```cpp
#include "StdAfx.h"
#include <afxdllx.h>
#include <string>
using namespace std;

#include "Resource.h"

#ifndef _CPROPMEN_H
#include "cPropMen.h"
#endif

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/// Специальная структура используемая в течении инициализации DLL
static AFX_EXTENSION_MODULE SMP_DLL = { NULL, NULL };

/// Компас Application 7
IApplicationPtr application;
/// Менеджер панели свойств
IPropertyManagerPtr propMng;
IPropertyTabPtr tab;
/// Окно, в котором будет отрисовываться текущий документ
IPropertySlideBoxPtr slideBox;

/// Положение панели
PropertyManagerLayout g_Layout = PropertyManagerLayout::pmAlignRight;
/// Размер
CRect g_Rect(20, 20, 270, 320);

/// Меню создания модели по профилю режущей кромки
MenuProfile* menu_profile = new MenuProfile;
/// Меню создания модели по режимам резания
MenuConditions* menu_conditions = new MenuConditions;
/// Тип текущего меню
enum MenuType* menu_type = new enum MenuType(MenuType::type_none);

void ClosePropertyManager(int mes);
void CreateAndSubscriptionPropertyManager();

// Функции создания подписок на события Компаса
extern cBaseEvent* NewApplicationEvent();
extern cBaseEvent* NewDocumentEvent(reference doc);
extern cBaseEvent* NewDocument2DEvent(reference doc);
extern cBaseEvent* NewDocument3DEvent(reference doc);
extern cBaseEvent* NewObj2DEvent(reference doc, int objType);
extern cBaseEvent* NewObj3DEvent(reference doc, int objType, LPUNKNOWN iObj = NULL);
extern cBaseEvent* NewSpcDocEvent(reference doc);
extern cBaseEvent* NewSpcObjectEvent(reference doc, int objType);
extern cBaseEvent* NewSpecificationEvent(reference doc);
extern cBaseEvent* NewStampEvent(reference doc);
```

```
55  ISurfaceContainerPtr GetSurfaceContainer(IKompasDocumentPtr& doc);
56  IDocumentFramePtr GetActiveFrame(IKompasDocumentPtr& doc);
57  void AdviseDoc(long pDoc);
58
59  void buildProfileObject();
60  void buildConditionsObject();
61
62  /// <summary>
63  /// Стандартная точка входа
64  /// Инициализация и завершение DLL
65  /// </summary>
66  /// <param name="hInstance"></param>
67  /// <param name="dwReason"></param>
68  /// <param name="lpReserved"></param>
69  /// <returns></returns>
70  extern "C" int APIENTRY
71  DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID lpReserved) {
72      UNREFERENCED_PARAMETER(lpReserved);
73
74      if (dwReason == DLL_PROCESS_ATTACH) {
75          TRACE0("DLL Initializing!");
76
77          if (!AfxInitExtensionModule(SMP_DLL, hInstance))
78              return 0;
79
80          new CDynLinkLibrary(SMP_DLL);
81      }
82      else if (dwReason == DLL_PROCESS_DETACH) {
83          // Отписка событий
84          ABaseEvent::TerminateEvents();
85          cBaseEvent::TerminateEvents();
86          // Удаление панели
87          ClosePropertyManager(0/* mes*/);
88          // Удаление списка событий
89          ABaseEvent::DestroyList();
90          cBaseEvent::DestroyList();
91
92          application = NULL;
93          TRACE0("DLL Terminating!");
94          AfxTermExtensionModule(SMP_DLL);
95      }
96      return 1;
97  }
98
99  /// <summary>
100 /// Получить доступ к новому API
101 /// </summary>
102 void Getapplication() {
103     if (!(IApplication*)application) {
104         CString filename;
105         if (::GetModuleFileName(NULL, filename.GetBuffer(255), 255)) {
106             filename.ReleaseBuffer(255);
107             CString libname;
108
109 #ifdef __LIGHT_VERSION__
```

```
110             libname.LoadString(IDS_API7LT); /// klAPI7.dll
111 #else
112             libname.LoadString(IDS_API7); /// kAPI7.dll
113 #endif
114
115             filename.Replace(filename.Right(filename.GetLength() -
                  (filename.ReverseFind(_T('\\')) + 1)),
116                 libname);
117
118             HINSTANCE hAppAuto = LoadLibrary(filename); ///
                  идентификатор kAPI7.dll
119             if (hAppAuto) {
120                 /// Указатель на функцию возвращающую интерфейс
                      KompasApplication
121                 typedef LPDISPATCH(WINAPI* FCreateKompasApplication)();
122
123                 FCreateKompasApplication pCreateKompasApplication =
124                     (FCreateKompasApplication)GetProcAddress(hAppAuto,
                      "CreateKompasApplication");
125                 if (pCreateKompasApplication)
126                     application = IDispatchPtr(pCreateKompasApplication
                      (), false /*AddRef*/);
127                 FreeLibrary(hAppAuto);
128             }
129         }
130     }
131 }
132
133 /// <summary>
134 /// Загрузить строку из ресурсов
135 /// </summary>
136 /// <param name="ID"></param>
137 /// <returns></returns>
138 TCHAR* LoadStr(int ID) {
139     static TCHAR buf[255];
140     ksConvertLangStrExT(SMP_DLL.hModule, ID, buf, 255);
141     return buf;
142 }
143
144 /// <summary>
145 /// Вывод сообщения
146 /// </summary>
147 /// <param name="str"></param>
148 /// <param name="flags"></param>
149 /// <returns></returns>
150 int LibMessage(LPCTSTR str, int flags = MB_OK) {
151     int res = 0;
152
153     if (str && str[0]) {                        /// строка передана
154         int enable = ::IsEnableTaskAccess();    /// проверка доступа
155         if (enable)                             /// если доступ к
              задаче разрешен
156             EnableTaskAccess(0);            /// запрещаем доступ
157
158         // текст сообщения заголовок параметры
159         res = MessageBox((HWND)GetHWindow(), str, LoadStr(IDR_LIB),
```

```
            flags);
160
161         if (enable)                             /// если доступ к задаче был запрещен
162             ::EnableTaskAccess(1);      /// разрешаем доступ к задаче
163     }
164
165     return res;
166 }
167
168 /// <summary>
169 /// Вывод сообщения
170 /// </summary>
171 /// <param name="strId"></param>
172 /// <param name="flags"></param>
173 /// <returns></returns>
174 int LibMessage(int strId, int flags = MB_OK) {
175     return ::LibMessage(LoadStr(strId), flags);
176 }
177
178 /// <summary>
179 /// Определить имя библиотеки
180 /// </summary>
181 /// <returns></returns>
182 int WINAPI LIBRARYID() {
183     return IDR_LIB;
184 }
185
186 /// <summary>
187 /// Головная функция библиотеки
188 /// </summary>
189 /// <param name="comm"></param>
190 /// <returns></returns>
191 void WINAPI LIBRARYENTRY(unsigned int comm) {
192     // Выдача сообщения пользователю
193     ::Getapplication(); // Захват интерфейса приложения КОМПАС
194     if (application)
195     {
196         switch (comm)
197         {
198         case 1: // Создать закладку и подписаться
199             *menu_type = MenuType::type_profile;
200             ClosePropertyManager(0);
201             CreateAndSubscriptionPropertyManager();
202             break;
203         case 2: // Создать закладку и подписаться
204             *menu_type = MenuType::type_conditions;
205             ClosePropertyManager(0);
206             CreateAndSubscriptionPropertyManager();
207             break;
208         case 3: // Отписаться
209             ClosePropertyManager(1);
210             // Запоминаем положение панели и гасим ее
211             break;
212         }
213     }
```

```
214 }
215
216 /// <summary>
217 /// Провести подписку библиотеки на сообщения системы
218 /// </summary>
219 /// <param name="application">Интерфейс Компаса</param>
220 /// <returns>0</returns>
221 int WINAPI LibInterfaceNotifyEntry(IDispatch* application) {
222     application = application;
223     // CreateAndSubscriptionPropertyManager( false/*mes*/); // Создание ↵
        панели библиотеки
224     return 0;
225 }
226
227 /// <summary>
228 /// Использовать новое API
229 /// </summary>
230 /// <returns>1</returns>
231 int WINAPI LibIsOnApplication7() {
232     return 1;
233 }
234
235 /// <summary>
236 /// Получить интерфейс контейнера поверхностей
237 /// </summary>
238 /// <param name="doc">указатель на текущий документ</param>
239 /// <returns>указатель на интерфейс контейнера поверхностей</returns>
240 ISurfaceContainerPtr GetSurfaceContainer(IKompasDocumentPtr& doc) {
241     /// Контейнер поверхностей
242     ISurfaceContainerPtr surfCont = NULL;
243     if (doc) {
244         /// Документ 3Д
245         IKompasDocument3DPtr document3D(doc);
246         if (document3D) {
247             /// Верхний компонент
248             IPart7Ptr part7(document3D->GetTopPart());
249             if (part7) {
250                 surfCont = ISurfaceContainerPtr(part7);
251             }
252         }
253     }
254     return surfCont;
255 }
256
257 IDocumentFramePtr GetActiveFrame(IKompasDocumentPtr& doc) {
258     IDocumentFramePtr res = NULL;
259     if (doc) {
260         IDocumentFramesPtr frames(doc->DocumentFrames);
261         if (frames) {
262             IDocumentFramePtr next;
263             for (long i = 0, count = frames->Count; i < count; i++) {
264                 next = frames->Item[i];
265                 if ((bool)next && next->Active) {
266                     res = next;
267                     break;
268                 }
```

```
269                 }
270             }
271         }
272         return res;
273     }
274
275     /// <summary>
276     /// Создание пользовательской панели и подписка
277     /// </summary>
278     void CreateAndSubscriptionPropertyManager() {
279         if (propMng == NULL) {
280             NewApplicationEvent();
281             propMng = application->CreatePropertyManager(TRUE/*FALSE*/);
282             propMng->Layout = PropertyManagerLayout::pmAlignLeftInGroup;
283             propMng->Caption = _T("Панель СМП");
284             propMng->SpecToolbar =
                  SpecPropertyToolBarEnum::pnEnterEscCreateSaveSearchHelp;
285             // propMng->SetGabaritRect(g_Rect.left, g_Rect.top,
                  g_Rect.right, g_Rect.bottom);
286
287             /// Подписываемся на события процесса
288             new PropertyManagerEvent(propMng);
289             tab = propMng->PropertyTabs->Add(_T("Закладка СМП"));
290
291             /// Коллекция контролов
292             IPropertyControlsPtr collection = tab->PropertyControls;
293             reference docRef = ::ksGetCurrentDocument(0);
294             AdviseDoc(docRef);
295
296             if (*menu_type == MenuType::type_profile) {
297                 menu_profile->init(collection);
298             }
299             else if (*menu_type == MenuType::type_conditions) {
300                 menu_conditions->init(collection);
301             }
302             propMng->ShowTabs();
303             propMng->Caption = _T("Панель СМП");
304             propMng->UpdateTabs();
305         }
306         else {
307             /// Если панель была скрыта пользователем, по крестику
                  восстановим её
308             propMng->Visible = true;
309             if (*menu_type != MenuType::type_none) {
310                 LibMessage(_T("Панель уже загружена"));
311             }
312         }
313     }
314
315     /// <summary>
316     /// Построение объекта по параметрам меню
317     /// </summary>
318     void buildProfileObject() {
319         menu_profile->model->init(application, SMP_DLL.hModule);
320         if (menu_profile->updateMenuParameters()) {
321             menu_profile->model->createModel();
```

```
322     }
323 }
324
325 /// <summary>
326 /// Построение объекта по параметрам меню
327 /// </summary>
328 void buildConditionsObject() {
329     menu_profile->model->init(application, SMP_DLL.hModule);
330     if (menu_conditions->updateMenuParameters()) {
331         menu_conditions->model->createModel();
332         if (menu_conditions->findAngleEta()) {
333             propMng->UpdateTabs();
334         }
335     }
336 }
337
338 /// <summary>
339 /// Остановить работу с панелью свойств и отписаться
340 /// </summary>
341 /// <param name="mes"></param>
342 void ClosePropertyManager(int mes) {
343     if (propMng) {
344         // propMng->GetGabaritRect(&g_Rect.left, &g_Rect.top,
            &g_Rect.right, &g_Rect.bottom);
345         g_Layout = propMng->Layout;
346         ABaseEvent::TerminateEvents(DIID_ksPropertyManagerNotify);
347         propMng->HideTabs();
348         propMng = NULL;
349         // tab = NULL;
350     }
351     else {
352         if (mes == 1)
353             LibMessage(_T("Панель уже выгружена"));
354     }
355 }
356
357 /// <summary>
358 /// Обновление окна
359 /// </summary>
360 void UpdateSlideBox() {
361     if (slideBox)
362         slideBox->UpdateParam();
363 }
364
365 /// <summary>
366 /// Обновление окна
367 /// </summary>
368 void UpdateSlideBox(reference docRef) {
369     if (slideBox) {
370         slideBox->DrawingSlide = docRef;
371         slideBox->Value = (docRef != 0);
372         slideBox->CheckBoxVisibility = (docRef != 0);
373         slideBox->UpdateParam();
374     }
375 }
376
```

```
377 /// <summary>
378 /// Получить идентификаторы инструментальных и компактных панелей
379 /// </summary>
380 /// <param name="barType">Тип запрашиваемой панелей ( 0 - компактная
      панель, 1 - простая инструментальная панель )</param>
381 /// <param name="index">Индекс панели</param>
382 int WINAPI LibToolBarId(int barType, int index) {
383     if (!barType)
384         return !index ? 2000 : -1;
385     return -1;
386 }
387
388 /// <summary>
389 /// Подписка на события документа
390 /// </summary>
391 /// <param name="pDoc">Указатель на документ</param>
392 void AdviseDoc(long pDoc) {
393     if (!pDoc)
394         pDoc = ::ksGetCurrentDocument(0);
395     else
396         ::SetObjParam(pDoc, NULL, 0, DOCUMENT_STATE);
397
398     if (pDoc) {
399         int docType = ::ksGetDocumentType(pDoc);
400         if (NewDocumentEvent(pDoc)) {
401             switch (docType) {
402             case lt_DocSheetStandart:
403             case lt_DocSheetUser:
404                 NewStampEvent(pDoc);
405             case lt_DocFragment:
406                 NewDocument2DEvent(pDoc);
407                 NewObj2DEvent(pDoc, ALL_OBJ);
408                 break;
409             case lt_DocPart3D:
410             case lt_DocAssemble3D:
411                 NewDocument3DEvent(pDoc);
412                 NewObj3DEvent(pDoc, Obj3dType::o3d_unknown, NULL); ///
                    На все
413                 break;
414
415             case lt_DocSpc:
416             case lt_DocSpcUser:
417                 NewSpcDocEvent(pDoc);
418                 NewStampEvent(pDoc);
419                 break;
420             case lt_DocTxtStandart:
421             case lt_DocTxtUser:
422                 NewStampEvent(pDoc);
423                 break;
424             }
425
426             switch (docType) {
427             case lt_DocSheetStandart:
428             case lt_DocSheetUser:
```

```
429                 case lt_DocSpc:
430                 case lt_DocSpcUser:
431                 case lt_DocAssemble3D:
432                     NewSpecificationEvent(pDoc);
433                     NewSpcObjectEvent(pDoc, objType: SPC_BASE_OBJECT); /// На ↵
                        базовые объекты
434                     break;
435                 }
436             }
437         }
438 }
```