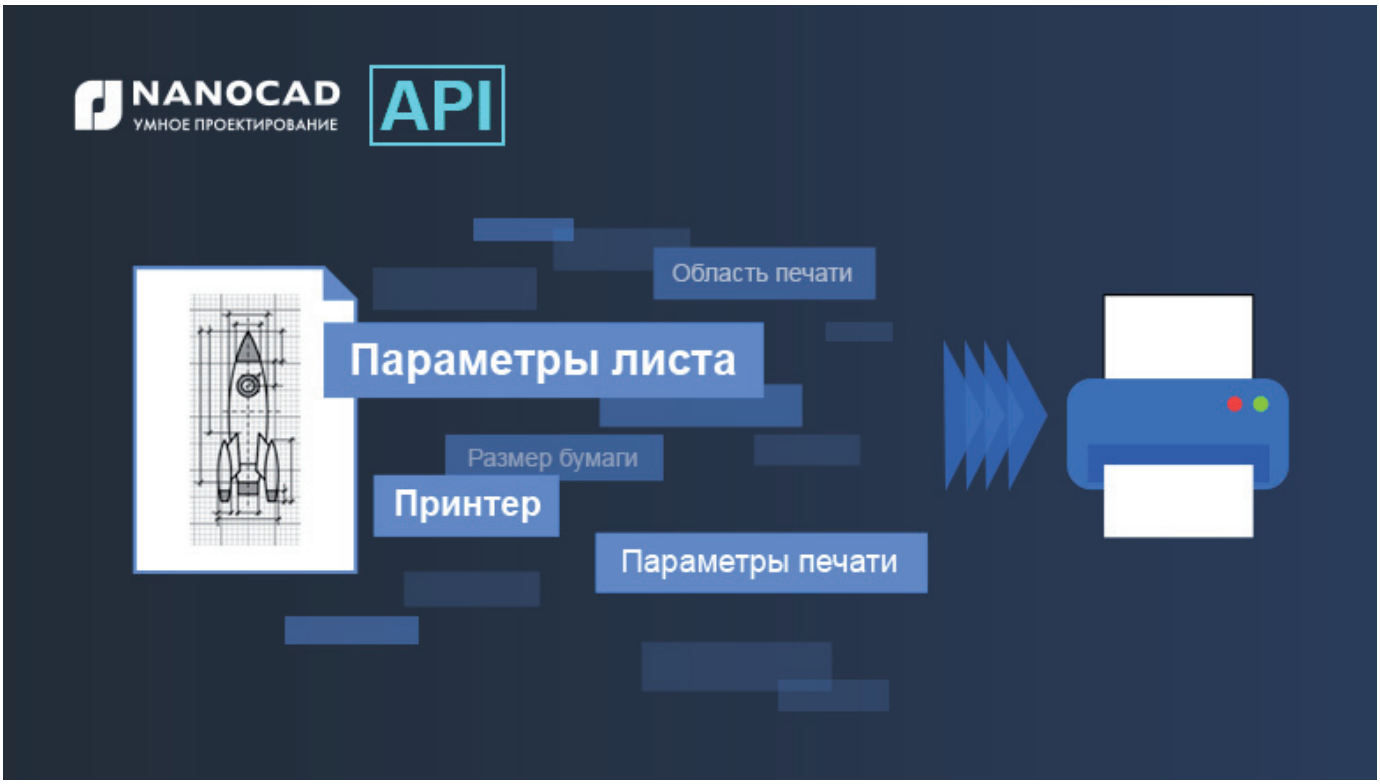

РАБОТА С НАБОРОМ ПАРАМЕТРОВ ЛИСТА ЧЕРЕЗ API nanoCAD

При подготовке чертежа к печати необходимо настраивать большое количество параметров: принтер, формат бумаги, масштаб, область печати и т.д.

В nanoCAD все необходимые для этого параметры объединены в наборы параметров листов. Однажды созданный набор можно применять в разных документах и разных листах, что позволит значительно сократить время подготовки документа к печати. В этой статье мы рассмотрим, как работать с наборами параметров листов через API nanoCAD.

Через пользовательский интерфейс nanoCAD наборами параметров листов можно управлять в окне *Диспетчер параметров листов* (рис. 1).

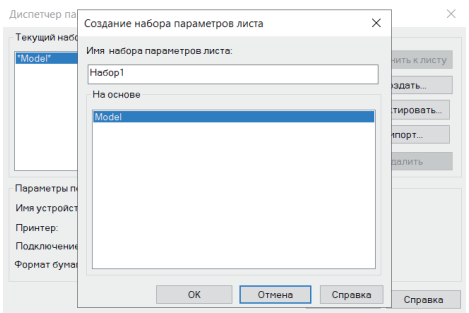

Рис. 1. Окно *Диспетчер параметров листов*

В API за наборы параметров листов отвечает свойство PlotConfigurations документа nanoCAD.

```
nanoCAD.Document comDoc = doc.AcadDocument as nanoCAD.Document;
comDoc.PlotConfigurations.Add("Набор1", true);
```

Чтобы научиться работать с наборами параметров листов через API nanoCAD, создадим приложение .NET(C#), которое:

- программно добавляет в документ новые наборы параметров листов;
- изменяет в них значения некоторых параметров;
- копирует набор параметров листов из другого документа nanoCAD;
- делает добавленные наборы параметров текущими для листов документа nanoCAD.

Разберем все этапы создания такого приложения.

Шаг 1. Создаем в Visual Studio проект типа "библиотека классов", подключаем к нему библиотеки hostmgd.dll, hostdbmgd.dll и ncauto.dll.

Шаг 2. Создаем класс, в котором будет находиться метод, создающий новый набор параметров листа. Далее речь пойдет о реализации этого метода.

```
    public partial class Commands
    {
        [Teigha.Runtime.CommandMethod("AddPlotConfiguration")]
        public void AddPlotConfiguration()
        {
        }
}
```

Шаг 3. Получаем доступ к документу nanoCAD и его настройкам:

```
// Получение ссылки на активный документ
HostMgd.ApplicationServices.Document doc =
    HostMgd.ApplicationServices.Application.DocumentManager.
    MdiActiveDocument;
nanoCAD.Document comDoc = doc.AcadDocument as nanoCAD.Document;
```

Шаг 4. Создаем новый набор параметров и добавляем его в документ:

```
// Добавление нового набора параметров листа к пространству модели документа
OdaX.IAcadPlotConfiguration plotConfig =
    comDoc.PlotConfigurations.Add("NewModelSpaceConfig", true);

// Добавление нового набора параметров печати к листам документа
OdaX.IAcadPlotConfiguration plotConfig1 =
    comDoc.PlotConfigurations.Add("NewLayoutConfig", false);
```

Здесь мы обращаемся к коллекции PlotConfigurations, которая упоминалась выше, и используем один из ее методов.
Метод Add() принимает два параметра:

1) Name — имя нового набора параметров в виде строки. Необходимо помнить, что nanoCAD не позволит создать наборы параметров листов для пространства модели и для листа с одинаковыми именами. Набор параметров листов, созданный для пространства модели, не будет доступен для листов, и наоборот;
2) ModelType — тип набора параметров листа в виде значения bool: true — набор параметров для пространства модели, false — набор параметров для листов.

В чем разница между набором параметров для пространств модели и листа? Фактически только в значении свойства ModelType. Но на практике есть и более глубокие различия (например, когда устанавливается область печати: для пространства модели существует свой набор возможных областей печати, для листа — свой).
Также стоит обратить внимание, что в коллекции PlotConfigurations наборы параметров расположены в алфавитном порядке по имени набора.
Кроме метода Add(), в коллекции PlotConfigurations доступны другие методы, позволяющие:

- обращаться к конкретному набору параметров через метод Item(), который в качестве параметра принимает порядковый номер набора в коллекции: comDoc.PlotConfigurations.Item(0);
- удалить набор параметров из коллекции. Для этого нужно сначала получить к нему доступ методом Item(), а затем воспользоваться методом Delete(): comDoc.PlotConfigurations.Item(0).Delete();
- очистить коллекцию PlotConfigurations методом Delete (): comDoc.PlotConfigurations.Delete().

Шаг 5. Изменяем значения некоторых параметров в первом созданном наборе параметров:

```
// Добавление принтера в созданный набор параметров печати
plotConfig.ConfigName = plotConfig.GetPlotDeviceNames()[1];

// Получение списка форматов бумаги, доступных для выбранного принтера, и
// добавление формата бумаги в созданный набор параметров печати
plotConfig.CanonicalMediaName =
    plotConfig.GetCanonicalMediaNames()[1];
```

Здесь вы можете видеть пример получения перечня принтеров и форматов бумаги, доступных для документа, а также изменения значений параметров, отвечающих за принтер и формат бумаги:

- метод GetPlotDeviceNames() возвращает массив строк с названиями принтеров;
- метод GetCanonicalMediaNames() возвращает массив строк с названиями форматов бумаги;
- свойство ConfigName хранит в себе название принтера;
- свойство CanonicalMediaName хранит в себе название формата бумаги.

У читателя наверняка возникнет вопрос, какие еще параметры можно настроить и как это сделать. Для справки пробежимся по основным параметрам набора, которые мы можем менять через API в параллели с настройкой тех же параметров через интерфейс (рис. 2).

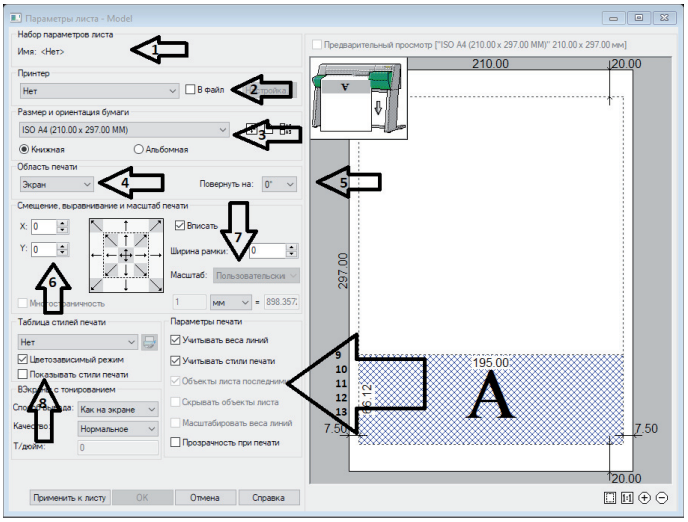

Рис. 2. Окно *Параметры листа*

1. **Имя набора параметров листа:** задается при создании набора в методе Add(). Впоследствии изменить это имя через интерфейс не получится, а поменять его через API можно: plotConfig.Name = "NewName".
2. **Принтер:** задается свойством plotConfig.ConfigName. Название принтера для этого свойства получаем методом plotConfig.GetPlotDeviceNames(), который возвращает массив строк.
3. **Формат бумаги:** задается свойством plotConfig.CanonicalMediaName. Наименование формата бумаги для этого свойства получаем методом plotConfig.GetCanonicalMediaNames(), который возвращает массив строк.

4. **Область печати:** задается свойством plotConfig.PlotType. Область печати для этого свойства берем из перечисления OdaX.AcPlotType.
5. **Ориентация чертежа на листе:** задается свойством plotConfig.PlotRotation. Угол поворота для этого свойства берем из перечисления OdaX.AcPlotRotation.
6. **Положение левого нижнего угла чертежа относительно левого нижнего угла границы листа:** задается свойством plotConfig.PlotOrigin. В свойство нужно передать массив double с координатами X и Y.
7. **Масштаб печати чертежа:** задается свойством plotConfig.StandardScale. Значение масштаба для свойства берем из перечисления OdaX.AcPlotScale. Также можно установить пользовательский масштаб методом plotConfig.SetCustomScale(double Numerator, double Denominator).
8. **Стиль печати:** задается свойством plotConfig.StyleSheet. Стиль печати для свойства берем из массива строк, возвращаемого методом plotConfig.GetPlotStyleTableNames().
9. **"Учитывать веса линий":** задается свойством plotConfig.PlotWithLineweights. Переменная логического типа данных: при значении true веса линий будут учитываться, при значении false – нет.
10. **"Учитывать стили печати":** задается свойством plotConfig.PlotWithPlotStyles. Переменная логического типа данных: при значении true назначенный стиль печати будет учтен, при значении false – нет.
11. **"Объекты листа последними":** задается свойством plotConfig.PlotViewportsFirst. Переменная логического типа данных: при значении true объекты листа будут отображены на заднем фоне, при значении false – нет.
12. **"Скрывать объекты листа":** задается свойством plotConfig.PlotHidden. Переменная логического типа данных: при значении true объекты листа будут скрыты, при значении false – нет.
13. **"Масштабировать веса линий":** задается свойством plotConfig.ScaleLineweights. Переменная логического типа данных: при значении true веса линий будут отмасштабированы, при значении false – нет.

Последние три пункта списка предназначены только для листов – в пространстве модели изменение этих свойств через API не будет влиять на конечный результат, а через интерфейс в настройках параметров печати пространства модели их изменить невозможно.

Шаг 6. Копируем набор параметров листов из другого документа nanoCAD во второй созданный набор "NewLayoutConfig".

В приведенном примере копируется набор параметров листов с названием "LayoutPlotConfig" из документа nanoCAD *nanoCAD_configs.dwg*, расположенного в директории *C:\\Work*.

В копируемом наборе задано два параметра:

- Принтер: Microsoft XPS Document Writer;
- Формат бумаги: Конверт №14, ориентация Книжная.

```
// Получение ссылки на документ, в котором сохранены нужные нам наборы
// параметров листов
HostMgd.ApplicationServices.Document docToCopy =
        HostMgd.ApplicationServices.Application.DocumentManager.
            Open("C:\\Work\\nanoCAD_configs.dwg", true);
nanoCAD.Document comDocToCopy = docToCopy.AcadDocument as nanoCAD.Document;

// Получение ссылки на набор параметров листа, который нужно скопировать
OdaX.IAcadPlotConfiguration configToCopy =
            comDocToCopy.PlotConfigurations.Item(0);

// Копируем набор параметров листа из одного документа в другой
plotConfig1.CopyFrom((OdaX.AcadPlotConfiguration)configToCopy);

// Закрываем документ, из которого копировали набор параметров листа
docToCopy.CloseAndDiscard();
```

Здесь основное действие выполняет метод CopyFrom(), который принимает один параметр типа OdaX.AcadPlotConfiguration – набор параметров листов.

И это был последний пункт из перечня функций, которые должно выполнять наше приложение. Далее приведен полный текст программы .NET, которая в nanoCAD будет представлять собой команду *AddPlotConfiguration*.

```
public partial class Commands
{
    [Teigha.Runtime.CommandMethod("AddPlotConfiguration")]
    public void AddPlotConfiguration()
    {

        // Получение ссылки на активный документ
        HostMgd.ApplicationServices.Document doc = HostMgd.ApplicationServices.Application.DocumentManager.MdiActiveDocument;
        nanoCAD.Document comDoc = doc.AcadDocument as nanoCAD.Document;

        // Получение ссылки на редактор активного документа
        HostMgd.EditorInput.Editor ed = doc.Editor;

        // Добавление нового набора параметров печати к пространству модели документа
        OdaX.IAcadPlotConfiguration plotConfig =
        comDoc.PlotConfigurations.Add("NewModelSpaceConfig", true);

        // Добавление нового набора параметров печати к листам документа
        OdaX.IAcadPlotConfiguration plotConfig1 =
            comDoc.PlotConfigurations.Add("NewLayoutConfig", false);

        // Вывод в консоль nanoCAD наименований наборов параметров печати документа
        ed.WriteMessage("Наборы параметров листов документа '{0}' до изменения их параметров:", doc.Name);
        foreach (var pc in comDoc.PlotConfigurations)
        {
            OdaX.IAcadPlotConfiguration config = pc as OdaX.IAcadPlotConfiguration;
            ed.WriteMessage("{0}:{1},{2},{3}", config.Name, config.ConfigName, config.CanonicalMediaName,config.PlotType);
        }
        ed.WriteMessage("");

        // Добавление принтера в созданный набор параметров печати
        plotConfig.ConfigName = plotConfig.GetPlotDeviceNames()[1];
```

```
        // Получение списка форматов бумаги, доступных для выбранного принтера, и
        // добавление формата бумаги в созданный набор параметров печати
        plotConfig.CanonicalMediaName =
            plotConfig.GetCanonicalMediaNames()[1];

        // Получение ссылки на документ, в котором сохранены нужные нам наборы
        // параметров листов
        HostMgd.ApplicationServices.Document docToCopy =
            HostMgd.ApplicationServices.Application.DocumentManager.
            Open("C:\\Work\\nanoCAD_configs.dwg", true);
        nanoCAD.Document comDocToCopy =
            docToCopy.AcadDocument as nanoCAD.Document;

        // Получение ссылки на набор параметров листа, который нужно скопировать
        OdaX.IAcadPlotConfiguration configToCopy =
            comDocToCopy.PlotConfigurations.Item(0);

        // Копируем набор параметров листа из одного документа в другой
        plotConfig1.CopyFrom((OdaX.AcadPlotConfiguration)configToCopy);

        // Закрываем документ, из которого копировали набор параметров листа
        docToCopy.CloseAndDiscard();

        // Вывод в консоль nanoCAD наименований наборов параметров печати документа
        ed.WriteMessage("Наборы параметров листов документа '{0}' после изменения их параметров:", doc.Name);
        foreach (var pc in comDoc.PlotConfigurations)
        {
            OdaX.IAcadPlotConfiguration config =
              pc as OdaX.IAcadPlotConfiguration;
            ed.WriteMessage("{0}:{1},{2}",
              config.Name, config.ConfigName, config.
CanonicalMediaName);
        }
    }
}
```

Шаг 7. Компилируем наше приложение и загружаем в nanoCAD. Теперь после запуска команды *AddPlotConfiguration* мы увидим в консоли сообщения, показанные на рис. 3.

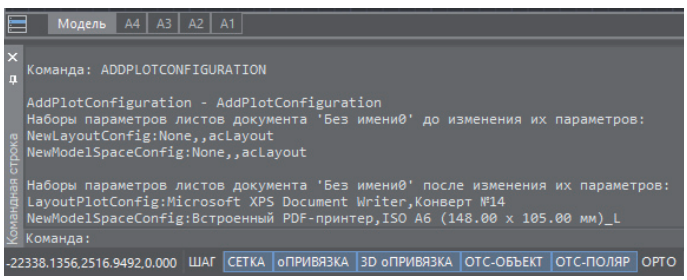

Рис. 3. Сообщения команды *AddPlotConfiguration*

Первый раз содержимое коллекции PlotConfigurations было выведено в консоль сразу после добавления в нее новых элементов.

Во второй раз содержимое коллекции выведено в консоль после изменения параметров в первой добавленной конфигурации и копирования во вторую добавленную конфигурацию значений параметров из другого документа. Следует обратить внимание, что методом CopyFrom() скопировалось также и название набора параметров.

Ради интереса можно проверить через пользовательский интерфейс nanoCAD изменения в окне *Диспетчер параметров листов*. Картина должна быть такой, как показано на рис. 4 и 5.

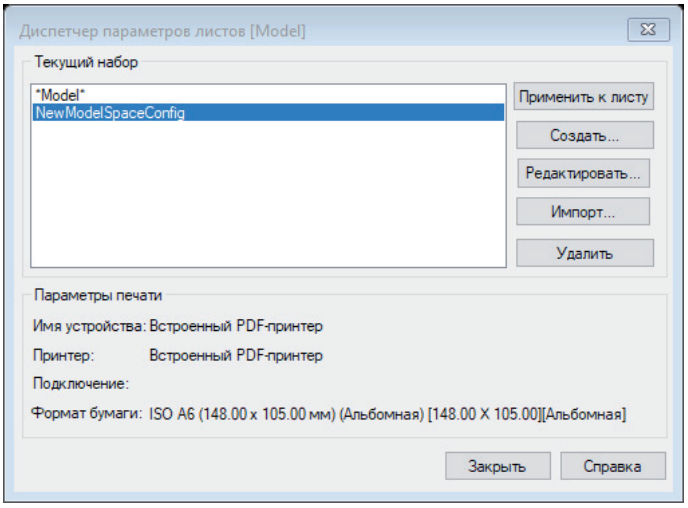

Рис. 4. Окно *Диспетчер параметров листов* для пространства модели

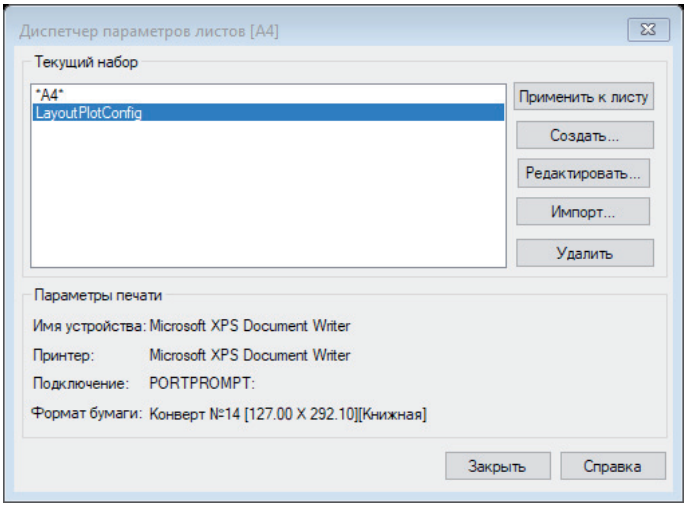

Рис. 5. Окно *Диспетчер параметров листов* для листа A4

Итак, в этой статье мы подробно разобрали процесс создания .NET-приложения, которое добавляет наборы параметров листов в документ nanoCAD, меняет в них параметры и копирует значения параметров из другого документа nanoCAD. Попутно был приведен пример получения перечня доступных принтеров и форматов бумаги. В следующей статье разберемся, как работать со стилями печати через API nanoCAD.

Светлана Мирончик,
Клуб разработчиков nanoCAD
ООО "Нанософт разработка"