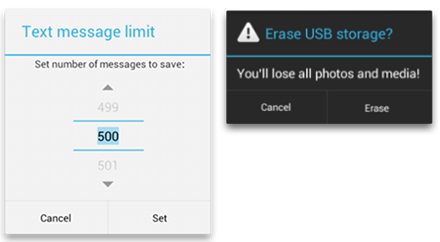
ДИАЛОГОВЫЕ ОКНА

Диалоговое окно — это небольшое окно, которое предлагает пользователю принять решение или ввести дополнительную информацию. Диалоговое окно не заполняет весь экран и, как правило, используется при модальных событиях, для продолжения которых требуется действие пользователя.

**Дизайн диалогового окна**

Подробную информацию о дизайне диалоговых окон, включая рекомендации для вашего языка см. в [Руководстве](https://developer.android.com/design/building-blocks/dialogs.html) по дизайну диалоговых окон.



Класс [Dialog](https://developer.android.com/reference/android/app/Dialog.html) — это базовый класс для создания диалоговых окон, но реализовывать напрямую класс [Dialog](https://developer.android.com/reference/android/app/Dialog.html) не рекомендуется. Вместо этого следует использовать один из следующих подклассов:

[AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html)

Диалоговое окно, в котором могут отображаться заголовок, кнопки вплоть до трех штук, список из выбираемых элементов либо пользовательский макет.

[DatePickerDialog](https://developer.android.com/reference/android/app/DatePickerDialog.html) или [TimePickerDialog](https://developer.android.com/reference/android/app/TimePickerDialog.html)

Диалоговое окно с предопределенным пользовательским интерфейсом, с помощью которого пользователь указывает значения даты или времени.

**Следует избегать ProgressDialog**

В Android имеется другой класс диалоговых окон под названием [ProgressDialog](https://developer.android.com/reference/android/app/ProgressDialog.html), которые отображают диалоговое окно с индикатором выполнения процесса. Тем не менее, если необходимо отобразить прогресс загрузки или неопределенный процес, рекомендуется следовать инструкциям по дизайну[Выполнение & Операции](https://developer.android.com/design/building-blocks/progress.html), чтобы использовать [ProgressBar](https://developer.android.com/reference/android/widget/ProgressBar.html) в вашем макете.

Эти классы определяют стиль и структуру вашего диалогового окна, однако следует использовать [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) в качестве контейнера вашего диалогового окна. Класс [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html)предоставляет все функции, необходимые для создания диалогового окна и управления его внешним видом, вместо вызова методов к объекту[Dialog](https://developer.android.com/reference/android/app/Dialog.html).

Использование [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) для управления диалоговым окном обеспечивает корректную обработку событий жизненного цикла , таких как нажатие пользователем кнопки *Назад* или поворот экрана. С помощью класса [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) также происходит повторное использование пользовательского интерфейса диалогового окна в качестве встраиваемого компонента в пользовательский интерфейс более высокого уровня — подобно традиционному классу [Fragment](https://developer.android.com/reference/android/support/v4/app/Fragment.html)(например, когда необходимо различное отображение пользовательского интерфеса диалогового окна на больших и маленьких экранах).

В следующих разделах руководства описано использование [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) в сочетании с объектом [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html) . Если необходимо создать элемент выбора даты или времени, обратитесь к руководству [Элементы выбора](https://developer.android.com/guide/topics/ui/controls/pickers.html).

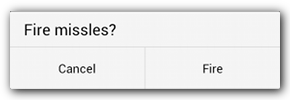
**Примечание:** Поскольку класс [DialogFragment](https://developer.android.com/reference/android/app/DialogFragment.html) изначально включен в Android 3.0 (уровень API 11), в настоящем документе описывается использование класса [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html), предоставляемого в [Библиотеке поддержки](https://developer.android.com/tools/support-library/index.html). После добавления этой библиотеки в приложение появится возможность использовать[DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) и множество других API на устройствах, работающих на Android 1.6 и выше. Если минимальная версия вашего приложения поддерживает уровень API 11 и выше, можно использовать фреймворк-версию[DialogFragment](https://developer.android.com/reference/android/app/DialogFragment.html), но следует иметь в виду, что данный документ ссылается на API со вспомогательными библиотеками. При использовании вспомогательной библиотеки необходимо импортировать класс android.support.v4.app.DialogFragment , а *не*android.app.DialogFragment.

Создание фрагмента диалогового окна

Вы можете реализовать широкие возможности дизайна диалоговых окон, включая создание пользовательских макетов, а также пути, описанные в руководстве по дизайну [Диалоговых окон](https://developer.android.com/design/building-blocks/dialogs.html) —путем расширения [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) и создания[AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html) в методе обратного вызова [onCreateDialog()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "onCreateDialog(android.os.Bundle)).

Например, имеется базовый класс [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html), управляемый в рамках [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html):

public class FireMissilesDialogFragment extends DialogFragment {  
    @Override  
    public Dialog onCreateDialog(Bundle savedInstanceState) {  
        // Use the Builder class for convenient dialog construction  
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
        builder.setMessage(R.string.dialog\_fire\_missiles)  
               .setPositiveButton(R.string.fire, new DialogInterface.OnClickListener() {  
                   public void onClick(DialogInterface dialog, int id) {  
                       // FIRE ZE MISSILES!  
                   }  
               })  
               .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {  
                   public void onClick(DialogInterface dialog, int id) {  
                       // User cancelled the dialog  
                   }  
               });  
        // Create the AlertDialog object and return it  
        return builder.create();  
    }  
}



**Рисунок 1.** Диалоговое окно с сообщением и двумя кнопками действия.

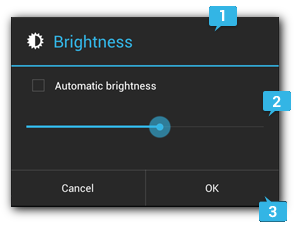
Итак, после создания экземпляра этого класса и вызова [show()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "show(android.support.v4.app.FragmentManager, java.lang.String)) к этому объекту появляется диалоговое окно, показанное на рисунке 1.

В следующем разделе предоставлена более подробная информация об использовании API [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html) для создания диалоговых окон.

В зависимости от сложности создаваемого диалогового окна возможно реализовать множество других методов обратного вызова в [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html), включая все базовые[методы жизненных циклов фрагментов](https://developer.android.com/guide/components/fragments.html#Lifecycle).

Создание диалогового окна оповещения

С помощью класса [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html) создается многообразие решений касательно внешнего вида диалогового окна , и зачастую этого класса вполне достаточно. Как показано на рис. 2, диалоговое окно состоит из трех областей:



**Рисунок 2.** Макет диалогового окна.

1. **Заголовок**

Это дополнительная возможность, которая используется только в случае, если область содержимого занята подробным сообщением, списком или пользовательским макетом. Если необходимо отобразить простое сообщение или вопрос (как, например, в диалоге на рисунке 1), заголовок не нужен.

1. **Область содержимого**

Здесь может отображаться сообщение, список или другой пользовательский макет.

1. **Кнопки действия**

В диалоговом окне не должно содержаться более трех кнопок действия.

Класс [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html) предоставляет API, с помощью которых можно создавать [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html) с этими видами содержимого, включая пользовательский макет.

Создание [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html):

**// 1. Instantiate an** [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html) **with its constructor**  
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
  
**// 2. Chain together various setter methods to set the dialog characteristics**  
builder.setMessage(R.string.dialog\_message)  
       .setTitle(R.string.dialog\_title);  
  
**// 3. Get the** [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html) **from** [create()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html#create())  
AlertDialog dialog = builder.create();

В следующих главах показано, как определять различные атрибуты диалоговых окон с помощью класса [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html).

Добавление кнопок

Для добавления кнопок, изображенных на рисунке 2, вызывайте методы [setPositiveButton()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setPositiveButton(int, android.content.DialogInterface.OnClickListener)) и [setNegativeButton()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setNegativeButton(int, android.content.DialogInterface.OnClickListener)):

AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
// Add the buttons  
builder.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {  
           public void onClick(DialogInterface dialog, int id) {  
               // User clicked OK button  
           }  
       });  
builder.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {  
           public void onClick(DialogInterface dialog, int id) {  
               // User cancelled the dialog  
           }  
       });  
// Set other dialog properties  
...  
  
// Create the AlertDialog  
AlertDialog dialog = builder.create();

Методы set...Button() предполагают заголовок для кнопки (реализуемый через [строковый ресурс](https://developer.android.com/guide/topics/resources/string-resource.html)) и [DialogInterface.OnClickListener](https://developer.android.com/reference/android/content/DialogInterface.OnClickListener.html), который определяет действие, следующее за нажатием кнопки пользователем.

Реализована возможность добавлять три различных вида кнопок действий:

Положительные

Используются для подтверждения и продолжения дейстия (кнопка «ОК»).

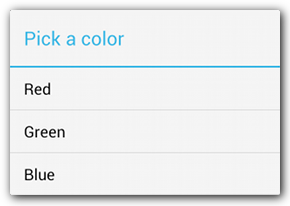
Отрицательные

Используются для отмены действия.

Нейтральные

Используются в случаях, когда пользователь может не желать продолжить действие, но при этом необязательно хочет его отменить. Появляется между положительными и отрицательнымиI кнопками. Примером такого действия может быть «Напомнить позже».

Можно добавлять только одну кнопку каждого вида в [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html). Это означает, что нельзя использовать более одной «положительной» кнопки.



**Рисунок 3.** Диалоговое окно с заголовком и списком.

Добавление списка

В API [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html) реализована возможность использования трех видов списков:

* Традиционный список с выбором одного варианта
* Интерактивный список с выбором одного варианта (переключатели)
* Интерактивный список с выбором нескольких вариантов (флажки)

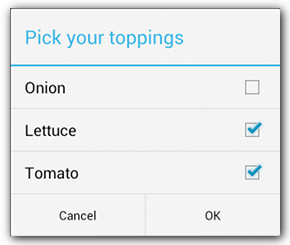
Для создания списка с выбором одного варианта, как на рисунке 3, используйте метод[setItems()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html#setItems(int, android.content.DialogInterface.OnClickListener)):

@Override  
public Dialog onCreateDialog(Bundle savedInstanceState) {  
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
    builder.setTitle(R.string.pick\_color)  
           .setItems(R.array.colors\_array, new DialogInterface.OnClickListener() {  
               public void onClick(DialogInterface dialog, int which) {  
               // The 'which' argument contains the index position  
               // of the selected item  
           }  
    });  
    return builder.create();  
}

Поскольку список отображается в области содержимого диалогового окна, диалоговое окно не может показать одновременно сообщение и список, поэтому необходимо задать заголовок диалогового окна с помощью [setTitle()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setTitle(int)). Для указания элементов списка необходимо вызвать [setItems()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setItems(int, android.content.DialogInterface.OnClickListener)), передающий указатель. В качестве другого варианта можно указать список с помощью [setAdapter()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setAdapter(android.widget.ListAdapter, android.content.DialogInterface.OnClickListener)). Наполнение списка динамическими данными (например, из базы данных) происходит с помощью [ListAdapter](https://developer.android.com/reference/android/widget/ListAdapter.html).

Если вы предпочтете реализовать список с помощью [ListAdapter](https://developer.android.com/reference/android/widget/ListAdapter.html), рекомендуется использовать [Loader](https://developer.android.com/reference/android/support/v4/content/Loader.html), чтобы содержимое загружалось асинхронно. Подробно этот процесс описан далее в руководстве по [Созданию макетов с помощью адаптера](https://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews) и [загрузчиков](https://developer.android.com/guide/components/loaders.html) .

**Примечание:** По умолчанию нажатие по элементу списка отменяет диалоговое окно, за исключением случаев, когда используется один из следующих интерактивных списков.



**Рисунок 4.** Список с несколькими вариантами ответов.

Добавление интерактивного списка с одним или несколькими вариантами ответов

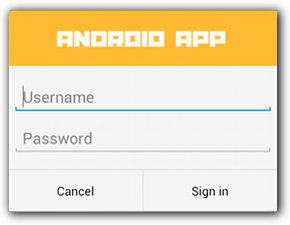
Для добавления списка с несколькими вариантами ответов (флажки) или списка с одним вариантом ответа (переключатели) используйте методы [setMultiChoiceItems()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html#setMultiChoiceItems(android.database.Cursor, java.lang.String, java.lang.String, android.content.DialogInterface.OnMultiChoiceClickListener))или [setSingleChoiceItems()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html#setSingleChoiceItems(int, int, android.content.DialogInterface.OnClickListener)) соответственно.

Например, таким образом можно создать список с несколькими вариантами ответов, как на рисунке 4, который сохраняет выбранные элементы в [ArrayList](https://developer.android.com/reference/java/util/ArrayList.html):

@Override  
public Dialog onCreateDialog(Bundle savedInstanceState) {  
    mSelectedItems = new ArrayList();  // Where we track the selected items  
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
    // Set the dialog title  
    builder.setTitle(R.string.pick\_toppings)  
    // Specify the list array, the items to be selected by default (null for none),  
    // and the listener through which to receive callbacks when items are selected  
           .setMultiChoiceItems(R.array.toppings, null,  
                      new DialogInterface.OnMultiChoiceClickListener() {  
               @Override  
               public void onClick(DialogInterface dialog, int which,  
                       boolean isChecked) {  
                   if (isChecked) {  
                       // If the user checked the item, add it to the selected items  
                       mSelectedItems.add(which);  
                   } else if (mSelectedItems.contains(which)) {  
                       // Else, if the item is already in the array, remove it   
                       mSelectedItems.remove(Integer.valueOf(which));  
                   }  
               }  
           })  
    // Set the action buttons  
           .setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {  
               @Override  
               public void onClick(DialogInterface dialog, int id) {  
                   // User clicked OK, so save the mSelectedItems results somewhere  
                   // or return them to the component that opened the dialog  
                   ...  
               }  
           })  
           .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {  
               @Override  
               public void onClick(DialogInterface dialog, int id) {  
                   ...  
               }  
           });  
  
    return builder.create();  
}

Несмотря на то, что и традиционный список, и список с переключателями предполагают действие по выбору одного элемента, вам необходимо использовать [setSingleChoiceItems()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setSingleChoiceItems(int, int, android.content.DialogInterface.OnClickListener)), чтобы сохранить выбор пользователя. Это значит, что при повторном открытии диалогового окна будет отображаться текущий выбор пользователя, а затем создается список с переключателями.

Создание пользовательского макета



**Рисунок 5**. Пользовательский макет диалогового окна.

Если в диалоговом окне необходим пользовательский макет, нужно создать макет и добавить его в [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html) путем вызова [setView()](https://developer.android.com/reference/android/app/AlertDialog.Builder.html" \l "setView(android.view.View)) в объекте [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html).

По умолчанию пользовательский мает заполняет окно диалога, при это все равно можно использовать методы [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html) для добавления кнопок и заголовка.

В качестве примера на рисунке 5 приведен файл макета для диалогового окна.

res/layout/dialog\_signin.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="wrap\_content"  
    android:layout\_height="wrap\_content">  
    <ImageView  
        android:src="@drawable/header\_logo"  
        android:layout\_width="match\_parent"  
        android:layout\_height="64dp"  
        android:scaleType="center"  
        android:background="#FFFFBB33"  
        android:contentDescription="@string/app\_name" />  
    <EditText  
        android:id="@+id/username"  
        android:inputType="textEmailAddress"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginTop="16dp"  
        android:layout\_marginLeft="4dp"  
        android:layout\_marginRight="4dp"  
        android:layout\_marginBottom="4dp"  
        android:hint="@string/username" />  
    <EditText  
        android:id="@+id/password"  
        android:inputType="textPassword"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginTop="4dp"  
        android:layout\_marginLeft="4dp"  
        android:layout\_marginRight="4dp"  
        android:layout\_marginBottom="16dp"  
        android:fontFamily="sans-serif"  
        android:hint="@string/password"/>  
</LinearLayout>

**Совет.** По умолчанию при настройке элемента [EditText](https://developer.android.com/reference/android/widget/EditText.html) для типа ввода "textPassword" используется семейство шрифтов фиксированной ширины, поэтому необходимо изменить семейство шрифтов на"sans-serif", чтобы в обоих текстовых полях использовались одинаковые стили шрифта.

Для применения макета в вашем [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) вам понадобится [LayoutInflater](https://developer.android.com/reference/android/view/LayoutInflater.html) с [getLayoutInflater()](https://developer.android.com/reference/android/app/Activity.html" \l "getLayoutInflater()) и вызов [inflate()](https://developer.android.com/reference/android/view/LayoutInflater.html" \l "inflate(int, android.view.ViewGroup)), где первым параметром будет являться ID ресурса макета, а вторым параметром — исходный вид макета. Затем можно вызвать[setView()](https://developer.android.com/reference/android/app/AlertDialog.html#setView(android.view.View)) для размещения макета в диалоговом окне.

@Override  
public Dialog onCreateDialog(Bundle savedInstanceState) {  
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
    // Get the layout inflater  
    LayoutInflater inflater = getActivity().getLayoutInflater();  
  
    // Inflate and set the layout for the dialog  
    // Pass null as the parent view because its going in the dialog layout  
    builder.setView(inflater.inflate(R.layout.dialog\_signin, null))  
    // Add action buttons  
           .setPositiveButton(R.string.signin, new DialogInterface.OnClickListener() {  
               @Override  
               public void onClick(DialogInterface dialog, int id) {  
                   // sign in the user ...  
               }  
           })  
           .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {  
               public void onClick(DialogInterface dialog, int id) {  
                   LoginDialogFragment.this.getDialog().cancel();  
               }  
           });        
    return builder.create();  
}

**Совет.** Если необходимо пользовательское диалоговое окно, можно отображать [Activity](https://developer.android.com/reference/android/app/Activity.html) в качестве диалога вместо API [Dialog](https://developer.android.com/reference/android/app/Dialog.html). Нужно создать операцию и установить тему [Theme.Holo.Dialog](https://developer.android.com/reference/android/R.style.html" \l "Theme_Holo_Dialog) в элементе манифеста[<операция>](https://developer.android.com/guide/topics/manifest/activity-element.html):

<activity android:theme="@android:style/Theme.Holo.Dialog" >

Готово. Операция теперь отображается в диалоговом окне, а не в полноэкранном режиме.

Передача событий обратно в основное диалоговое приложение

Когда пользователь нажимает одну из кнопок действий в диалоговом окне, либо выбирает элемент из списка, [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) может самостоятельно произвести необходимое действие, однако зачастую вам может понадобиться доставить информацию о событии операции или фрагменту, которые открыли диалоговое окно. Для этого нобходимо определить интерфейс метода для каждого типа события нажатия. Затем этот интерфейс применяется в основном компоненте приложения, которое получает информацию о событиях из диалогового окна.

Например, [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) определяет интерфейс, по который доставляет события обратно в основной компонент операции:

public class NoticeDialogFragment extends DialogFragment {  
      
    /\* The activity that creates an instance of this dialog fragment must  
     \* implement this interface in order to receive event callbacks.  
     \* Each method passes the DialogFragment in case the host needs to query it. \*/  
    public interface NoticeDialogListener {  
        public void onDialogPositiveClick(DialogFragment dialog);  
        public void onDialogNegativeClick(DialogFragment dialog);  
    }  
      
    // Use this instance of the interface to deliver action events  
    NoticeDialogListener mListener;  
      
    // Override the Fragment.onAttach() method to instantiate the NoticeDialogListener  
    @Override  
    public void onAttach(Activity activity) {  
        super.onAttach(activity);  
        // Verify that the host activity implements the callback interface  
        try {  
            // Instantiate the NoticeDialogListener so we can send events to the host  
            mListener = (NoticeDialogListener) activity;  
        } catch (ClassCastException e) {  
            // The activity doesn't implement the interface, throw exception  
            throw new ClassCastException(activity.toString()  
                    + " must implement NoticeDialogListener");  
        }  
    }  
    ...  
}

Операция, выполняемая диалоговым окном, создает экземпляр диалогового окна с помощью конструктора фрагментов диалогового окна и получает события диалога с помощью реализации интерфейса NoticeDialogListener:

public class MainActivity extends FragmentActivity  
                          implements NoticeDialogFragment.NoticeDialogListener{  
    ...  
      
    public void showNoticeDialog() {  
        // Create an instance of the dialog fragment and show it  
        DialogFragment dialog = new NoticeDialogFragment();  
        dialog.show(getSupportFragmentManager(), "NoticeDialogFragment");  
    }  
  
    // The dialog fragment receives a reference to this Activity through the  
    // Fragment.onAttach() callback, which it uses to call the following methods  
    // defined by the NoticeDialogFragment.NoticeDialogListener interface  
    @Override  
    public void onDialogPositiveClick(DialogFragment dialog) {  
        // User touched the dialog's positive button  
        ...  
    }  
  
    @Override  
    public void onDialogNegativeClick(DialogFragment dialog) {  
        // User touched the dialog's negative button  
        ...  
    }  
}

Поскольку выполняемая операция реализуется через NoticeDialogListener с помощью метода обратного вызова [onAttach()](https://developer.android.com/reference/android/support/v4/app/Fragment.html" \l "onAttach(android.app.Activity)), во фрагменте диалога могут использоваться методы обратного вызова интерфейса для доставки событий нажатий к операциям:

public class NoticeDialogFragment extends DialogFragment {  
    ...  
  
    @Override  
    public Dialog onCreateDialog(Bundle savedInstanceState) {  
        // Build the dialog and set up the button click handlers  
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
        builder.setMessage(R.string.dialog\_fire\_missiles)  
               .setPositiveButton(R.string.fire, new DialogInterface.OnClickListener() {  
                   public void onClick(DialogInterface dialog, int id) {  
                       // Send the positive button event back to the host activity  
                       mListener.onDialogPositiveClick(NoticeDialogFragment.this);  
                   }  
               })  
               .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {  
                   public void onClick(DialogInterface dialog, int id) {  
                       // Send the negative button event back to the host activity  
                       mListener.onDialogNegativeClick(NoticeDialogFragment.this);  
                   }  
               });  
        return builder.create();  
    }  
}

Отображение диалогового окна

Для отображения диалогового окна необходимо создать экземпляр [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) и вызвать [show()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "show(android.support.v4.app.FragmentManager, java.lang.String)), передавая [FragmentManager](https://developer.android.com/reference/android/support/v4/app/FragmentManager.html) и наименование тега для фрагмента диалога.

Можно получить [FragmentManager](https://developer.android.com/reference/android/support/v4/app/FragmentManager.html) путем вызова [getSupportFragmentManager()](https://developer.android.com/reference/android/support/v4/app/FragmentActivity.html#getSupportFragmentManager()) из [FragmentActivity](https://developer.android.com/reference/android/support/v4/app/FragmentActivity.html) или [getFragmentManager()](https://developer.android.com/reference/android/support/v4/app/Fragment.html#getFragmentManager()) из [Fragment](https://developer.android.com/reference/android/support/v4/app/Fragment.html). Пример:

public void confirmFireMissiles() {  
    DialogFragment newFragment = new FireMissilesDialogFragment();  
    newFragment.show(getSupportFragmentManager(), "missiles");  
}

Второй аргумент, "missiles", — это уникальное наименование тега, которое система использует для сохранения и восстановления состояния фрагмента, когда это необходимо. С помощью этого тега также можно управлять фрагментом путем вызова[findFragmentByTag()](https://developer.android.com/reference/android/support/v4/app/FragmentManager.html#findFragmentByTag(java.lang.String)).

Отображение диалогового окна в полноэкранном режиме или в виде встроенного фрагмента

Вам может понадобиться макет пользовательского интерфейса, в котором в некоторых ситуациях часть пользовательского интерфейса должна появляться как диалоговое окно , отображаемое в полноэкранном режиме либо в виде встроенного фрагмента (возможно, в зависимости от того, маленький или большой экран у устройства). С помощью класса [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) обеспечивается гибкость решения, поскольку он может вести себя как встраиваемый [Fragment](https://developer.android.com/reference/android/support/v4/app/Fragment.html).

Тем не менее, в этом случае нельзя использовать[AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html) или другие объекты [Dialog](https://developer.android.com/reference/android/app/Dialog.html) для построения диалогового окна. Если необходимо сделать [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html) встраиваемым, нужно определить пользовательский интерфейс диалогового окна в макете методом обратного вызова [onCreateView()](https://developer.android.com/reference/android/support/v4/app/Fragment.html" \l "onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)).

В качестве примера приведен [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html), который появляется либо в виде диалогового окна, либо в виде встраиваемого фрагмента (используя макет с наименованием purchase\_items.xml):

public class CustomDialogFragment extends DialogFragment {  
    /\*\* The system calls this to get the DialogFragment's layout, regardless  
        of whether it's being displayed as a dialog or an embedded fragment. \*/  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
            Bundle savedInstanceState) {  
        // Inflate the layout to use as dialog or embedded fragment  
        return inflater.inflate(R.layout.purchase\_items, container, false);  
    }  
    
    /\*\* The system calls this only when creating the layout in a dialog. \*/  
    @Override  
    public Dialog onCreateDialog(Bundle savedInstanceState) {  
        // The only reason you might override this method when using onCreateView() is  
        // to modify any dialog characteristics. For example, the dialog includes a  
        // title by default, but your custom layout might not need it. So here you can  
        // remove the dialog title, but you must call the superclass to get the Dialog.  
        Dialog dialog = super.onCreateDialog(savedInstanceState);  
        dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);  
        return dialog;  
    }  
}

Приведен пример кода, реализующего принятие решения об отображении фргмента в качестве диалогового окна или полноэкранного пользовательского интерфейса на основе размера экрана:

public void showDialog() {  
    FragmentManager fragmentManager = getSupportFragmentManager();  
    CustomDialogFragment newFragment = new CustomDialogFragment();  
      
    if (mIsLargeLayout) {  
        // The device is using a large layout, so show the fragment as a dialog  
        newFragment.show(fragmentManager, "dialog");  
    } else {  
        // The device is smaller, so show the fragment fullscreen  
        FragmentTransaction transaction = fragmentManager.beginTransaction();  
        // For a little polish, specify a transition animation  
        transaction.setTransition(FragmentTransaction.TRANSIT\_FRAGMENT\_OPEN);  
        // To make it fullscreen, use the 'content' root view as the container  
        // for the fragment, which is always the root view for the activity  
        transaction.add(android.R.id.content, newFragment)  
                   .addToBackStack(null).commit();  
    }  
}

Подробные сведения о выполнении операций с фрагментами приведены в руководстве [Фрагменты](https://developer.android.com/guide/components/fragments.html).

В приведенном примере mIsLargeLayout булеан указывает, должно ли текущее устройство использовать большой макет приложения (и отображать фрагмент как диалоговое окно, а не в полноэкранном режиме). Лучшим способом установить такой вид булеана является объявление [значения булевой переменной](https://developer.android.com/guide/topics/resources/more-resources.html#Bool) с [альтернативным](https://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources) значением для других размеров экранов. В качестве примера приведены два варианта булевых ресурсов для различных размеров экранов:

res/values/bools.xml

<!-- Default boolean values -->  
<resources>  
    <bool name="large\_layout">false</bool>  
</resources>

res/values-large/bools.xml

<!-- Large screen boolean values -->  
<resources>  
    <bool name="large\_layout">true</bool>  
</resources>

Затем можно инизиализировать значение mIsLargeLayout в течение выполнения метода операции [onCreate()](https://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)):

boolean mIsLargeLayout;  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity\_main);  
  
    mIsLargeLayout = getResources().getBoolean(R.bool.large\_layout);  
}

Отображение операции в качестве диалога на больших экранах

Вместо отображения диалогового окна в полноэкранном режиме на экранах малого размера можно отображать [Activity](https://developer.android.com/reference/android/app/Activity.html) в качестве диалогового окна на экранах большого размера. Выбор зависит от дизайна приложения, но отображение операции в качестве диалогового окна имеет смысл, когда приложение предназначено для использования на малых экранах, и необходимо улучшить взаимодейтсвие с ним на планшетах, показывая кратковременные операции в качестве диалогового окна.

Для отображения операции в качестве диалогового окна только на больших экранах необходимо применить тему[Theme.Holo.DialogWhenLarge](https://developer.android.com/reference/android/R.style.html#Theme_Holo_DialogWhenLarge) к элементу манифеста [<операция>](https://developer.android.com/guide/topics/manifest/activity-element.html):

<activity android:theme="@android:style/Theme.Holo.DialogWhenLarge" >

Подробная информация о темах операций приведена в руководстве [Стили и темы](https://developer.android.com/guide/topics/ui/themes.html).

Закрытие диалогового окна

Когда пользователь нажимает кнопки, созданные с помощью [AlertDialog.Builder](https://developer.android.com/reference/android/app/AlertDialog.Builder.html), система закрывает диалоговое окно самостоятельно.

Система также закрывает диалоговое окно, когда пользователь нажимает на элемент списка в диалоговом окне, за исключением списков с переключателями или флажками. В иных случаях можно вручную закрыть диалоговое окно путем вызова [dismiss()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "dismiss()) в [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html).

В случае, если необходимо произвести определенные действия после закрытия диалогового окна, можно реализовать метод [onDismiss()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "onDismiss(android.content.DialogInterface)) в[DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html).

Также можно *отменить* диалоговое окно. Это особое событие, возникающее, когда пользователь покинул диалоговое окно, не завершив задачу. Так происходит, когда пользователь нажимает кнопку *Назад*, касается экрана за областью диалогового окна, либо когда задано[cancel()](https://developer.android.com/reference/android/app/Dialog.html#cancel()) в [Dialog](https://developer.android.com/reference/android/app/Dialog.html) (например, в качестве отклика на нажатие кнопки «Отмена» в диалоговом окне).

Как показано в примере выше, можно ответить на событие отмены с помощью [onCancel()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "onCancel(android.content.DialogInterface)) в классе [DialogFragment](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html).

**Примечание:** Система вызывает [onDismiss()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "onDismiss(android.content.DialogInterface)) при каждом событии, которое вызывается методом обратного вызова[onCancel()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html#onCancel(android.content.DialogInterface)). Тем не менее, при вызове[Dialog.dismiss()](https://developer.android.com/reference/android/app/Dialog.html#dismiss()) или [DialogFragment.dismiss()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "dismiss()), система вызывает [onDismiss()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "onDismiss(android.content.DialogInterface)) *, а не* [onCancel()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html" \l "onCancel(android.content.DialogInterface)). Поэтому в общих случаях вызов[dismiss()](https://developer.android.com/reference/android/support/v4/app/DialogFragment.html#dismiss()) производится при нажатии пользователем *положительной* кнопки в диалоговом окне, а после диалоговое окно закрывается.