_job.jo_product.prd_tab_width	
-vm	Xp_job_temp.jt_job.jo_product.prd_font.ft_vm	
-wa	Xp_job_temp.jt_job.jo_sche- dul.sch_wait_activation	
-wm	Xp_job_temp.jt_job.jo_product.prd_wrap_mode	WR_NONE, WR_WRAP, WR_CUT

Table 67: xpmmod -jtp/mod_all_cli() (Xp_job.temp.jt_name)

xpmmod -lrs/mod_all_cli() (Xp_soft_load.lr_name)

Option	Structure (element)	Permissible values
-ad	Xp_soft_load.lr_admin	
-ah	Xp_soft_load.lr_ad_host	
-ca	Xp_soft_load.lr_comment [CM_ADMIN]	
-cn	Xp_soft_load.lr_class	
-cr	Xp_soft_load.lr_call_resouce	XP_LR_NONE, XP_LR_FIRST, XP_LR_ALL
-cu	Xp_soft_load.lr_comment [CM_USER]	
-dl	Xp_soft_load.lr_devlst	
-ec	Xp_soft_load.lr_encapsulate	
-fi	Xp_soft_load.lr_file	
-pc	Xp_soft_load.lr_pcl	
-pe	Xp_soft_load.lr_permanent	
-pi	Xp_soft_load.lr_int_id	
-tp	Xp_soft_load.lr_type	

Table 68: xpmmod -lrs/mod_all_cli() (Xp_soft_load.lr_name)

xpmmod -pcl/mod_all_cli() (Xp_pcl.pl_name)

Option	Structure (element)	Permissible values
-ad	Xp_pcl.pl_admin	
-ca	Xp_pcl.pl_comment [CM_ADMIN]	
-cu	Xp_pcl.pl_comment [CM_USER]	
-fi	Xp_pcl.pl_file	

-ho	Xp_pcl.pl_host	
-----	----------------	--

Table 69: xpmmod -pcl/mod_all_cli() (Xp_pcl.pl_name)

xpmmod -prm/mod_all_cli() (Xp_perm.pm_name)

Option	Structure (element)	Permissible values
-ad	Xp_perm.pn_admin	
-ah	Xp_perm.pn_ad_host	
-al	Xp_perm.pn_allow	
-ca	Xp_perm.pn_comment	
-dn	Xp_perm.pn_deny	

Table 70: xpmmod -prm/mod_all_cli() (Xp_perm.pm_name)

xpmmod -rec/mod_all_cli() (Xp_rec_rule.rr_name)

Option	Structure (element)	Permissible values
-ad	Xp_rec_rule.rr_admin	
-ca	Xp_rec_rule.rr_comment	
-fi	Xp_rec_rule.rr_file	
-ho	Xp_rec_rule.rr_host	

Table 71: xpmmod -rec/mod_all_cli() (Xp_rec_rule.rr_name)

xpmmod -spv/mod_all_cli() (Xp_superv.sp_name)

Option	Structure (element)	Permissible values
-ad	Xp_superv.sp_admin	
-ca	Xp_superv.sp_comment	
-ho	Xp_superv.sp_net.nt_host	
-se	Xp_superv.sp_servname	

Table 72: xpmmod -spv/mod_all_cli() (Xp_superv.sp_name)

xpmmod -srv/mod_all_cli() (Xp_server.sv_name)

Option	Structure (element)	Permissible values
-ad	Xp_server.sv_admin	
-ca	Xp_server.sv_comment	
-ho	Xp_server.sv_net.nt_host	
-mj	Xp_server.sv_jbmax	XP_NO_LIMIT
-sp	Xp_server.sv_schedpol	XP_BIG, XP_SMALL, XP_PRIORITY, XP_SECURITY
-up	Xp_server.sv_perm	

Table 73: xpmmod -srv/mod_all_cli() (Xp_server.sv_name)

xpmmod -sys/mod_all_cli()

Option	Structure (element)	Permissible values

-ad	Xp_sys.sy_admin	
-ca	Xp_sys.sy_comment [CM_ADMIN]	
-cu	Xp_sys.sy_comment [CM_USER]	

Table 74: xpmmod -sys/mod_all_cli()

xpmmod -usr/mod_all_cli() (ECM_security.cs_usr_name)

Option	Structure (element)	Permissible values
-aj	ECM_security.sc_jobs_allowed	XP_NO_LIMIT
-ap	ECM_security.sc_pages_allowed	XP_NO_LIMIT
-ca	ECM_security.sc_comment	
-cl	ECM_security.host.lst	
-ho	ECM_security.sc_host	
-nc	ECM_security.sc_max_copies	XP_NO_LIMIT
-pr	ECM_security.sc_max_priority	XP_UNDEFINED
-pv	ECM_security.sc_privileges	ALL_PRIVIL, NONE_PRIVIL, ANY_DEVICE, TOP_PRIVIL
-sl	ECM_security.sc_securit_lvl	TOP_SECRET, SECRET, CONFIDENTIAL, SENSITIVE, UNCLASSIFIED
-ua	ECM_security.sc_usr_account	

Table 75: xpmmod -usr/mod_all_cli() (ECM_security.cs_usr_name)

8.3.6 The xpshow command**xpshow -dev/shw_all_cli() (ECM_printer.dv_name)**

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 76: xpshow -dev/shw_all_cli() (ECM_printer.dv_name)

xpshow -dgr/shw_all_cli() (Xp_dgroup.dg_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 77: xpshow -dgr/shw_all_cli() (Xp_dgroup.dg_name)

xpshow -flt/shw_all_cli() (Xp_filter.fl_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 78: xpshow -flt/shw_all_cli() (Xp_filter.fl_name)

xpshow -fnt/shw_all_cli() (Xp_font.ft_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 79: xpshow -fnt/shw_all_cli() (Xp_font.ft_name)

xpshow -frm/shw_all_cli() (Xp_form.fm_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 80: xpshow -frm/shw_all_cli() (Xp_form.fm_name)

xpshow -gtw/shw_all_cli() (Xp_gateway.gw_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 81: xpshow -gtw/shw_all_cli() (Xp_gateway.gw_name)

xpshow -hos/shw_all_cli() (Xp_host.ho_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY, HIERARCHY

Table 82: xpshow -hos/shw_all_cli() (Xp_host.ho_name)

xpshow -job/shw_job_cli() (Xp_job.jo_id.jid_jid)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 83: xpshow -job/shw_job_cli() (Xp_job.jo_id.jid_jid)

xpshow -jtp/shw_all_cli() (Xp_job_temp.jt_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 84: xpshow -jtp/shw_all_cli() (Xp_job_temp.jt_name)

xpshow -lrs/shw_all_cli() (Xp_soft_load.lr_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	REFERENCED_BY, LONG_DISPLAY, SHORT_DISPLAY

Table 85: xpshow -lrs/shw_all_cli() (Xp_soft_load.lr_name)

xpshow -pcl/shw_all_cli() (Xp_pcl.pl_name)

Option	Structure (element)	Permissible values
--------	---------------------	--------------------

-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY
-----	--------------------------	--------------------------------

Table 86: xpshow -pcl/shw_all_cli() (Xp_pcl.pl_name)

xpshow -prm/shw_all_cli() (Xp_perm.pn_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 87: xpshow -prm/shw_all_cli() (Xp_perm.pn_name)

xpshow -rec/shw_all_cli() (Xp_rec_rule.rr_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 88: xpshow -rec/shw_all_cli() (Xp_rec_rule.rr_name)

xpshow -spv/shw_all_cli() (Xp_superv.sp_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 89: xpshow -spv/shw_all_cli() (Xp_superv.sp_name)

xpshow -srv/shw_all_cli() (Xp_server.sv_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 90: xpshow -srv/shw_all_cli() (Xp_server.sv_name)

xpshow -sys/shw_all_cli()

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 91: xpshow -sys/shw_all_cli()

xpshow -usr/shw_all_cli() (ECM_security.sc_usr_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 92: xpshow -usr/shw_all_cli() (ECM_security.sc_usr_name)

8.3.7 The xpstat command**xpstat -dev/sta_all_cli() (ECM_printer.dv_name)**

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 93: xpstat -dev/sta_all_cli() (ECM_printer.dv_name)

xpstat -dgr/sta_all_cli() (Xp_dgroup.dg_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 94: xpstat -dgr/sta_all_cli() (Xp_dgroup.dg_name)

xpstat -gtw/sta_all_cli() (Xp_gateway.gw_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 95: xpstat -gtw/sta_all_cli() (Xp_gateway.gw_name)

xpstat -hos/sta_all_cli() (Xp_host.ho_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 96: xpstat -hos/sta_all_cli() (Xp_host.ho_name)

xpstat -job/sta_all_cli() (Xp_job.job_id.jid_jid)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 97: xpstat -job/sta_all_cli() (Xp_job.job_id.jid_jid)

xpstat -spv/sta_all_cli() (Xp_superv.sp_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 98: xpstat -spv/sta_all_cli() (Xp_superv.sp_name)

xpstat -srv/sta_all_cli() (Xp_server.sv_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 99: xpstat -srv/sta_all_cli() (Xp_server.sv_name)

xpstat -usr/sta_all_cli() (ECM_security.sc_usr_name)

Option	Structure (element)	Permissible values
-di	shw_all_params.disp_mode	LONG_DISPLAY, SHORT_DISPLAY

Table 100: xpstat -usr/sta_all_cli() (ECM_security.sc_usr_name)

8.3.8 The xpstart -sys and xptest -sys commands**xptest -sys/xptest_cli**

--	--	--

Option	Structure (element)	Permissible values
-fv	xpstart_params.force_version	
-sa	xpstart_params.activation	XP_COLD, XP_WARM

Table 101: xpstart -sys/xpstart_cli

xpstop -sys/xpstop_cli

Option	Structure (element)	Permissible values
-ho	xpstop_params.host_list	
-wn	xpstop_params.when_tbd	WT_DELETE, WT_NOW, WT_END, WT_LAST

Table 102: xpstop -sys/xpstop_cli

8.4 List of messages

Each message is provided in two parts; the message code and the message text. The message code is an identifier in all uppercase characters which is prefixed with "CL_". Identifiers are defined (via *define* statements) as numeric values in the *client.h* header file such that each message identifier is a unique numeric value.



The message text is dependent on the national language used. In this guide the message text is given in English.

8.4.1 Confirmation messages

(CL_OBADD)	XP000001: <%.128s> is being added
(CL_JOADD)	XP000002: Job <%.6s> (<%.128s>) added
(CL_OBCHG)	XP000003: <%.128s> is being changed
(CL_JOCHG)	XP000004: Job <%.6s> (<%.128s>)is being ☐ changed
(CL_OBCPY)	XP000005: <%.128s> is being copied into ☐ <%.128s>
(CL_OBDEL)	XP000006: <%.128s>is deleted
(CL_JODEL)	XP000007: Job <%.6s> (<%.128s>)is being ☐ deleted
(CL_OBMOD)	XP000008: <%.128s> is being modified
(CL_JOMOD)	XP000009: Job <%.6s> (<%.128s>) modified
	XP000011: Copyright (C) Siemens Nixdorf ☐ Software S.A. [1991,1993,1995] ☐ All Rights Reserved
(CL_SETAVAIL)	XP000012: Xprint <%s%s> is being set or is left ☐ ACTIVE
(CL_SETUNAVAIL)	XP000013: Xprint is being set or is left NOT ☐ ACTIVE on <%.128s>
(CL_GWACT)	XP000017: Gateway <%.128s> is being activated

8.4.2 Warning messages

CL_FILWARN)	XP000019: WARNING : file <%.128s> not found □ on host<%.128s>
(CL_WAINFILE)	XP000020: WARNING : system input file cannot □ be executed
(CL_RRWARN)	XP000021: WARNING in recovery rule file
(CL_PLWARN)	XP000022: WARNING in printer control□ language
(CL_WANOUS)	XP000023: WARNING : no user modified or□ deleted
(CL_WAHOUNK)	XP000024: WARNING : host <%.128s> not found □ in (system host file)
(CL_LANWARN)	XP000025: WARNING : device address <%.128s> □ cannot be reached
(CL_WAUSDEF)	XP000026: WARNING : user <%.128s> not found □ in system user file on host □ <%.128s> or cannot reach the host
(CL_WASTSC)	XP000027: WARNING : Spool startup file not □ executed correctly
(CL_GWREFWARN)	XP000028: WARNING : No reference to print □ protocol and network protocol □ <%.128s> found in <%.128s> on host
(CL_WADVNOACT)	XP000029: WARNING : Device <%.128s> cannot □ be activated.The remote server □ <%.128s> is not active.
(CL_WALOGWGA)	XP000030: WARNING : Local gateway(s) with □ same gateway address exist(s).

8.4.3 Error messages

(CL_ERMSG)	XP000039: Error during construction of a □ message
(CL_WRSELECT)	XP000040: Syntax error in selection struc-□ ture
(CL_WRMOD)	XP000041: Syntax error in modification□ structure
(CL_OBNAMOD)	XP000042: Object names cannot be modified
(CL_CNTMNT)	XP000043: Cannot mount form <%.128s> on □ device<%.128s>
(CL_JOACT)	XP000044: Job <%.6s> (<%.128s>) is ACTIVE
(CL_NOJOACT)	XP000045: No job ACTIVE on device <%.128s>
(CL_SPCNTR)	XP000046: Cannot reach supervisor <%.128s>
(CL_SVCNTR)	XP000047: Cannot reach server <%.128s>

(CL_OBSTREF)	XP000048: <%.128s> still referenced by <%d>□ object(s)
(CL_XNTAVA)	XP000049: Xprint not available
(CL_WRJID)	XP000050: <%.128s> : wrong job identifier
(CL_NTASV)	XP000051: <%.128s>'s server not active
(CL_NOSV)	XP000052: No server available
(CL_DBINC)	XP000053: Database inconsistency detected
(CL_NTASP)	XP000054: <%.128s>'s supervisor not active
(CL_XPNTACT)	XP000055: Xprint not active on host <%.128s>
(CL_DVBADPL)	XP000056: <%.128s>'s attributes not sup-□ ported by PCL <%.128s>
(CL_OBSTACT)	XP000057: <%.128s> still active
(CL_NETERR)	XP000058: Network error detected
(CL_ATADM)	XP000059: System administrator reserved command
(CL_RRERR)	XP000060: error(s) in recovery rule file
(CL_PLERR)	XP000061: error(s) in printer control lan-□ guage
(CL_WRVERS)	XP000063: Wrong structure version <%.128s>
(CL_WRPAR)	XP000064: Syntax error in parameter struc-□ ture
(CL_WRADD)	XP000065: Syntax error in addition structure
(CL_DVENSPPO)	XP000066: <%.128s>'s spoolout must be □ disabled
(CL_MANFM)	XP000067: Default forms cannot be MANUAL
(CL_DVACT)	XP000068: Device <%.128s> is being activated
(CL_SPACT)	XP000069: Supervisor <%.128s> is being □ activated
(CL_SVACT)	XP000070: Server <%.128s> is being activated
(CL_DVWRACT)	XP000071: Device <%.128s> cannot be □ activated
(CL_SPWRACT)	XP000072: Supervisor <%.128s> cannot be □ activated
(CL_SVWRACT)	XP000073: Server <%.128s> cannot be □ activated
(CL_CONKEY)	XP000074: Invalid context key <%.128s>
(CL_WRREXP)	XP000075: Syntax error in regular expression□

	<%.128s>
(CL_NOPLRQ)	XP000076: PCL <%.128s> does not fit request
(CL_HOSTAC)	XP000077: Xprint still active on host ☐ <%.128s>
(CL_DLEEMPTY)	XP000078: <%.128s>'s device list would ☐ become empty
(CL_HOEMPTY)	XP000079: <%.128s>'s host list would become ☐ empty
(CL_SUPHOS)	XP000080: Supporting host <%.128s> not ☐ active
(CL_PARHOS)	XP000081: Parasite host <%.128s> still ☐ active
(CL_RRNTAC)	XP000082: Recovery rules <%.128s> not ☐ accessible
(CL_CLNTAC)	XP000083: Class <%.128s> not accessible
(CL_FLNTAC)	XP000084: Filter <%.128s> not accessible
(CL_HONTAC)	XP000085: Host <%.128s> not accessible
(CL_PLNTAC)	XP000086: PCL <%.128s> not accessible
(CL_JTNTAC)	XP000087: Job template <%.128s> not ☐ accessible
(CL_DGNTAC)	XP000088: Device group <%.128s> not ☐ accessible
(CL_OPRNUL)	XP000089: Missing operand : <%.128s>
(CL_OPTINV)	XP000090: Invalid option : <%.128s>
(CL_OBEXIS)	XP000091: <%.128s> already in Xprint ☐ configuration
(CL_OBNTAC)	XP000093: <%.128s> not accessible
(CL_MODFLD)	XP000094: Job <%u> (<%.128s>) not modified ☐ because of filter(s)
(CL_SYNTAC)	XP000096: Form <%.128s> not accessible
(CL_FMNTAC)	XP000097: Font <%.128s> not accessible
(CL_FTNTAC)	XP000098: Access error on file/directory ☐ <%.128s>
(CL_FILACC)	XP000100: Device <%.128s> not accessible
(CL_USNTAC)	XP000101: User <%.128s> not accessible
(CL_PLNTFD)	XP000102: No device found according to ☐ specification

(CL_PLNTFD)	XP000103: Permission <%.128s> not accessible
(CL_TYNTX)	XP000104: Illegal object type
(CL_OPNTX)	XP000105: Unknown command
(CL_OPXPADM)	XP000107: Xprint administrator reserved ☐ command
(CL_OPADM)	XP000108: Administrator reserved command
(CL_STRLEN)	XP000109: Invalid length for value of ☐ <%.128s>
(CL_STRINV)	XP000110: Invalid character in value of ☐ <%.128s>
(CL_INTERR)	XP000111: Internal error
(CL_DENTOK)	XP000112: Destination not accessible
(CL_DESTIN)	XP000113: Invalid environment variable ☐ XP_DESTIN
(CL_SPNTAC)	XP000114: Supervisor <%.128s> not accessible
(CL_SVNTAC)	XP000115: Server <%.128s> not accessible
(CL_JOFULL)	XP000116: Maximum daily jobs quota reached
(CL_PGFULL)	XP000117: Maximum pages quota reached
(CL_CPFULL)	XP000118: Maximum copies quota reached
(CL_FROMTO)	XP000119: Data stream limit conflict ☐ (-fr/-pf or -to/-pt)
(CL_DVEXIS)	XP000120: Device <%.128s> already in Xprint ☐ configuration
(CL_DGEXIS)	XP000121: Device group <%.128s> already in ☐ Xprint configuration
(CL_NOTPWD)	XP000124: User <%.128s> not found in system ☐ user file of host <%.128s>
(CL_MEMLA)	XP000129: Memory lack
(CL_CONERR)	XP000131: Connection error
(CL_DBERR)	XP000132: Database access error
(CL_DVNACT)	XP000133: Device <%.128s> does not enable ☐ spoolin
(CL_PLNTOK)	XP000134: Device <%.128s> does not fit ☐ request
(CL_SPNACT)	XP000136: <%.128s>'s supervisor not active
(CL_SVNTIN)	XP000137: <%.128s>'s server does not enable ☐ spoolin

(CL_SVFULL)	XP000138: <%.128s>'s server has reached its □ maximum job quota
(CL_ICLLIC)	XP000139: Licence violation! Details have □ been sent□to root
(CL_NOFIL)	XP000141: No access to local database files
(CL_EXIST)	XP000142: New object already exists
(CL_ERREF)	XP000143: A referenced object does not exist
(CL_NTFND)	XP000144: Object not found
(CL_NOMAST)	XP000145: No database master present
(CL_REQREF)	XP000146: Request refused by the server
(CL_LOCKF)	XP000147: Xprint lock file already locked
(CL_SEND)	XP000148: Data send error
(CL_RECEIVE)	XP000149: Data receive error
(CL_MANFNT)	XP000150: Invalid option <%.128s> due to □ mandatory default font
(CL_NODEST)	XP000151: No destination could be found
(CL_FLNTFD)	XP000152: Filter <%.128s> not found
(CL_DIRPRT)	XP000153: Direct printing needs explicit □ destination (flag -de mandatory)
(CL_SPOUTON)	XP000154: <%.128s>'s spoolout must be OFF
(CL_FLTSYS)	XP000155: System call failed during filter □ processing
(CL_FLTCON)	XP000156: Connection with SERVER/SUPERVISOR□ failed after filter processing
(CL_FLTARG)	XP000157: Execution of filter <%.128s> □ failed with arguments <%s>
(CL_DBINST)	XP000158: Database installation error
(CL_DBILL)	XP000159: Illegal database startup
(CL_DBLOCH)	XP000160: Local host not defined in database
(CL_DBPORT)	XP000161: Database public port open failed
(CL_DBNOTM)	XP000162: Local host may not be master
(CL_DBCONM)	XP000163: Connection to current master□ failed
(CL_DBSYNC)	XP000164: Resynchronization from master□ failed
(CL_DBXPUS)	XP000165: Search of Xprint user ID failed

(CL_DBABRT)	XP000166: aborted by user
(CL_MSTSTUP)	XP000167: Unable to start up since another □ host is starting up. Retry later.
(CL_DBMCNTR)	XP000168: Cannot reach DB_manager
(CL_DBSTSPC)	XP000169: Filesystem is full
(CL_DBNOSPC)	XP000170: Filesystem is full on master host
(CL_BNDTKG)	XP000186: An object name cannot begin by a □ '!
(CL_ONEGTW)	XP000187: Different gateway names are not □ allowed
(CL_MIXLGT)	XP000188: Mix between local and remote □ object names are not allowed
(CL_WCDGTW)	XP000189: Wild cards are not allowed in □ gateway name
(CL_HOEXDB)	XP000190: Host <%.128s> defined in database
(CL_OPTMAN)	XP000191: The option <%.128s> is mandatory □ in this context
(CL_GWNTAC)	XP000192: Gateway <%.128s> not accessible
(CL_GWBLPL	XP000193: The network protocol of gateway □ <%.128s> and <%.128s> is not the □ same
(CL_GWBLTP)	XP000194: Type of gateway <%.128s> and □ gateway not compatible
(CL_GWINOP)	XP000195: The specific option <%.128s> is □ not correct
(CL_GWWRACT)	XP000196: Gateway <%.128s> cannot be □ activated
(CL_DEVADIR)	XP000197: Device <%.128s> is redirected
(CL_DEVRDIR)	XP000198: A redirection to device <%.128s> □ exists
(CL_DVNTIN)	XP000199: Device <%.128s> must enable □ spoolin
(CL_DVSCTG)	XP000200: Device <%.128s> is redirected to □ itself
(CL_DEVNDIR)	XP000201: Device <%.128s> is not redirected
(CL_ONEOBJ)	XP000202: Only one object (without wildcard) □ may be specified
(CL_NOREF)	XP000203: No reference to specified object

(CL_JOINLO)	XP000204: Job information lost
(CL_FLTEXE	XP000205: Slow filter execution problem
(CL_PAINV)	XP000206: The pair <%s>,<%s> already exist
(CL_GWCNTR)	XP000207: Cannot reach gateway <%s>
(CL_GWAUTH)	XP000208: Access permission denied for □ gateway
(CL_GWNOTSUPP)	XP000209: Requested service not yet supported by gateway
(CL_GWREMGWERR)	XP000210: Unable to process request on □ remote gateway
(CL_GTWRBL)	XP000214: <%s> is referenced in a □ backward list.
(CL_ILLOBN)	XP000215: <%s> : illegal object name.
(CL_PROTOERR)	XP000216: Protocol error
(CL_FBDSTA)	XP000217: Forbidden xpstat on PARTNER □ gateway <%s>
(CL_FBDCHG)	XP000218: Forbidden xpchange on PARTNER □ gateway <%s>
(CL_FBDOPE)	XP000219: Meaningless operation on such type □ of gateway
(CL_DBVERS)	XP000220: Cannot take control from host with □ old version
(CL_GWERR)	XP000221: Unknown error message <%s>
(CL_GWNOKEY)	XP000222: This service needs the Spool-N □ key_disk to be installed
(CL_NOREMSPOOL)	XP000223: Print services not available
(CL_NOLOGGW)	XP000224: No active or accessible local □ gateway to reach remote gateway□ <%s>
(CL_WRDBVERS)	XP000225: No database (installation needed) □ or database with a wrong format □ (migration required).
CL_GWREJREQ	XP000226: Unable to process request on remote gateway:<%s>
CL_OPPEDE	XP000232: Permission denied for object <%s>
CL_OPINPR	XP000233: Insufficient privilege for object <%s>
CL_OPOPDE	XP000234: Operation denied for object <%s>
CL_NODVDEF	XP000235: No device defined In configuration.

CL_NODVACC	XP000236: No device accessible in configuration (criteria: spoolin, capabilities, permission and security level)
CL_DESTNOTDEF	XP000237: Requested destination <%s> not defined in configuration
CL_NOPERMDG	XP000238: Permission denied to device group <%s>.
CL_NODVDGACC	XP000239: No device accessible in device group <%s> (criteria: spoolin, capabilities, permission and security level)
CL_RDDVNOTDEF	XP000240: Redirected device <%s> not defined in configuration
CL_NODVVARACC	XP000241: No device accessible in environment variable XP_DESTIN (criteria: spoolin, capabilities, permission and security level)
CL_JORDIRREMNOT ALL	XP000242: Job redirection to remote destination <%s> not allowed
CL_INVSECLVL	XP000243: Invalid security level to access device.
CL_NOPERMDV	XP000244: Permission denied to print on device <%s>
CL_NOFMPERMDV	XP000245: Permission denied to use the form <%s> on device <%s>
CL_NOLRDV	XP000246: Loadable resource <%.128s> does not allow device <%.128s>
CL_IVLSTAT	XP000247: Current state of job <%s> (<%.128s>) incompatible with request
CL_SVINPR	XP000248: Insufficient privilege for server <%.128s>
CL_SVPEDE	XP000249: Permission denied for server <%.128s>
CL_WROBJID	XP000250: <%.128s>: wrong object identifier
CL_LRECPI	XP000251:; When encapsulation is requested, the printer internal identifier of the loadable resource must be specified
CL_LRCRPI	XP000252: Wenn a call_resource type is requested, the printer internal identifier of the loadable resource must be specified.
CL_JONWNIRDIRNA LL	XP000257: Cannot redirect a job which is not in WAIT or INTERRUPTED state
CL_JONWRDIRREM NOT ALL	XP000258: Remote destination cannot be redirected to another domain.
CL_INVALIDKEY	XP000259: No valid license available

CL_CTFLSPECMPFIXP000260:	-ct or -fl options are forbidden with empty or L non existing files
CL_UNKMSG	XP000262: Unknown message
CL_DVHON64	XP000263: Host of device <%s> does not support large files
CL_SVHON64	XP000264: Server host of device <%s> does not support large files
CL_LCTOOMANYOBJXP000265:	License limit exceeded - too many objects
CL_DEVALRD	XP000266: Device is already redirected
CL_INVALIDFTCONF	XP000267: Invalid FT account on gateway host
CL_INVFILACCADM	XP000268: Access error on file/directory <%.128s>, option <-cp> could solve

8.4.4 PCL/RR compilation messages

The following messages may be issued in response to errors detected in the compilation of PCL (Printer Capability List) or RR (Recovery Rules) statements. Compilation of these statements occurs when a PCL or RR object is added to the database with the *add_all_cli()* function, or when a PCL or RR object is modified by specifying a new statement file. The add and modify functions only allow for one message to be returned in the (*rc_tab*) structure. To enable all problems detected in the statements to be reported, all error messages are concatenated into one message, with the XP000061 as the first line of the message followed by all subsequent 'Err' reports as separate lines. Therefore the message code CL_PLERR applies to the whole (concatenated) message.

(CL_PLERR) XP000061: Comments about analysis of <%s> <%s> file □

Err: <%s> failure - errno = <%d> □

Err: can not open <%s> <%s> file - errno = <%d> □

Err: no entry found in file <%s> for pcl <%s> □

Wng: <%s> capability not supported - line <%d> □

Err: double definition of <%s> capability - line <%d> □

Err: invalid <%s> capability value for <%s> - line <%d> □

Err: <p%c> - too many parameters in parameterized string - line □□

<%d> □

Err: not enough operands for <%c> - line <%d> □

Err: invalid <%s> identifier : <%c%c> - line <%d> □

Err: no closing quote - line <%d> □

Err: no closing double quote - line <%d> □

Err: no closing bracket '}' - line <%d> □

Err: operands of invalid type for <%c> - line <%d> □

Err: comma expected at end of capability - line <%d> □

Err: invalid control character <%c> - line <%d> □

Err: invalid numerical capability <%s%c> - line <%d> □

Err: only one decimal point allowed in scaled value (<%s>) - line <%d> □

Err: no significant value found for <%s> - line <%d> □

Err: decimal point not allowed for <%c> unit - line <%d> □

Err: unknown standard capability <%s> - line <%d> □

Err: invalid 'keep' value field <%c> - line <%d> □

Err: invalid key word <%s> - line <%d> □

Err: invalid printf flag field %#<%c> for capability <%s> - line □

<%d> □
 Err: invalid printf converter field <%c> for capability <%s> - □
 line <%d> □
 Err: no other printf fields allowed with %c for capability <%s> □
 - line <%d> □
 Err: found forbidden character(s) <%c> after <%s> field - line<%d> □
 Err: <%c> expected after <%s> field (skipping until end of □
 line) - line <%d> □
 Wng: <%c> expected after <%s> field - line <%d> □
 Err: <%s> : multi definition - line <%d> □
 Err: undefined generated code <%s> - line <%d> □
 Err: generated code makes only sense if time field is specified □
 - line <%d> □
 Wng: <%s> not defined (can only be used by private backends) - □
 line <%d> □
 Err: possible loop starting with generated code <%s> - line□
 <%d> □
 Err: <%s> field is mandatory - line <%d> □
 Err: no recovery rules found

Related publications

Ordering manuals

Please apply to your local office for ordering the manuals.

- [1] **Xprint V6.0
User and Administrator Guide**
Target Group
Users and administrators of Xprint
Contents
The manual describes the methods used for working with Xprint and for its administration and installation.
- [2] **Xprint V6.0
Reference Manual**
Target Group
Users and Administrators of Xprint.
Contents
This manual serves as a reference book containing all information required for using and administering Xprint. All Xprint commands and files are described.
- [3] **Xprint V6.0
Graphical User Interface**
Target Group
Xprint administrators using Xprint management functions.
Contents
The manual describes the Graphical User Interface (GUI) for Xprint. It allows you to monitor and configure all the hosts, printers and jobs in the domain. It also allows you to see which forms and fonts exist in the domain.