Rocky Enterprise Linux 9.2 Manual Pages on command 'xsm.1'

$ man xsm.1

XSM(1)   General Commands Manual   XSM(1)

NAME

   xsm - X Session Manager

SYNOPSIS

   xsm [-display display] [-session sessionName] [-verbose]

DESCRIPTION

   xsm is a session manager. A session is a group of applications, each of which has a par?

   ticular state. xsm allows you to create arbitrary sessions - for example, you might have

   a "light" session, a "development" session, or an "xterminal" session. Each session can

   have its own set of applications. Within a session, you can perform a "checkpoint" to

   save application state, or a "shutdown" to save state and exit the session. When you log

   back in to the system, you can load a specific session, and you can delete sessions you no

   longer want to keep.

   Some session managers simply allow you to manually specify a list of applications to be

   started in a session. xsm is more powerful because it lets you run applications and have

   them automatically become part of the session. On a simple level, xsm is useful because

   it gives you this ability to easily define which applications are in a session. The true

   power of xsm, however, can be taken advantage of when more and more applications learn to

   save and restore their state.

OPTIONS

   -display display

        Causes xsm to connect to the specified X display.

   -session sessionName

Causes xsm to load the specified session, bypassing the session menu.

-verbose

Turns on debugging information.

SETUP

.xsession file

Using xsm requires a change to your .xsession file:

The last program executed by your .xsession file should be xsm. With this configuration,

when the user chooses to shut down the session using xsm, the session will truly be over.

Since the goal of the session manager is to restart clients when logging into a session,

your .xsession file, in general, should not directly start up applications.   Rather, the

applications should be started within a session. When xsm shuts down the session, xsm

will know to restart these applications. Note however that there are some types of appli?

cations that are not "session aware". xsm allows you to manually add these applications

to your session (see the section titled Client List).

SM_SAVE_DIR environment variable

If the SM_SAVE_DIR environment variable is defined, xsm will save all configuration files

in this directory. Otherwise, they will be stored in the user's home directory. Session

aware applications are also encouraged to save their checkpoint files in the SM_SAVE_DIR

directory, although the user should not depend on this convention.

Default Startup Applications

The first time xsm is started, it will need to locate a list of applications to start up.

For example, this list might include a window manager, a session management proxy, and an

xterm.   xsm will first look for the file .xsmstartup in the user's home directory. If

that file does not exist, it will look for the system.xsm file that was set up at instal?

lation time. Note that xsm provides a "fail safe" option when the user chooses a session

to start up. The fail safe option simply loads the default applications described above.

Each line in the startup file should contain a command to start an application. A sample

startup file might look this:

<start of file>

twm

smproxy

xterm

STARTING A SESSION

   When xsm starts up, it first checks to see if the user previously saved any sessions. If
   no saved sessions exist, xsm starts up a set of default applications (as described above
   in the section titled Default Startup Applications). If at least one session exists, a
   session menu is presented. The -session option forces the specified sessionName session
   to be loaded, bypassing the session menu.

 The session menu

   The session menu presents the user with a list of sessions to choose from. The user can
   change the currently selected session with the mouse, or by using the up and down arrows
   on the keyboard.   Note that sessions which are locked (i.e. running on a different dis?
   play) can not be loaded or deleted.

   The following operations can be performed from the session menu:

   Load Session   Pressing this button will load the currently selected session.   Al?

                       ternatively, hitting the Return key will also load the currently se?

                       lected session, or the user can double click a session from the

                       list.

   Delete Session   This operation will delete the currently selected session, along

                       with all of the application checkpoint files associated with the

                       session. After pressing this button, the user will be asked to

                       press the button a second time in order to confirm the operation.

   Default/Fail Safe   xsm will start up a set of default applications (as described above

                       in the section titled Default Startup Applications). This is useful

                       when the user wants to start a fresh session, or if the session con?

                       figuration files were corrupted and the user wants a "fail safe"

                       session.

   Cancel   Pressing this button will cause xsm to exit. It can also be used to

                       cancel a "Delete Session" operation.

CONTROLLING A SESSION

   After xsm determines which session to load, it brings up its main window, then starts up
   all applications that are part of the session. The title bar for the session manager's
   main window will contain the name of the session that was loaded.

   The following options are available from xsm's main window:

   Client List   Pressing this button brings up a window containing a list of all clients

that are in the current session. For each client, the host machine that
the client is running on is presented. As clients are added and removed
from the session, this list is updated to reflect the changes. The user
is able to control how these clients are restarted (see below).

By pressing the View Properties button, the user can view the session
management properties associated with the currently selected client.

By pressing the Clone button, the user can start a copy of the selected
application.

By pressing the Kill Client button, the user can remove a client from
the session.

By selecting a restart hint from the Restart Hint menu, the user can
control the restarting of a client. The following hints are available:

- The Restart If Running hint indicates that the client should be
restarted in the next session if it is connected to the session manager
at the end of the current session.

- The Restart Anyway hint indicates that the client should be restarted
in the next session even if it exits before the current session is ter?
minated.

- The Restart Immediately hint is similar to the Restart Anyway hint,
but in addition, the client is meant to run continuously. If the client
exits, the session manager will try to restart it in the current ses?
sion.

- The Restart Never hint indicates that the client should not be
restarted in the next session.

Note that all X applications may not be "session aware". Applications
that are not session aware are ones that do not support the X Session
Management Protocol or they can not be detected by the Session Manage?
ment Proxy (see the section titled THE PROXY). xsm allows the user to
manually add such applications to the session. The bottom of the Client
List window contains a text entry field in which application commands
can be typed in. Each command should go on its own line. This informa?
tion will be saved with the session at checkpoint or shutdown time.
When the session is restarted, xsm will restart these applications in

addition to the regular "session aware" applications.

Pressing the Done button removes the Client List window.

Session Log...   The Session Log window presents useful information about the session.
For example, when a session is restarted, all of the restart commands
will be displayed in the log window.

Checkpoint   By performing a checkpoint, all applications that are in the session are
asked to save their state. Not every application will save its complete
state, but at a minimum, the session manager is guaranteed that it will
receive the command required to restart the application (along with all
command line options). A window manager participating in the session
should guarantee that the applications will come back up with the same
window configurations.

If the session being checkpointed was never assigned a name, the user
will be required to specify a session name. Otherwise, the user can
perform the checkpoint using the current session name, or a new session
name can be specified. If the session name specified already exists,
the user will be given the opportunity to specify a different name or to
overwrite the already existing session. Note that a session which is
locked can not be overwritten.

When performing a checkpoint, the user must specify a Save Type which
informs the applications in the session how much state they should save.

The Local type indicates that the application should save enough infor?
mation to restore the state as seen by the user. It should not affect
the state as seen by other users. For example, an editor would create a
temporary file containing the contents of its editing buffer, the loca?
tion of the cursor, etc...

The Global type indicates that the application should commit all of its
data to permanent, globally accessible storage. For example, the editor
would simply save the edited file.

The Both type indicates that the application should do both of these.
For example, the editor would save the edited file, then create a tempo?
rary file with information such as the location of the cursor, etc...

In addition to the Save Type, the user must specify an Interact Style.

The None type indicates that the application should not interact with the user while saving state.

The Errors type indicates that the application may interact with the user only if an error condition arises.

The Any type indicates that the application may interact with the user for any purpose. Note that xsm will only allow one application to in‐teract with the user at a time.

After the checkpoint is completed, xsm will, if necessary, display a window containing the list of applications which did not report a suc‐cessful save of state.

Shutdown   A shutdown provides all of the options found in a checkpoint, but in ad‐dition, can cause the session to exit. Note that if the interaction style is Errors or Any, the user may cancel the shutdown. The user may also cancel the shutdown if any of the applications report an unsuccess‐ful save of state.

The user may choose to shutdown the session with our without performing a checkpoint.

HOW XSM RESPONDS TO SIGNALS

xsm will respond to a SIGTERM signal by performing a shutdown with the following options: fast, no interaction, save type local. This allows the user's session to be saved when the system is being shutdown. It can also be used to perform a remote shutdown of a ses‐sion.

xsm will respond to a SIGUSR1 signal by performing a checkpoint with the following op‐tions: no interaction, save type local. This signal can be used to perform a remote checkpoint of a session.

THE PROXY

Since not all applications have been ported to support the X Session Management Protocol, a proxy service exists to allow "old" clients to work with the session manager. In order for the proxy to detect an application joining a session, one of the following must be true:

- The application maps a top level window containing the WM_CLIENT_LEADER property. This property provides a pointer to the client leader window which contains the WM_CLASS, WM_NAME, WM_COMMAND, and WM_CLIENT_MACHINE properties.

or ...

- The application maps a top level window which does not contain the WM_CLIENT_LEADER
property. However, this top level window contains the WM_CLASS, WM_NAME, WM_COMMAND, and
WM_CLIENT_MACHINE properties.

An application that support the WM_SAVE_YOURSELF protocol will receive a WM_SAVE_YOURSELF
client message each time the session manager issues a checkpoint or shutdown. This allows
the application to save state. If an application does not support the WM_SAVE_YOURSELF
protocol, then the proxy will provide enough information to the session manager to restart
the application (using WM_COMMAND), but no state will be restored.

REMOTE APPLICATIONS

xsm requires a remote execution protocol in order to restart applications on remote ma‐
chines. Currently, xsm supports the rstart protocol. In order to restart an application
on remote machine X, machine X must have rstart installed. In the future, additional re‐
mote execution protocols may be supported.

SEE ALSO

smproxy(1), rstart(1)

AUTHORS

Ralph Mor, X Consortium

Jordan Brown, Quarterdeck Office Systems

X Version 11   xsm 1.0.4   XSM(1)