

```

*****
31079 Wed Jul 1 08:55:04 2009
new/src/gui/modules/beadmin.py
*** NO COMMENTS ***
*****
1 #!/usr/bin/python2.4
2 #
3 # CDDL HEADER START
4 #
5 # The contents of this file are subject to the terms of the
6 # Common Development and Distribution License (the "License").
7 # You may not use this file except in compliance with the License.
8 #
9 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 # or http://www.opensolaris.org/os/licensing.
11 # See the License for the specific language governing permissions
12 # and limitations under the License.
13 #
14 # When distributing Covered Code, include this CDDL HEADER in each
15 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 # If applicable, add the following below this CDDL HEADER, with the
17 # fields enclosed by brackets "[]" replaced with your own identifying
18 # information: Portions Copyright [yyyy] [name of copyright owner]
19 #
20 # CDDL HEADER END
21 #
22 # Copyright 2009 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #

26 import sys
27 import os
28 import pango
29 import time
30 import datetime
31 import locale
32 import pkg.pkgsubprocess as subprocess
33 from threading import Thread

35 try:
36     import gobject
37     gobject.threads_init()
38     import gnome
39     import gtk
40     import gtk.glade
41     import pygtk
42     pygtk.require("2.0")
43 except ImportError:
44     sys.exit(1)
45 import pkg.gui.misc as gui_misc

47 nobe = False

49 try:
50     import libbe as be
51 except ImportError:
52     # All actions are disabled when libbe can't be imported.
53     nobe = True
54 import pkg.misc

56 #BE_LIST
57 (
58 BE_ID,
59 BE_MARKED,
60 BE_NAME,
61 BE_ORIG_NAME,

```

```

62 BE_DATE_TIME,
63 BE_CURRENT_PIXBUF,
64 BE_ACTIVE_DEFAULT,
65 BE_SIZE,
66 BE_EDITABLE
67 ) = range(9)

69 class Beadmin:
70     def __init__(self, parent):
71         self.parent = parent

73         if nobe:
74             msg = _("The <b>libbe</b> library was not "
75                   "found on your system."
76                   "\nAll functions for managing Boot Environments are
77                   msgbox = gtk.MessageDialog(
78                       buttons = gtk.BUTTONS_CLOSE,
79                       flags = gtk.DIALOG_MODAL, type = gtk.MESSAGE_INFO,
80                       message_format = None)
81                   msgbox.set_markup(msg)
82                   msgbox.set_title(_("BE management"))
83                   msgbox.run()
84                   msgbox.destroy()
85                   return

87         self.be_list = \
88             gtk.ListStore(
89                 gobject.TYPE_INT,           # BE_ID
90                 gobject.TYPE_BOOLEAN,      # BE_MARKED
91                 gobject.TYPE_STRING,       # BE_NAME
92                 gobject.TYPE_STRING,       # BE_ORIG_NAME
93                 gobject.TYPE_STRING,       # BE_DATE_TIME
94                 gtk.gdk.Pixbuf,           # BE_CURRENT_PIXBUF
95                 gobject.TYPE_BOOLEAN,      # BE_ACTIVE_DEFAULT
96                 gobject.TYPE_STRING,       # BE_SIZE
97                 gobject.TYPE_BOOLEAN,      # BE_EDITABLE
98             )

99         self.progress_stop_thread = False
100         self.initial_active = 0
101         self.initial_default = 0
102         gladefile = os.path.join(self.parent.application_dir,
103                                   "usr/share/package-manager/packagemanager.glade")
104         w_tree_beadmin = gtk.glade.XML(gladefile, "beadmin")
105         w_tree_progress = gtk.glade.XML(gladefile, "progressdialog")
106         w_tree_beconfirmation = gtk.glade.XML(gladefile,
107                                               "beconfirmationdialog")
108         self.w_beadmin_dialog = w_tree_beadmin.get_widget("beadmin")
109         self.w_be_treeview = w_tree_beadmin.get_widget("betreeview")
110         self.w_cancel_button = w_tree_beadmin.get_widget("cancelbebutton")
111         self.w_ok_button = w_tree_beadmin.get_widget("okbebutton")
112         w_active_gtkimage = w_tree_beadmin.get_widget("activebeimage")
113         self.w_progress_dialog = w_tree_progress.get_widget("progressdia
114         self.w_progress_dialog.connect('delete-event', lambda stub1, stu
115         self.w_progressinfo_label = w_tree_progress.get_widget("progress
116         progress_button = w_tree_progress.get_widget("progresscancel")
117         self.w_progressbar = w_tree_progress.get_widget("progressbar")
118         self.w_beconfirmation_dialog = \
119             w_tree_beconfirmation.get_widget("beconfirmationdialog")
120         self.w_beconfirmation_textview = \
121             w_tree_beconfirmation.get_widget("beconfirmtext")
122         self.w_cancelbe_button = w_tree_beconfirmation.get_widget("cance
123         self.w_ok_button.set_sensitive(False)
124         progress_button.hide()

```

```

125     self.w_progressbar.set_pulse_step(0.1)
126     self.list_filter = self.be_list.filter_new()
127     self.w_be_treeview.set_model(self.list_filter)
128     self.__init_tree_views()
129     self.active_image = gui_misc.get_icon(
130         self.parent.icon_theme, "status_checkmark")
131     w_active_gtkimage.set_from_pixbuf(self.active_image)

133     bebuffer = self.w_beconfirmation_textview.get_buffer()
134     bebuffer.create_tag("bold", weight=pango.WEIGHT_BOLD)

136     try:
137         dic = \
138         {
139             "on_cancel_be_clicked": \
140                 self.__on_cancel_be_clicked,
141             "on_ok_be_clicked": \
142                 self.__on_ok_be_clicked,
143             "on_help_bebutton_clicked": \
144                 self.__on_help_bebutton_clicked,
145         }
146         dic_conf = \
147         {
148             "on_cancel_be_conf_clicked": \
149                 self.__on_cancel_be_conf_clicked,
150             "on_ok_be_conf_clicked": \
151                 self.__on_ok_be_conf_clicked,
152             "on_beconfirmationdialog_delete_event": \
153                 self.__on_beconfirmationdialog_delete_event,
154         }
155         w_tree_beadmin.signal_autoconnect(dic)
156         w_tree_beconfirmation.signal_autoconnect(dic_conf)
157     except AttributeError, error:
158         print _("GUI will not respond to any event! %s. "
159               "%Check beadmin.py signals") \
160               % error
161     Thread(target = self.__progress_pulse).start()
162     Thread(target = self.__prepare_beadmin_list).start()
163     sel = self.w_be_treeview.get_selection()
164     self.w_cancel_button.grab_focus()
165     sel.set_mode(gtk.SELECTION_SINGLE)
166     self.w_beadmin_dialog.show_all()
167     self.w_progress_dialog.set_title(
168         _("Loading Boot Environment Information"))
169     self.w_progressinfo_label.set_text(
170         _("Fetching BE entries..."))
171     self.w_progress_dialog.show()

173     def __progress_pulse(self):
174         while not self.progress_stop_thread:
175             GObject.idle_add(self.w_progressbar.pulse)
176             time.sleep(0.1)
177         GObject.idle_add(self.w_progress_dialog.hide)

179     def __prepare_beadmin_list(self):
180         be_list = be.beList()
181         GObject.idle_add(self.__create_view_with_be, be_list)
182         self.progress_stop_thread = True
183         return

185     def __init_tree_views(self):
186         model = self.w_be_treeview.get_model()

188         column = gtk.TreeViewColumn()
189         column.set_title("")
190         render_pixbuf = gtk.CellRendererPixbuf()

```

```

191         column.pack_start(render_pixbuf, expand = True)
192         column.add_attribute(render_pixbuf, "pixbuf", BE_CURRENT_PIXBUF)
193         self.w_be_treeview.append_column(column)

195     name_renderer = gtk.CellRendererText()
196     name_renderer.connect('edited', self.__be_name_edited, model)
197     column = gtk.TreeViewColumn(_("Boot Environment"),
198         name_renderer, text = BE_NAME)
199     column.set_cell_data_func(name_renderer, self.__cell_data_func)
200     column.set_expand(True)
201     if "beVerifyBEName" in be.__dict__:
202         column.add_attribute(name_renderer, "editable",
203             BE_EDITABLE)
204     self.w_be_treeview.append_column(column)

206     datetime_renderer = gtk.CellRendererText()
207     datetime_renderer.set_property('xalign', 0.0)
208     column = gtk.TreeViewColumn(_("Created"), datetime_renderer,
209         text = BE_DATE_TIME)
210     column.set_cell_data_func(datetime_renderer,
211         self.__cell_data_function, None)
212     column.set_expand(True)
213     self.w_be_treeview.append_column(column)

215     size_renderer = gtk.CellRendererText()
216     size_renderer.set_property('xalign', 1.0)
217     column = gtk.TreeViewColumn(_("Size"), size_renderer,
218         text = BE_SIZE)
219     column.set_cell_data_func(size_renderer, self.__cell_data_func)
220     column.set_expand(False)
221     self.w_be_treeview.append_column(column)

223     radio_renderer = gtk.CellRendererToggle()
224     radio_renderer.connect('toggled', self.__active_pane_default, mo)
225     column = gtk.TreeViewColumn(_("Active on Reboot"),
226         radio_renderer, active = BE_ACTIVE_DEFAULT)
227     radio_renderer.set_property("activatable", True)
228     radio_renderer.set_property("radio", True)
229     column.set_cell_data_func(radio_renderer,
230         self.__cell_data_default_function, None)
231     column.set_expand(False)
232     self.w_be_treeview.append_column(column)

234     toggle_renderer = gtk.CellRendererToggle()
235     toggle_renderer.connect('toggled', self.__active_pane_toggle, mo)
236     column = gtk.TreeViewColumn(_("Delete"), toggle_renderer,
237         active = BE_MARKED)
238     toggle_renderer.set_property("activatable", True)
239     column.set_cell_data_func(toggle_renderer,
240         self.__cell_data_delete_function, None)
241     column.set_expand(False)
242     self.w_be_treeview.append_column(column)

244     def __on_help_bebutton_clicked(self, widget):
245         if self.parent != None:
246             gui_misc.display_help(self.parent.application_dir, "mana")
247         else:
248             gui_misc.display_help()

249     def __on_ok_be_clicked(self, widget):
250         self.w_progress_dialog.set_title(_("Applying changes"))
251         self.w_progressinfo_label.set_text(
252             _("Applying changes, please wait ..."))
253         if self.w_ok_button.get_property('sensitive') == 0:
254             self.progress_stop_thread = True
255             self.__on_beadmin_delete_event(None, None)
256

```

```

257         return
258         Thread(target = self.__activate).start()
259
260 def __on_cancel_be_clicked(self, widget):
261     self.__on_beadmin_delete_event(None, None)
262     return False
263
264 def __on_beconfirmationdialog_delete_event(self, widget, event):
265     self.__on_cancel_be_conf_clicked(widget)
266     return True
267
268 def __on_cancel_be_conf_clicked(self, widget):
269     self.w_beconfirmation_dialog.hide()
270
271 def __on_ok_be_conf_clicked(self, widget):
272     self.w_beconfirmation_dialog.hide()
273     self.progress_stop_thread = False
274     Thread(target = self.__on_progressdialog_progress).start()
275     Thread(target = self.__delete_activate_be).start()
276
277 def __on_beadmin_delete_event(self, widget, event, stub=None):
278     self.w_beadmin_dialog.destroy()
279     return True
280
281 def __activate(self):
282     active_text = _("Active on reboot:\n")
283     delete_text = _("Delete boot environments:\n")
284     rename_text = _("Rename boot environments:\n")
285     active = ""
286     delete = {}
287     rename = {}
288     for row in self.be_list:
289
290         if row[BE_MARKED]:
291             delete += row[BE_NAME] + "\n"
292         if row[BE_ACTIVE_DEFAULT] == True and row[BE_ID] != \
293             self.initial_default:
294             active += row[BE_NAME] + "\n"
295         if row[BE_NAME] != row[BE_ORIG_NAME]:
296             rename[row[BE_ORIG_NAME]] = row[BE_NAME]
297     textbuf = self.w_beconfirmation_textview.get_buffer()
298     textbuf.set_text("")
299     textiter = textbuf.get_end_iter()
300     if len(active) > 0:
301         textbuf.insert_with_tags_by_name(textiter,
302             active_text, "bold")
303         textbuf.insert_with_tags_by_name(textiter,
304             active)
305     if len(delete) > 0:
306         if len(active) > 0:
307             textbuf.insert_with_tags_by_name(textiter,
308                 "\n")
309         textbuf.insert_with_tags_by_name(textiter,
310             delete_text, "bold")
311         textbuf.insert_with_tags_by_name(textiter,
312             delete)
313     if len(rename) > 0:
314         if len(delete) > 0 or len(active) > 0:
315             textbuf.insert_with_tags_by_name(textiter,
316                 "\n")
317         textbuf.insert_with_tags_by_name(textiter,
318             rename_text, "bold")
319     for orig in rename:
320         textbuf.insert_with_tags_by_name(textiter,
321             orig)
322         textbuf.insert_with_tags_by_name(textiter,

```

```

323         _(" to "), "bold")
324         textbuf.insert_with_tags_by_name(textiter,
325             rename.get(orig) + "\n")
326     self.w_cancelbe_button.grab_focus()
327     GObject.idle_add(self.w_beconfirmation_dialog.show)
328     self.progress_stop_thread = True
329
330 def __on_progressdialog_progress(self):
331     # This needs to be run in GObject.idle_add, otherwise we will ge
332     # Xlib: unexpected async reply (sequence 0x2db0)!
333     GObject.idle_add(self.w_progress_dialog.show)
334     while not self.progress_stop_thread:
335         GObject.idle_add(self.w_progressbar.pulse)
336         time.sleep(0.1)
337     GObject.idle_add(self.w_progress_dialog.hide)
338
339 def __delete_activate_be(self):
340     not_deleted = []
341     not_default = None
342     not_renamed = {}
343     # The while gtk.events_pending():
344     #     gtk.main_iteration(False)
345     # Is not working if we are calling libbe, so it is required
346     # To have sleep in few places in this function
347     # Remove
348     for row in self.be_list:
349         if row[BE_MARKED]:
350             time.sleep(0.1)
351             result = self.__destroy_be(row[BE_NAME])
352             if result != 0:
353                 not_deleted.append(row[BE_NAME])
354
355     # Rename
356     for row in self.be_list:
357         if row[BE_NAME] != row[BE_ORIG_NAME]:
358             time.sleep(0.1)
359             result = self.__rename_be(row[BE_ORIG_NAME],
360                 row[BE_NAME])
361             if result != 0:
362                 not_renamed[row[BE_ORIG_NAME]] = row[BE_
363
364     # Set active
365     for row in self.be_list:
366         if row[BE_ACTIVE_DEFAULT] == True and row[BE_ID] != \
367             self.initial_default:
368             time.sleep(0.1)
369             result = self.__set_default_be(row[BE_NAME])
370             if result != 0:
371                 not_default = row[BE_NAME]
372
373     if len(not_deleted) == 0 and not_default == None \
374         and len(not_renamed) == 0:
375         self.progress_stop_thread = True
376     else:
377         self.progress_stop_thread = True
378         msg = ""
379         if not_default:
380             msg += _("<b>Couldn't change Active "
381                 "Boot Environment to:</b>\n") + not_default
382         if len(not_deleted) > 0:
383             if not_default:
384                 msg += "\n\n"
385             msg += _("<b>Couldn't delete Boot "
386                 "Environments:</b>\n")
387             for row in not_deleted:
388                 msg += row + "\n"
389         if len(not_renamed) > 0:
390             if not_default or len(not_deleted):
391                 msg += "\n"

```



```

521         unicode(
522             _("%m/%d/%y %H:%M"),
523             "utf-8").encode(
524             locale.getpreference
525         except (UnicodeError, LookupError,
526             locale.Error):
527             date_format = "%F %H:%M"
528             date_time = \
529                 date_tmp2.strftime(date_form
530                 i += 1
531         except (NameError, ValueError, TypeError):
532             date_time = None
533     else:
534         date_tmp = time.localtime(be_date)
535         try:
536             date_format = \
537                 unicode(
538                     _("%m/%d/%y %H:%M"),
539                     "utf-8").encode(
540                     locale.getpreference
541         except (UnicodeError, LookupError, local
542             date_format = "%F %H:%M"
543             date_time = \
544                 time.strftime(date_format, date_tmp)
545         if active:
546             active_img = self.active_image
547             self.initial_active = j
548         if active_boot:
549             self.initial_default = j
550         if date_time != None:
551             try:
552                 date_time = unicode(date_time,
553                     locale.getpreference
554                     "utf-8")
555             except (UnicodeError, LookupError, local
556                 pass
557             self.be_list.insert(j, [j, False,
558                 name, name,
559                 date_time, active_img,
560                 active_boot, converted_size, active_img == N
561                 j += 1
562             self.w_be_treeview.set_cursor(self.initial_active, None,
563                 start_editing=True)
564             self.w_be_treeview.scroll_to_cell(self.initial_active)
565
566     def __destroy_be(self, be_name):
567         return be.beDestroy(be_name, 1, True)
568
569     def __set_default_be(self, be_name):
570         return be.beActivate(be_name)
571
572     def __cell_data_default_function(self, column, renderer, model, itr, dat
573         if itr:
574             if model.get_value(itr, BE_MARKED):
575                 self.__set_renderer_active(renderer, False)
576             else:
577                 self.__set_renderer_active(renderer, True)
578
579     def __cell_data_delete_function(self, column, renderer, model, itr, data
580         if itr:
581             if model.get_value(itr, BE_ACTIVE_DEFAULT) or \
582                 (self.initial_active == model.get_value(itr, BE_ID))
583                 (model.get_value(itr, BE_NAME) !=
584                 model.get_value(itr, BE_ORIG_NAME)):
585                 self.__set_renderer_active(renderer, False)
586         else:

```

```

587         self.__set_renderer_active(renderer, True)
588
589     @staticmethod
590     def __set_renderer_active(renderer, active):
591         if active:
592             renderer.set_property("sensitive", True)
593             renderer.set_property("mode", gtk.CELL_RENDERER_MODE_ACT
594         else:
595             renderer.set_property("sensitive", False)
596             renderer.set_property("mode", gtk.CELL_RENDERER_MODE_INE
597
598     @staticmethod
599     def __get_dates_of_creation(be_list):
600         #zfs list -H -o creation rpool/ROOT/opensolaris-1
601         cmd = [ "/sbin/zfs", "list", "-H", "-o", "creation" ]
602         for bee in be_list:
603             if bee.get("orig_be_name"):
604                 name = bee.get("orig_be_name")
605                 pool = bee.get("orig_be_pool")
606                 cmd += [pool+"/ROOT/"+name]
607         if len(cmd) <= 5:
608             return None
609         list_of_dates = []
610         try:
611             proc = subprocess.Popen(cmd, stdout = subprocess.PIPE,
612                 stderr = subprocess.PIPE,)
613             line_out = proc.stdout.readline()
614             while line_out:
615                 list_of_dates.append(line_out)
616                 line_out = proc.stdout.readline()
617         except OSError:
618             return list_of_dates
619         return list_of_dates
620
621     @staticmethod
622     def __convert_size_of_be_to_string(be_size):
623         if not be_size:
624             be_size = 0
625         return pkg.misc.bytes_to_str(be_size)
626
627     @staticmethod
628     def __cell_data_function(column, renderer, model, itr, data):
629         if itr:
630             if model.get_value(itr, BE_CURRENT_PIXBUF):
631                 renderer.set_property("weight", pango.WEIGHT_BOLD)
632             else:
633                 renderer.set_property("weight", pango.WEIGHT_NORMAL)

```

```

*****
49185 Wed Jul 1 08:55:06 2009
new/src/gui/modules/installupdate.py
*** NO COMMENTS ***
*****
1 #!/usr/bin/python2.4
2 #
3 # CDDL HEADER START
4 #
5 # The contents of this file are subject to the terms of the
6 # Common Development and Distribution License (the "License").
7 # You may not use this file except in compliance with the License.
8 #
9 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 # or http://www.opensolaris.org/os/licensing.
11 # See the License for the specific language governing permissions
12 # and limitations under the License.
13 #
14 # When distributing Covered Code, include this CDDL HEADER in each
15 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 # If applicable, add the following below this CDDL HEADER, with the
17 # fields enclosed by brackets "[]" replaced with your own identifying
18 # information: Portions Copyright [yyyy] [name of copyright owner]
19 #
20 # CDDL HEADER END
21 #
22 # Copyright 2009 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #

26 MIN_IND_ELEMENTS_BOUNCE = 5      # During indexing the progress will be progress
27                                 # the number of indexing elements is greater th
28                                 # otherwise it will bounce

30 import errno
31 import sys
32 import time
33 import pango
34 import datetime
35 import traceback
36 import string
37 from threading import Thread
38 from urllib2 import URLError
39 try:
40     import gobject
41     import gtk
42     import gtk.glade
43     import pygtk
44     pygtk.require("2.0")
45 except ImportError:
46     sys.exit(1)
47 nobe = False
48 try:
49     import libbe as be
50 except ImportError:
51     nobe = True
52 import pkg.client.progress as progress
53 import pkg.misc
54 from pkg.client.retrieve import ManifestRetrievalError
55 from pkg.client.retrieve import DatastreamRetrievalError
56 from pkg.client.filelist import FileListRetrievalError
57 import pkg.client.api_errors as api_errors
58 from pkg.misc import TransferTimedOutException, TransportException
59 import pkg.gui.beadm as beadm
60 import pkg.gui.misc as gui_misc
61 import pkg.gui.enumerations as enumerations

```

```

63 ERROR_FORMAT = "<span color = \"red\">%s</span>"

66 class InstallUpdate(progress.ProgressTracker):
67     def __init__(self, list_of_packages, parent, api_o,
68                 ips_update = False, action = -1, be_name = None,
69                 parent_name = "", pkg_list = None, main_window = None,
70                 icon_confirm_dialog = None, title = None, web_install = False):
71         if action == -1:
72             return
73         progress.ProgressTracker.__init__(self)
74         self.web_install = web_install
75         self.web_updates_list = None
76         self.api_o.progressstracker = self
77         self.api_o = api_o
78         self.parent = parent
79         self.be_list = None
80         self.be_name = be_name
81         self.parent_name = parent_name
82         self.ipkg_ipkgui_list = pkg_list
83         self.icon_confirm_dialog = icon_confirm_dialog
84         self.title = title
85         self.w_main_window = main_window
86         self.ips_update = ips_update
87         self.list_of_packages = list_of_packages
88         self.act_phase_last = None
89         self.action = action
90         self.canceling = False
91         self.current_stage_name = None
92         self.ind_started = None
93         self.ip = None
94         self.operations_done = False
95         self.prev_ind_phase = None
96         self.prev_pkg = None
97         self.progress_stop_timer_running = False
98         self.proposed_be_name = None
99         self.stages = {
100             1:[_("Preparing..."), _("Preparation")],
101             2:[_("Downloading..."), _("Download")],
102             3:[_("Installing..."), _("Install")],
103         }
104         self.stop_bouncing_progress = False
105         self.stopped_bouncing_progress = True
106         self.update_list = {}
107         gladefile = os.path.join(self.parent.application_dir,
108                                 "usr/share/package-manager/packagemanager.glade")
109         gladefile = self.parent.application_dir + \
110             "/usr/share/package-manager/packagemanager.glade"
111         w_tree_dialog = gtk.glade.XML(gladefile, "createplandialog")
112         w_tree_uconfirm = gtk.glade.XML(gladefile, "ua_confirm_dialog")
113         w_tree_removeconfirm = \
114             gtk.glade.XML(gladefile, "removeconfirmation")
115         self.w_dialog = w_tree_dialog.get_widget("createplandialog")
116         self.w_expander = w_tree_dialog.get_widget("expander3")
117         self.w_cancel_button = w_tree_dialog.get_widget("cancelcreatepla")
118         self.w_progressbar = w_tree_dialog.get_widget("createplanprogres")
119         self.w_details_textview = w_tree_dialog.get_widget("createplante")
120         self.w_removeconfirm_dialog = \
121             w_tree_removeconfirm.get_widget("removeconfirmation")
122         w_removeproceed_button = w_tree_removeconfirm.get_widget("remove")
123         self.w_remove_treeview = w_tree_removeconfirm.get_widget("removetreev")
124         w_stage2 = w_tree_dialog.get_widget("stage2")
125         self.w_stages_box = w_tree_dialog.get_widget("stages_box")
126         self.w_stages_label = w_tree_dialog.get_widget("label_stage1")
127         self.w_stagel_icon = w_tree_dialog.get_widget("icon_stagel")

```

```

126 self.w_stage2_label = w_tree_dialog.get_widget("label_stage2")
127 self.w_stage2_icon = w_tree_dialog.get_widget("icon_stage2")
128 self.w_stage3_label = w_tree_dialog.get_widget("label_stage3")
129 self.w_stage3_icon = w_tree_dialog.get_widget("icon_stage3")
130 self.w_stages_label = w_tree_dialog.get_widget("label_stages")
131 self.w_stages_icon = w_tree_dialog.get_widget("icon_stages")
132 self.current_stage_label = self.w_stage1_label
133 self.current_stage_icon = self.w_stage1_icon
134 self.current_stage_label_done = None

136 self.done_icon = gui_misc.get_icon(
137     self.parent.icon_theme, "progress_checkmark")
138 blank_icon = gui_misc.get_icon(
139     self.parent.icon_theme, "progress_blank")

141 self.w_stage1_icon.set_from_pixbuf(blank_icon)
142 self.w_stage2_icon.set_from_pixbuf(blank_icon)
143 self.w_stage3_icon.set_from_pixbuf(blank_icon)

145 infobuffer = self.w_details_textview.get_buffer()
146 infobuffer.create_tag("bold", weight=pango.WEIGHT_BOLD)
147 infobuffer.create_tag("level1", left_margin=30, right_margin=10)
148 infobuffer.create_tag("level2", left_margin=50, right_margin=10)
149 self.w_ua_dialog = w_tree_uaconfirm.get_widget("ua_confirm_dialo
150 self.w_ua_error_label = w_tree_uaconfirm.get_widget(
151     "ua_confirm_error_label")
152 self.w_ua_proceed_button = w_tree_uaconfirm.get_widget(
153     "ua_proceed_button")
154 self.w_ua_be_name_entry = w_tree_uaconfirm.get_widget(
155     "ua_be_name_entry")
156 self.w_ua_be_name_box = w_tree_uaconfirm.get_widget(
157     "ua_be_name_box")

159 w_ua_proceed_button = w_tree_uaconfirm.get_widget("ua_proceed_bu
160 self.w_progressbar.set_pulse_step(0.02)
161 try:
162     dic_createplan = \
163     {
164         "on_cancelcreateplan_clicked": \
165         self.__on_cancelcreateplan_clicked,
166         "on_createplandialog_delete_event": \
167         self.__on_createplandialog_delete,
168     }
169     dic_uaconfirm = \
170     {
171         "on_ua_cancel_button_clicked": \
172         self.__on_ua_cancel_button_clicked,
173         "on_ua_proceed_button_clicked": \
174         self.__on_ua_proceed_button_clicked,
175         "on_ua_be_name_entry_changed": \
176         self.__on_ua_be_name_entry_changed,
177         "on_ua_help_button_clicked": \
178         self.__on_ua_help_button_clicked,
179     }
180     dic_removeconfirm = \
181     {
182         "on_proceed_button_clicked": \
183         self.__on_remove_proceed_button_clicked,
184         "on_cancel_button_clicked": \
185         self.__on_remove_cancel_button_clicked,
186     }
187     w_tree_dialog.signal_autoconnect(dic_createplan)
188     w_tree_uaconfirm.signal_autoconnect(dic_uaconfirm)
189     w_tree_removeconfirm.signal_autoconnect(dic_removeconfir
190 except AttributeError, error:
191     print _("GUI will not respond to any event! %s. ")

```

```

192         "Check installupdate.py signals") \
193         % error

196 self.w_dialog.set_transient_for(self.w_main_window)
197 self.w_ua_dialog.set_transient_for(self.w_main_window)
198 if self.icon_confirm_dialog != None:
199     self.w_ua_dialog.set_icon(self.icon_confirm_dialog)

201 if self.action == enumerations.REMOVE:
202     #We are not showing the download stage in the main stage
203     self.stages[3] = [(_("Removing..."), _("Remove"))]
204     self.w_stage3_label.set_text(self.stages[3][1])
205     w_stage2.hide()
206     self.w_dialog.set_title(_("Remove"))
207     w_removeproceed_button.grab_focus()
208     cell = gtk.CellRendererText()
209     remove_column = gtk.TreeViewColumn('Removed')
210     remove_column.pack_start(cell, True)
211     remove_column.add_attribute(cell, 'text', 0)
212     w_remove_treeview.append_column(remove_column)

214     liststore = gtk.ListStore(str)
215     for sel_pkg in list_of_packages:
216         liststore.append([sel_pkg])
217     w_remove_treeview.set_model(liststore)
218     w_remove_treeview.expand_all()
219     self.w_removeconfirm_dialog.show()

221 elif self.action == enumerations.IMAGE_UPDATE:
222     self.w_dialog.set_title(_("Update All"))
223     w_ua_proceed_button.grab_focus()
224     if not self.be_name:
225         if nobe or not "beVerifyBEName" in be.__dict__:
226             self.w_ua_be_name_box.set_property(
227                 "visible", False)
228         else:
229             self.__setup_be_list()
230             self.w_ua_dialog.show()
231     else:
232         self.proposed_be_name = self.be_name
233         self.__proceed_with_stages()
234     else:
235         if self.title != None:
236             self.w_dialog.set_title(self.title)
237         else:
238             self.w_dialog.set_title(_("Install/Update"))
239         self.__proceed_with_stages()

242 def __on_createplandialog_delete(self, widget, event):
243     self.__on_cancelcreateplan_clicked(None)
244     return True

246 def __on_cancelcreateplan_clicked(self, widget):
247     '''Handler for signal send by cancel button, which user might pr
248     evaluation stage - while the dialog is creating plan'''
249     if self.api_o.can_be_canceled():
250         self.canceling = True
251         Thread(target = self.api_o.cancel, args = []).start()
252         cancel_txt = _("Canceling...")
253         txt = "<b>" + self.current_stage_label_done + " - " \
254             + cancel_txt + "</b>"
255         gobject.idle_add(self.current_stage_label.set_markup, tx
256         gobject.idle_add(self.current_stage_icon.set_from_stock,
257             gtk.STOCK_CANCEL, gtk.ICON_SIZE_MENU)

```

```

258         gobject.idle_add(self.w_stages_label.set_markup, cancel_
259         self.w_cancel_button.set_sensitive(False)
260     if self.operations_done:
261         self.w_dialog.hide()
262     if self.web_install:
263         gobject.idle_add(self.parent.update_package_list
264         self.web_updates_list)
265     return
266     gobject.idle_add(self.parent.update_package_list, None)

268 def __on_ua_help_button_clicked(self, widget):
269     gui_misc.display_help(self.parent.application_dir, "update_all")
270
271 def __on_ua_cancel_button_clicked(self, widget):
272     self.w_ua_dialog.hide()
273     if self.web_install:
274         gobject.idle_add(self.parent.update_package_list,
275         self.web_updates_list)
276     return
277     gobject.idle_add(self.parent.update_package_list, None)

279 def __on_ua_proceed_button_clicked(self, widget):
280     proposed_be_name = self.w_ua_be_name_entry.get_text()
281     if proposed_be_name != "":
282         self.proposed_be_name = proposed_be_name
283     self.w_ua_dialog.hide()
284     self.__proceed_with_stages()

286 def __setup_be_list(self):
287     be_list = be.beList()
288     error_code = None
289     if len(be_list) > 1 and type(be_list[0]) == type(-1):
290         error_code = be_list[0]
291     if error_code != None and error_code == 0:
292         self.be_list = be_list[1]
293     elif error_code == None:
294         self.be_list = be_list

296     # Now set proposed name in entry field.
297     active_name = None
298     for bee in self.be_list:
299         name = bee.get("orig_be_name")
300         if name:
301             if bee.get("active"):
302                 active_name = name
303                 break
304     if active_name != None:
305         proposed_name = None
306         list = string.rsplit(active_name, '-', 1)
307         if len(list) == 1:
308             proposed_name = self.__construct_be_name(
309             active_name, 0)
310         else:
311             try:
312                 i = int(list[1])
313                 proposed_name = self.__construct_be_name
314                 list[0], i)
315             except ValueError:
316                 proposed_name = self.__construct_be_name
317                 active_name, 0)

319     if proposed_name != None:
320         self.w_ua_be_name_entry.set_text(proposed_name)

322 def __construct_be_name(self, name, i):

```

```

324         in_use = True
325         proposed_name = None
326         while in_use:
327             i += 1
328             proposed_name = name + '-' + str(i)
329             in_use = self.__is_be_name_in_use(proposed_name)
330         return proposed_name
331
332 def __is_be_name_in_use(self, name):
333     in_use = False
334     if name == "":
335         return in_use
336     for bee in self.be_list:
337         be_name = bee.get("orig_be_name")
338         if be_name == name:
339             in_use = True
340             break
341     return in_use
342
343 def __is_be_name_valid(self, name):
344     if name == "":
345         return True
346     return be.beVerifyBENAME(name) == 0
347
348 def __validate_be_name(self, widget):
349     name = widget.get_text()
350     is_name_valid = self.__is_be_name_valid(name)
351     self.w_ua_error_label.hide()
352     error_str = None
353     if is_name_valid:
354         is_name_in_use = self.__is_be_name_in_use(name)
355         if is_name_in_use:
356             error_str = ERROR_FORMAT % _("BE name is in use")
357     else:
358         error_str = ERROR_FORMAT % _("BE name is invalid")
359     if error_str != None:
360         self.w_ua_error_label.set_markup(error_str)
361         self.w_ua_error_label.show()
362
363     self.w_ua_proceed_button.set_sensitive(error_str == None)

365 def __on_ua_be_name_entry_changed(self, widget):
366     self.__validate_be_name(widget)

368 def __on_remove_cancel_button_clicked(self, widget):
369     self.w_removeconfirm_dialog.hide()

371 def __on_remove_proceed_button_clicked(self, widget):
372     self.w_removeconfirm_dialog.hide()
373     self.__proceed_with_stages()

375 def __ipkg_ipkgui_uptodate(self):
376     if self.ipkg_ipkgui_list == None:
377         return True
378     upgrade_needed, cre = self.api_o.plan_install(
379     self.ipkg_ipkgui_list, filters = [])
380     return not upgrade_needed

382 def __proceed_with_stages(self):
383     self.__start_stage_one()
384     self.w_dialog.show()
385     Thread(target = self.__proceed_with_stages_thread_ex,
386     args = ()).start()

388 def __proceed_with_stages_thread_ex(self):
389     try:

```

```

390         if self.action == enumerations.IMAGE_UPDATE:
391             self.__start_substage(
392                 _("Ensuring %s is up to date...") % self.par
393                 bounce_progress=True)
394             opensolaris_image = True
395             ips_uptodate = True
396             notfound = self.__installed_fmris_from_args(
397                 ["SUNWipkg", "SUNWcs"])
398             if notfound:
399                 opensolaris_image = False
400             if opensolaris_image:
401                 ips_uptodate = self.__ipkg_ipkgui_uptoda
402             if not ips_uptodate:
403                 #Do the stuff with installing ipkg ipkkg
404                 #restart in the special mode
405                 self.ips_update = True
406                 self.__proceed_with_ipkg_thread()
407                 return
408             else:
409                 self.api_o.reset()
410             self.__proceed_with_stages_thread()
411         except api_errors.CertificateError:
412             self.stop_bouncing_progress = True
413             msg = _("Accessing this restricted repository failed."
414                 "\nYou either need to register to access this repositi
415                 "\nthe certificate expired, or you need to accept th
416                 " repository\ncertificate.")
417             self.__g_error_stage(msg)
418             return
419         except api_errors.PlanCreationException, e:
420             self.__g_error_stage(str(e))
421             return
422         except api_errors.InventoryException, e:
423             msg = _("Inventory exception:\n")
424             if e.illegal:
425                 for i in e.illegal:
426                     msg += "\tpkg:\t" + i + "\n"
427             self.__g_error_stage(msg)
428             return
429         except api_errors.CatalogRefreshException, e:
430             msg = _("Please check the network "
431                 "connection.\nIs the repository accessible?")
432             if e.message and len(e.message) > 0:
433                 msg = e.message
434             self.__g_error_stage(msg)
435             return
436         except (api_errors.NetworkUnavailableException,
437             TransferTimedOutException, TransportException, URLError,
438             ManifestRetrievalError, DatastreamRetrievalError,
439             FileListRetrievalError), ex:
440             msg = _("Please check the network "
441                 "connection.\nIs the repository accessible?\n\n"
442                 "%s") % str(ex)
443             self.__g_error_stage(msg)
444             return
445         except api_errors.InvalidDepotResponseException, e:
446             msg = _("Unable to contact a valid package depot. "
447                 "Please check your network\nsettings and "
448                 "attempt to contact the server using a web "
449                 "browser.\n\n%s") % str(e)
450             self.__g_error_stage(msg)
451             return
452         except api_errors.IpkgOutOfDateException:
453             msg = _("pkg(5) appears to be out of "
454                 "date and should be\nupdated before running "
455                 "Update All.\nPlease update SUNWipkg package")

```

```

456             self.__g_error_stage(msg)
457             return
458         except api_errors.NonLeafPackageException, nlpe:
459             msg = _("Cannot remove:\n\t%s\n"
460                 "Due to the following packages that "
461                 "depend on it:\n") % nlpe[0].get_name()
462             for pkg_a in nlpe[1]:
463                 msg += "\t" + pkg_a.get_name() + "\n"
464             self.__g_error_stage(msg)
465             return
466         except api_errors.ProblematicPermissionsIndexException, err:
467             msg = str(err)
468             msg += _("\nFailure of consistent use of pexec or gksu
469                 "running\n%s is often a source of this problem.") %
470                 self.parent_name
471             msg += _("\nTo rebuild index, please use the terminal co
472                 "msg += _("\ntpexec pkg rebuild-index")
473             self.__g_error_stage(msg)
474             return
475         except api_errors.CorrruptedIndexException:
476             msg = _("There was an error during installation. The sea
477                 "index is corrupted. You might want try to fix this\
478                 "problem by running command:\n"
479                 "\tpexec pkg rebuild-index")
480             self.__g_error_stage(msg)
481             return
482         except api_errors.ImageUpdateOnLiveImageException:
483             msg = _("This is an Live Image. The install"
484                 "\noperation can't be performed.")
485             self.__g_error_stage(msg)
486             return
487         except api_errors.PlanMissingException:
488             msg = _("There was an error during installation.\n"
489                 "The Plan of the operation is missing and the operat
490                 "can't be finished. You might want try to fix this\n
491                 "problem by restarting %s\n") % self.parent_name
492             self.__g_error_stage(msg)
493             return
494         except api_errors.ImageplanStateException:
495             msg = _("There was an error during installation.\n"
496                 "The State of the image is incorrect and the operati
497                 "can't be finished. You might want try to fix this\n
498                 "problem by restarting %s\n") % self.parent_name
499             self.__g_error_stage(msg)
500             return
501         except api_errors.CanceledException:
502             gobject.idle_add(self.w_dialog.hide)
503             self.stop_bouncing_progress = True
504             return
505         except api_errors.BENamingNotSupported:
506             msg = _("Specifying BE Name not supported.\n")
507             self.__g_error_stage(msg)
508             return
509         except api_errors.InvalidBENAMEException:
510             msg = _("Invalid BE Name: %s.\n") % self.proposed_be_nam
511             self.__g_error_stage(msg)
512             return
513         except api_errors.PermissionsException, pex:
514             msg = str(pex)
515             self.__g_error_stage(msg)
516             return
517         except (api_errors.UnableToCopyBE,
518             api_errors.UnableToMountBE,
519             api_errors.BENAMEGivenOnDeadBE,
520             api_errors.UnableToRenameBE), ex:
521             msg = str(ex)

```

```

522         self.__g_error_stage(msg)
523         return
524     except Exception, uex:
525         # We do want to prompt user to load BE admin if there is
526         # not enough disk space. This error can either come as a
527         # error within API exception, see bug #7642 or as a stan
528         # error, that is why we need to check for both situation
529         if ("error" in uex.__dict__ and isinstance(uex.error, OS
530             and ("args" in uex.error.__dict__ and uex.error.args
531                 (uex.error.args[0] == errno.EDQUOT or
532                 uex.error.args[0] == errno.ENOSPC)) \
533                 or ("args" in uex.__dict__ and uex.args and (uex.arg
534                     errno.EDQUOT or uex.args[0] == errno.ENOSPC)):
535             gobject.idle_add(self.__prompt_to_load_beadm)
536             gobject.idle_add(self.w_dialog.hide)
537             self.stop_bouncing_progress = True
538         else:
539             traceback_lines = traceback.format_exc().splitli
540             traceback_str = ""
541             for line in traceback_lines:
542                 traceback_str += line + "\n"
543             self.__g_exception_stage(traceback_str)
544             sys.exc_clear()

546     def __proceed_with_ipkg_thread(self):
547         self.__start_substage(_("Updating %s") % self.parent_name,
548                               bounce_progress=True)
549         self.__afterplan_information()
550         self.prev_pkg = None
551         self.__start_substage(_("Downloading..."), bounce_progress=False)
552         self.api_o.prepare()
553         self.__start_substage(_("Executing..."), bounce_progress=False)
554         self.api_o.execute_plan()
555         gobject.idle_add(self.__operations_done)

558     def __proceed_with_stages_thread(self):
559         self.__start_substage(
560             _("Gathering package information, please wait..."))
561         stuff_todo = self.__plan_stage()
562         if stuff_todo:
563             self.__afterplan_information()
564             self.prev_pkg = None
565             # The api.prepare() mostly is downloading the files so w
566             # Not showing this stage in the main stage dialog. If do
567             # is necessary, then we are showing it in the details vi
568             if not self.action == enumerations.REMOVE:
569                 self.__start_stage_two()
570                 self.__start_substage(None,
571                                       bounce_progress=False)
572             self.api_o.prepare()
573             self.__start_stage_three()
574             self.__start_substage(None,
575                                   bounce_progress=False)
576             self.api_o.execute_plan()
577             gobject.idle_add(self.__operations_done)
578         else:
579             if self.web_install:
580                 gobject.idle_add(self.w_expander.hide)
581                 gobject.idle_add(self.__operations_done,
582                                   _("All packages already installed.))
583             return
584         msg = None
585         if self.action == enumerations.INSTALL_UPDATE:
586             msg = _("Selected package(s) cannot be updated o

```

```

588         "their own.\nClick Update All to update all pack
589         elif self.action == enumerations.IMAGE_UPDATE:
590             msg = _("Your system has already been updated.")
591             self.__g_error_stage(msg)

593     def __start_stage_one(self):
594         self.current_stage_label = self.w_stage1_label
595         self.current_stage_icon = self.w_stage1_icon
596         self.__start_stage(self.stages.get(1))
597         self.__g_update_details_text(self.stages.get(1)[0]+\n", "bold")
598
599     def __start_stage_two(self):
600         # End previous stage
601         self.__end_stage()
602         self.current_stage_label = self.w_stage2_label
603         self.current_stage_icon = self.w_stage2_icon
604         self.__start_stage(self.stages.get(2))
605         self.__g_update_details_text(self.stages.get(2)[0]+\n", "bold")

607     def __start_stage_three(self):
608         self.__end_stage()
609         self.current_stage_label = self.w_stage3_label
610         self.current_stage_icon = self.w_stage3_icon
611         self.__start_stage(self.stages.get(3))
612         self.__g_update_details_text(self.stages.get(3)[0]+\n", "bold")

614     def __start_stage(self, stage_text):
615         self.current_stage_label_done = stage_text[1]
616         gobject.idle_add(self.current_stage_label.set_markup,
617                           "<b>"+stage_text[0]+"</b>")
618         gobject.idle_add(self.current_stage_icon.set_from_stock,
619                           gtk.STOCK_GO_FORWARD, gtk.ICON_SIZE_MENU)

621     def __end_stage(self):
622         gobject.idle_add(self.current_stage_label.set_text,
623                           self.current_stage_label_done)
624         gobject.idle_add(self.current_stage_icon.set_from_pixbuf, self.d

626     def __g_error_stage(self, msg):
627         if msg == None or len(msg) == 0:
628             msg = _("No futher information available")
629         self.operations_done = True
630         self.stop_bouncing_progress = True
631         self.__g_update_details_text(_("Error:\n"), "bold")
632         self.__g_update_details_text("%s" % msg, "levell")
633         self.__g_update_details_text("\n")
634         txt = "<b>" + self.current_stage_label_done + _(" - Failed </b>"
635         gobject.idle_add(self.current_stage_label.set_markup, txt)
636         gobject.idle_add(self.current_stage_icon.set_from_stock,
637                           gtk.STOCK_DIALOG_ERROR, gtk.ICON_SIZE_MENU)
638         gobject.idle_add(self.w_expander.set_expanded, True)
639         gobject.idle_add(self.w_cancel_button.set_sensitive, True)

641     def __g_exception_stage(self, traceback):
642         self.operations_done = True
643         self.stop_bouncing_progress = True
644         txt = "<b>" + self.current_stage_label_done + _(" - Failed </b>"
645         gobject.idle_add(self.current_stage_label.set_markup, txt)
646         gobject.idle_add(self.current_stage_icon.set_from_stock,
647                           gtk.STOCK_DIALOG_ERROR, gtk.ICON_SIZE_MENU)
648         msg_1 = _("An unknown error occurred in the %s stage.\n")
649         "Please let the developers know about this problem\n"
650         "by filing a bug together with exception value at:\n")
651         ) % self.current_stage_name
652         msg_2 = _("http://defect.opensolaris.org\n")
653         msg_3 = _("Exception value:\n")

```

```

654     self.__g_update_details_text(_("\nError:\n"), "bold")
655     self.__g_update_details_text("%s" % msg_1, "level1")
656     self.__g_update_details_text("%s" % msg_2, "bold", "level2")
657     if tracebk:
658         msg = _("Exception traceback:\n")
659         self.__g_update_details_text("%s" % msg,
660                                     "bold", "level1")
661         self.__g_update_details_text("%s\n" % tracebk, "level2")
662     else:
663         msg = _("No futher information available")
664         self.__g_update_details_text("%s\n" % msg, "level2")
665     GObject.idle_add(self.w_expander.set_expanded, True)
666     GObject.idle_add(self.w_cancel_button.set_sensitive, True)

668 def __start_substage(self, text, bounce_progress=True):
669     if text:
670         GObject.idle_add(self.__stages_label.set_markup, text)
671         self.__g_update_details_text(text + "\n")
672     if bounce_progress:
673         if self.stopped_bouncing_progress:
674             self.__start_bouncing_progress()
675     else:
676         self.stop_bouncing_progress = True

678 def __stages_label_set_markup(self, markup_text):
679     if not self.canceling == True:
680         self.w_stages_label.set_markup(markup_text)

682 def __start_bouncing_progress(self):
683     self.stop_bouncing_progress = False
684     self.stopped_bouncing_progress = False
685     Thread(target =
686            self.__g_progressdialog_progress_pulse).start()

688 def __g_progressdialog_progress_pulse(self):
689     while not self.stop_bouncing_progress:
690         GObject.idle_add(self.w_progressbar.pulse)
691         time.sleep(0.1)
692     self.stopped_bouncing_progress = True

694 def __g_update_details_text(self, text, *tags):
695     GObject.idle_add(self.__update_details_text, text, *tags)

697 def __update_details_text(self, text, *tags):
698     buf = self.w_details_textview.get_buffer()
699     textiter = buf.get_end_iter()
700     if tags:
701         buf.insert_with_tags_by_name(textiter, text, *tags)
702     else:
703         buf.insert(textiter, text)
704     self.w_details_textview.scroll_to_iter(textiter, 0.0)

706 def __update_download_progress(self, cur_bytes, total_bytes):
707     prog = float(cur_bytes)/total_bytes
708     self.w_progressbar.set_fraction(prog)
709     size_a_str = ""
710     size_b_str = ""
711     if cur_bytes >= 0:
712         size_a_str = pkg.misc.bytes_to_str(cur_bytes)
713     if total_bytes >= 0:
714         size_b_str = pkg.misc.bytes_to_str(total_bytes)
715     c = _("Downloaded %(current)s of %(total)s") % \
716         {"current" : size_a_str,
717          "total" : size_b_str}
718     self.__stages_label_set_markup(c)

```

```

720     def __update_install_progress(self, current, total):
721         prog = float(current)/total
722         self.w_progressbar.set_fraction(prog)

724     def __plan_stage(self):
725         '''Function which plans the image'''
726         stuff_to_do = False
727         if self.action == enumerations.INSTALL_UPDATE:
728             stuff_to_do, cre = self.api_o.plan_install(
729                 self.list_of_packages, refresh_catalogs = False,
730                 filters = [])
731             if cre and not cre.succeeded:
732                 # cre is either None or a catalog refresh except
733                 # which was caught while planning.
734                 raise api_errors.CatalogRefreshException(None, N
735                    ,_("Catalog refresh failed during install.))
736             elif self.action == enumerations.REMOVE:
737                 plan_uninstall = self.api_o.plan_uninstall
738                 stuff_to_do = \
739                     plan_uninstall(self.list_of_packages, False, False)
740             elif self.action == enumerations.IMAGE_UPDATE:
741                 # we are passing force, since we already checked if the
742                 # SUNWipkg and SUNWipkg-gui are up to date.
743                 stuff_to_do, opensolaris_image, cre = \
744                     self.api_o.plan_update_all(sys.argv[0],
745                 refresh_catalogs = False,
746                 noexecute = False, force = True,
747                 be_name = self.proposed_be_name)
748             if cre and not cre.succeeded:
749                 raise api_errors.CatalogRefreshException(None, N
750                    ,_("Catalog refresh failed during Update All
751                    return stuff_to_do

753     def __operations_done(self, alternate_done_txt = None):
754         done_txt = _("Installation completed successfully.")
755         if self.action == enumerations.REMOVE:
756             done_txt = _("Packages removed successfully.")
757         elif self.action == enumerations.IMAGE_UPDATE:
758             done_txt = _("Packages updated successfully.")
759         if alternate_done_txt != None:
760             done_txt = alternate_done_txt
761         self.w_stages_box.hide()
762         self.w_stages_icon.set_from_stock(
763             gtk.STOCK_OK, gtk.ICON_SIZE_DND)
764         self.w_stages_icon.show()
765         self.__stages_label_set_markup(done_txt)
766         self.__update_details_text("\n"+ done_txt, "bold")
767         self.w_cancel_button.set_label("gtk-close")
768         self.w_progressbar.hide()
769         self.stop_bouncing_progress = True
770         self.operations_done = True
771         if self.parent != None:
772             if not self.web_install and not self.ips_update \
773                 and not self.action == enumerations.IMAGE_UPDATE:
774                 self.parent.update_package_list(self.update_list)
775             if self.web_install:
776                 self.web_updates_list = self.update_list
777         if self.ips_update:
778             self.w_dialog.hide()
779             self.parent.restart_after_ips_update(self.proposed_be_na
780         elif self.action == enumerations.IMAGE_UPDATE:
781             self.w_dialog.hide()
782             self.parent.shutdown_after_image_update()

784     def __prompt_to_load_beadm(self):
785         msgbox = gtk.MessageDialog(parent = self.w_main_window,

```

```

786         buttons = gtk.BUTTONS_OK_CANCEL, flags = gtk.DIALOG_MODAL,
787         type = gtk.MESSAGE_ERROR,
788         message_format = _("
789             "Not enough disk space, the selected action cannot "
790             "be performed.\n\n"
791             "Click OK to manage your existing BEs and free up disk s
792             "Cancel to cancel the action.")
793         msgbox.set_title(_("Not Enough Disk Space"))
794         result = msgbox.run()
795         msgbox.destroy()
796         if result == gtk.RESPONSE_OK:
797             beadm.Beadmin(self.parent)

799     def __afterplan_information(self):
800         install_iter = None
801         update_iter = None
802         remove_iter = None
803         plan = self.api_o.describe().get_changes()
804         self.__g_update_details_text("\n")
805         for pkg_plan in plan:
806             origin_fmri = pkg_plan[0]
807             destination_fmri = pkg_plan[1]
808             if origin_fmri and destination_fmri:
809                 if not update_iter:
810                     update_iter = True
811                     txt = _("Packages To Be Updated:\n")
812                     self.__g_update_details_text(txt, "bold")
813                     pkg_a = self.__get_pkgstr_from_pkginfo(destination_fmri)
814                     self.__g_update_details_text(pkg_a+"\n", "level1")
815             elif not origin_fmri and destination_fmri:
816                 if not install_iter:
817                     install_iter = True
818                     txt = _("Packages To Be Installed:\n")
819                     self.__g_update_details_text(txt, "bold")
820                     pkg_a = self.__get_pkgstr_from_pkginfo(destination_fmri)
821                     self.__g_update_details_text(pkg_a+"\n", "level1")
822             elif origin_fmri and not destination_fmri:
823                 if not remove_iter:
824                     remove_iter = True
825                     txt = _("Packages To Be Removed:\n")
826                     self.__g_update_details_text(txt, "bold")
827                     pkg_a = self.__get_pkgstr_from_pkginfo(origin_fmri)
828                     self.__g_update_details_text(pkg_a+"\n", "level1")
829             self.__g_update_details_text("\n")

831     def __get_pkgstr_from_pkginfo(self, pkginfo):
832         dt_str = self.get_datetime(pkginfo.packaging_date)
833         if not dt_str:
834             dt_str = ""
835         s_ver = pkginfo.version
836         s_bran = pkginfo.branch
837         pkg_name = pkginfo.pkg_stem
838         pkg_publisher = pkginfo.publisher
839         if not pkg_publisher in self.update_list:
840             self.update_list[pkg_publisher] = []
841         pub_list = self.update_list.get(pkg_publisher)
842         if not pkg_name in pub_list:
843             pub_list.append(pkg_name)
844         l_ver = 0
845         version_pref = ""
846         while l_ver < len(s_ver) - 1:
847             version_pref += "%d%s" % (s_ver[l_ver], ".")
848             l_ver += 1
849         version_pref += "%d%s" % (s_ver[l_ver], "-")
850         l_ver = 0
851         version_suf = ""

```

```

852         if s_bran != None:
853             while l_ver < len(s_bran) - 1:
854                 version_suf += "%d%s" % (s_bran[l_ver], ".")
855                 l_ver += 1
856             version_suf += "%d" % s_bran[l_ver]
857         pkg_version = version_pref + version_suf + dt_str
858         return pkg_name + "@" + pkg_version

860     def act_output(self):
861         if self.act_phase != self.act_phase_last:
862             self.act_phase_last = self.act_phase
863             GObject.idle_add(self.__stages_label_set_markup, self.act_phase)
864             self.__g_update_details_text(_("%s\n") % self.act_phase)
865             GObject.idle_add(self.__update_install_progress,
866                             self.act_cur_nactions, self.act_goal_nactions)
867             return

869     def act_output_done(self):
870         return

872     def cat_output_start(self):
873         return

875     def cat_output_done(self):
876         return

878     def cache_cats_output_start(self):
879         return

881     def cache_cats_output_done(self):
882         return

884     def load_cat_cache_output_start(self):
885         return

887     def load_cat_cache_output_done(self):
888         return

890     def dl_output(self):
891         GObject.idle_add(self.__update_download_progress, \
892                         self.dl_cur_nbytes, self.dl_goal_nbytes)
893         if self.prev_pkg != self.dl_cur_pkg:
894             self.prev_pkg = self.dl_cur_pkg
895             self.__g_update_details_text(
896                 _("Package %d of %d: %s\n") % (self.dl_cur_npkgs+1,
897                                                 self.dl_goal_npkgs, self.dl_cur_pkg), "level1")

899     def dl_output_done(self):
900         self.__g_update_details_text("\n")

902     def eval_output_start(self):
903         '''Called by progress tracker when the evaluation of the package
904         started.'''
905         return

907     def eval_output_progress(self):
908         '''Called by progress tracker each time some package was evaluat
909         call is being done by calling progress tracker evaluate_progress
910         function'''
911         if self.prev_pkg != self.eval_cur_fmri:
912             self.prev_pkg = self.eval_cur_fmri
913             self.__g_update_details_text(_("%s\n") % self.eval_cur_fmri)
914             "level1")
915         text = _("Evaluating: %s") % self.eval_cur_fmri.get_name()
916         GObject.idle_add(self.__stages_label_set_markup, text)

```

```
918     def eval_output_done(self):
919         return
921     def ind_output(self):
922         if self.ind_started != self.ind_phase:
923             self.ind_started = self.ind_phase
924             gobject.idle_add(self.__stages_label_set_markup, self.in
925             self.__g_update_details_text(
926                 _("%s\n") % (self.ind_phase), "levell")
927             gobject.idle_add(self.__indexing_progress)
929     def __indexing_progress(self):
930         #It doesn't look nice if the progressive is just for few element
931         if self.ind_goal_nitems > MIN_IND_ELEMENTS_BOUNCE:
932             gobject.idle_add(self.__update_install_progress,
933                 self.ind_cur_nitems-1, self.ind_goal_nitems)
934         else:
935             if self.stopped_bouncing_progress:
936                 self.__start_bouncing_progress()
937
938     def ind_output_done(self):
939         gobject.idle_add(self.__update_install_progress, self.ind_cur_ni
940             self.ind_goal_nitems)
942     def ver_output(self):
943         return
945     def ver_output_error(self, actname, errors):
946         return
948     @staticmethod
949     def get_datetime(date_time):
950         '''Support function for getting date from the API.'''
951         date_tmp = None
952         try:
953             date_tmp = time.strptime(date_time, "%a %b %d %H:%M:%S %
954         except ValueError:
955             return None
956         if date_tmp:
957             date_tmp2 = datetime.datetime(*date_tmp[0:5])
958             return date_tmp2.strftime(":%m%d")
959         return None
961     def __installed_fmris_from_args(self, args_f):
962         found = []
963         notfound = []
964         try:
965             for m in self.api_o.img.inventory(args_f):
966                 found.append(m[0])
967         except api_errors.InventoryException, e:
968             notfound = e.notfound
969         return notfound
```