

将应用程序部署到安卓设备

在 Android 上，应用程序以特殊结构类型的 ZIP 包分发，称为应用程序包（APK）或 Android 应用捆绑包（AAB）。APK 文件可以下载到设备上并在设备上执行，而 AAB 旨在由 Google Play 商店解释，并用于生成 APK 文件。

Qt for Android 有 armv7a、arm64-v8a、x86 和 x86-64 的二进制文件。若要在应用程序中支持多个不同的 ABI，请生成一个 AAB，其中包含每个 ABI 的二进制文件。Google Play 商店使用 AAB 为发出下载请求的设备生成优化的 APK 包，并使用您的发布商密钥自动对其进行签名。

Qt Creator 支持以下 Android 应用程序的部署方法：

- › 作为独立的可分发应用程序包（APK）。
- › 从 Qt 5.14.0 开始，作为应用程序捆绑包（AAB），打算在 Google Play 商店中分发。

注意：从 Qt Creator 4.12 开始，不支持 Ministro。

要指定应用程序包的设置，请选择 **项目 > 构建 > 构建安卓 APK > 详细信息**。

有关运行应用程序的选项的详细信息，请参阅 [指定 Android 设备的运行设置](#)。

包装应用

由于将应用程序捆绑为 APK 包并非易事，因此 Qt 5 提供了一个称为的部署工具。当您使用 *适用于 Android 的 Qt kit* 部署应用程序时，Qt Creator 使用该工具创建必要的文件并将其捆绑到 APK 中：androiddeployqtandroiddeployqt

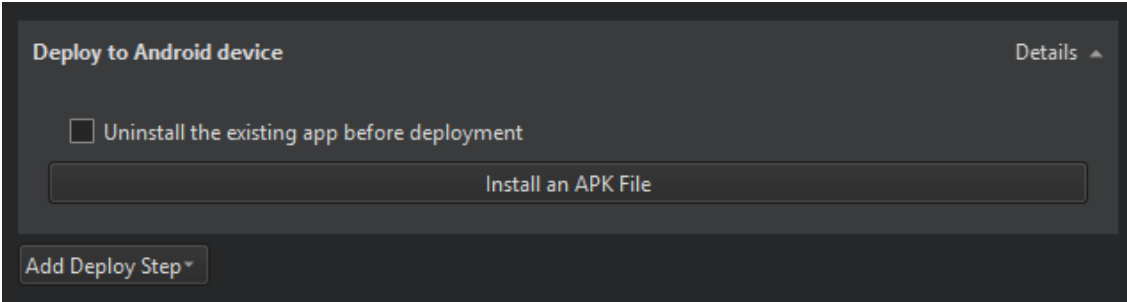
- › Java 文件，作为应用程序的入口点，自动加载 Qt 并在应用程序中执行本机代码。
- › AndroidManifest.xml，提供有关应用程序的元信息。
- › 其他 XML 文件，用于指定应用程序的依赖项。
- › 资源文件。
- › 库和 QML 文件，可以包含在项目中，具体取决于您选择的部署方法。
- › 下载和使用 Gradle 所需的 Gradle 包装器。
- › Java IDE 所需的 Gradle 脚本，例如 Android Studio。它允许用户在不复制我们的 Java 源代码的情况下扩展 Java 部分。它还允许 IDE 提供代码完成、语法突出显示等。

仅当您使用 Gradle 构建应用程序包时，才会捆绑 Gradle 包装器和脚本。有关更多信息，请参阅 [连接安卓设备](#)。

若要查看该工具创建的包，请选中“**生成后打开包位置**”复选框。androiddeployqt

指定部署设置

可用的部署设置列在“**方法**”字段中。若要为项目添加部署方法，请选择“**添加**”。



若要重命名当前部署方法，请选择“**重命名**”。

若要删除当前部署方法，请选择“**删除**”。

程序包部署在您在[工具包选择器](#)中选择的 Android 设备上。若要添加设备，请选择“**管理**”。

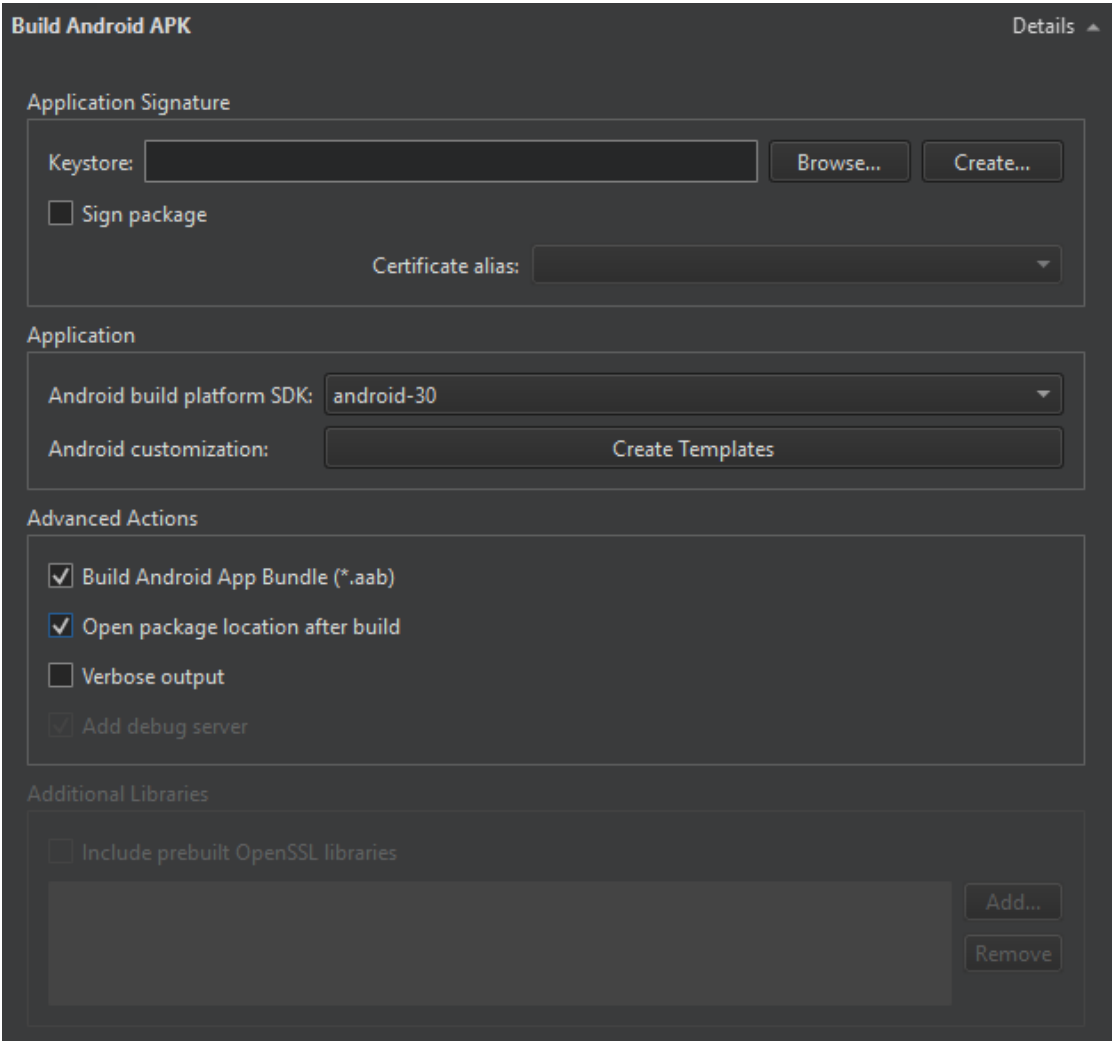
有关为应用程序指定其他启动选项的详细信息，请参阅[指定 Android 设备的运行设置](#)。

若要从设备中删除以前安装的文件，请选择“**在部署之前卸载现有应用**”。

要将预构建的 APK（例如第三方应用程序）安装到设备，请选择**安装 APK 文件**。

指定包的设置

要指定该工具的设置，请选择**项目 > 构建和运行 > 构建 > 构建 Android APK > 详细信息**。`androiddeployqt`



该工具使用配置信息来创建 APK。有关可用选项的更多信息，请参阅[androiddeployqt](#)。`androiddeployqt`

您可以在[编译输出](#)中查看有关该工具正在执行的操作的信息。若要查看其他信息，请选中“**详细输出**”复选框。`androiddeployqt`

选择 API 级别

在Android **构建平台** SDK字段中，您可以选择用于构建应用程序的 API 级别。通常，您应该选择可用的最高 API 级别。

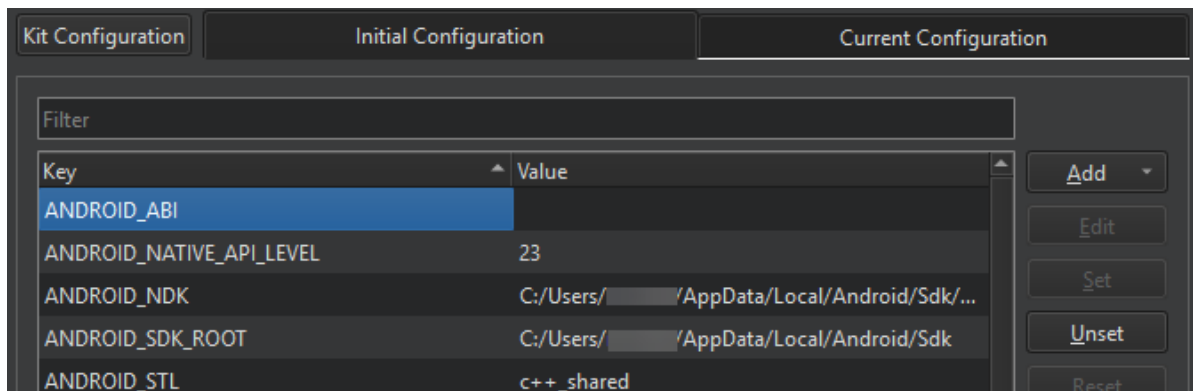
注意：对于 Qt 5.12.0 到 5.12.5 和 Qt 5.13.0 到 5.13.1，应使用 Android 构建平台 SDK 28。对于Qt 5.13.1更新的版本，应使用构建平台SDK 29或最新版本。

此字段不指定支持的最低 API 级别，也不指定可在 Android 清单中指定的目标 API 级别。请参阅[编辑清单文件](#)。有关 Android API 级别的更多信息，请参阅[什么是 API 级别？](#)。

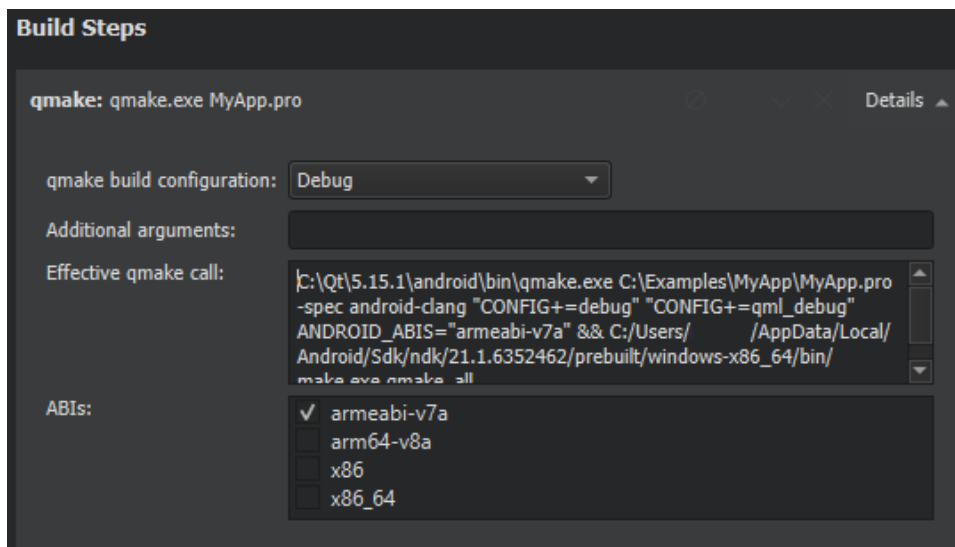
建立AAB

若要在本地测试应用程序，请使用 APK 格式，因为包可以直接上传到设备并运行。要分发到 Google Play 商店，请通过选中**构建 Android 应用捆绑包 (*.aab)** 复选框来创建 AAB。

使用 CMake 进行构建时，可以在CMake部分的**初始配置**中查看选定的 ABI。您可以将其他 ABI 设置为键的值：ANDROID_ABI



使用 Qbs 或 qmake 进行构建时，可以在“**构建步骤**”的“ABI”字段中选择 ABI：



对安卓软件包进行签名

若要发布应用程序，必须使用公钥-私钥对对其进行签名，公钥-私钥对由**证书**和相应的**私钥**组成，并由**别名**标识。密钥对用于验证应用程序的未来版本是否确实由您创建。

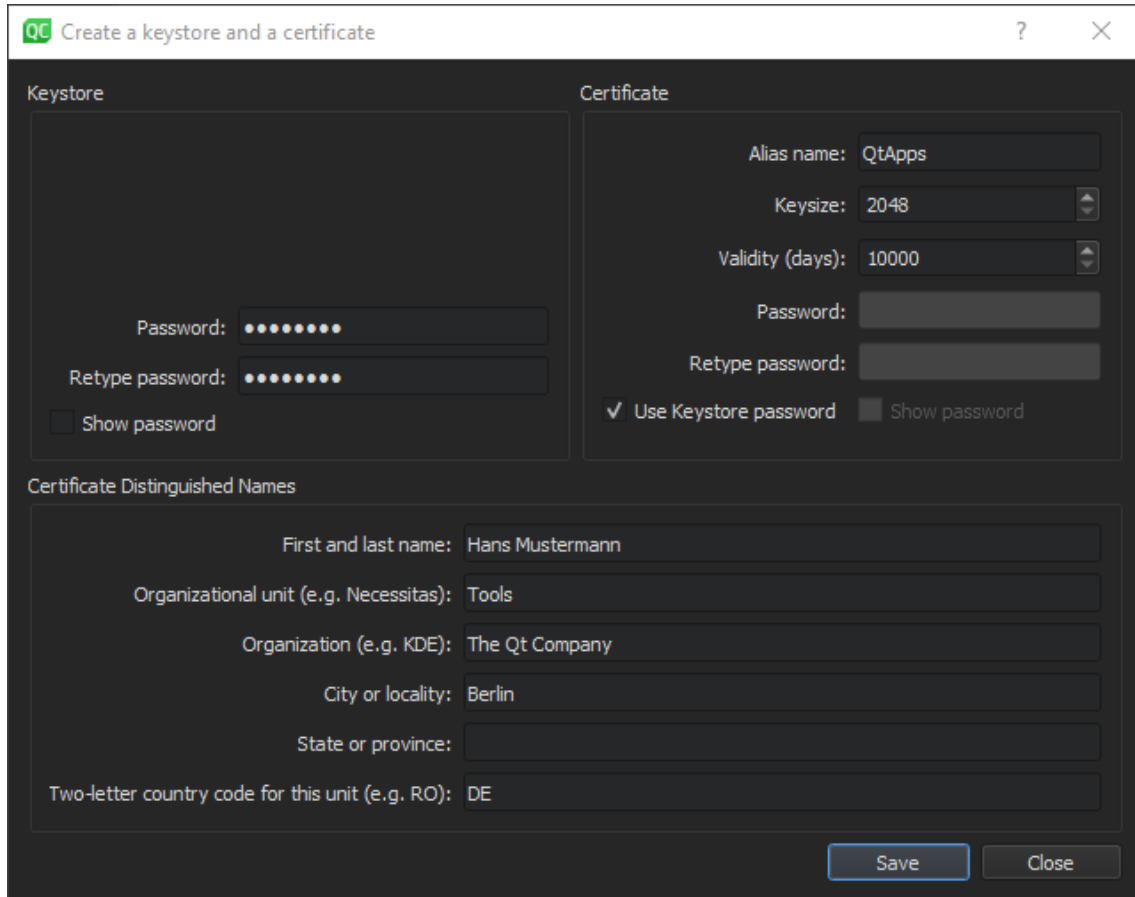
警告：将密钥对保存在安全的地方并备份副本，因为如果丢失密钥对，则无法更新应用程序。

您可以使用Qt Creator生成密钥库和**自签名证书**。生成的证书具有 X.509 v3 数字证书的结构。它包含有关证书的版本、序列号和有效期、用于加密数据的算法的 ID、颁发证书的组织以及证书的**使用者**（所有者）的信息。如果是自签名证书，证书的颁发者和

对 Android 应用程序进行签名时，必须从密钥库中选择一个包含证书和证书别名的密钥库。别名的公钥（证书）在签名期间嵌入到 APK 中。

要创建密钥库和自签名证书，请执行以下操作：

1. 在密钥库字段中，选择**创建**以创建一个新的密钥库，该**密钥库**在创建**密钥库和证书**对话框中包含一个密钥对：



2. 在密钥库组中，输入用于保护**密钥库**的密码。
3. 在“证书”组中，指定**证书**的密钥大小和有效期。您可以指定单独的密码来保护密钥对或使用密钥库密码。
4. 在“**证书可分辨名称**”组中，输入有关您自己以及标识密钥对颁发者和所有者的公司或组织的信息。
5. 选择“**保存**”。
6. 在“密钥库文件名”对话框中，输入**密钥库**的名称并为其选择位置。
7. 在密钥库对话框中，输入密钥库密码以在密钥库中创建**密钥对**。

要使用密钥对对 Android 程序包进行签名，请设置指定程序包**设置**中所述的**签名程序包**组设置：

1. 在密钥库字段中，选择**选择**以选择现有密钥库。
2. 在证书别名字段中，从密钥库包含的密钥对列表中选择**别名**。
3. 选中“**对包进行签名**”复选框以使用别名对 Android 包进行签名。

添加外部库

Qt Creator会自动检测应用程序使用的Qt库，并将它们添加为依赖项。如果应用需要外部库，请在**项目>构建>构建Android APK>其他库**字段中指定它们。这些库将复制到应用程序的库文件夹中，并在启动时加载。

若要添加 OpenSSL 库，请在“**其他库**”组中选择“**包括预构建的 OpenSSL 库**”。这将添加在Android OpenSSL组中的**设备设置**中定义的OpenSSL包含项目。这可用于 QMake 和 CMake 项目。

否则，您可以手动将必需库的路径添加到“**其他库**”字段中。libssl.solibcrypto.so

包名称

Android 应用程序包通常使用分层模式命名，层次结构中的级别用句点 (.) 分隔。通常，包名称以组织的顶级域名开头，后跟组织的域名和按相反顺序列出的任何子域名。然后，组织可以为其包选择特定名称。包名称应尽可能以全部小写字母书写。例如。`org.qtproject.example`

Java 语言规范的第 7.7 节中描述了消除软件包名称歧义的完整约定以及当 Internet 域名不能直接用作软件包名称时命名软件包的规则。

有关软件包名称的更多信息，请参阅[Android 应用程序基础知识](#)。

造型

Qt使用不同的方法来确定Qt控件和Qt快速控件的样式：

- › 在项目中使用 Qt Widgets 或 Qt Quick Controls 1 时，选择**默认或完整**。

注意：此方法使用一些 Android 非 SDK 接口，这些接口从 Android 9.0（API 28）开始受到 Google 的限制。

- › 使用 Qt 快速控件 2 但不使用 Qt 控件或 Qt 快速控件 1 时选择**最小**。这比使用默认或完整选项更快。
- › 既不使用 Qt Widgets 也不使用 Qt Quick Controls 1 或 2 时，请选择**“无”**。

屏幕方向

您可以根据传感器读数或用户首选项指定不同的选项来确定屏幕方向。下表列出了可用的选项。

取向	描述
未指定	系统选择方向。它使用的策略以及在特定上下文中做出的选择可能因设备而异。
后	使用与活动堆栈中紧靠其下方的活动相同的方向。
景观	横向，其中显示宽度大于其高度。
肖像	纵向，其中显示高度大于其宽度。
反向横向	景观方向与正常景观相反。
反向纵向	纵向方向与正常纵向相反。
传感器环境	横向，但根据设备传感器，它可以是正常或反向横向。即使用户锁定了基于传感器的旋转，也会使用传感器。
传感器组	纵向，但根据设备传感器，它可以是正常或反向纵向。即使用户锁定了基于传感器的旋转，也会使用传感器。

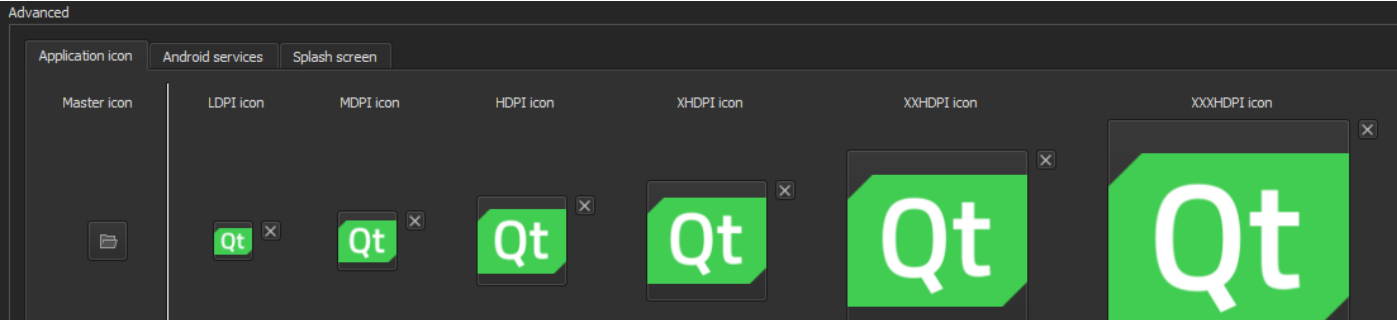
用户格局	
用户画像	纵向，但根据设备传感器和用户的偏好，它可以是正常或反向纵向。
传感器	方向由设备方向传感器确定。显示器的方向取决于用户握持设备的方式。当用户旋转设备时，它会发生变化。但是，默认情况下，某些设备不会旋转到所有四个可能的方向。要允许所有四个方向，请选择完整传感器选项。即使用户锁定了基于传感器的旋转，也会使用该传感器。
全传感器	方向由设备方向传感器确定四个方向中的任何一个。这类似于传感器选项，不同之处在于它允许四种可能的屏幕方向中的任何一种，而不管设备通常做什么。例如，某些设备通常不会使用纵向反向或反向横向，但此选项会启用它们。
无传感器	方向是在不参考物理方向传感器的情况下确定的。传感器将被忽略，因此显示器不会根据用户移动设备的方式旋转。
用户	用户当前的首选方向。
完全用户	如果用户已锁定基于传感器的旋转，则此选项的行为方式与用户选项相同。否则，它的行为与完整传感器选项相同，并允许四种可能的屏幕方向中的任何一种。
锁	将方向锁定为其当前旋转，无论该旋转是什么。

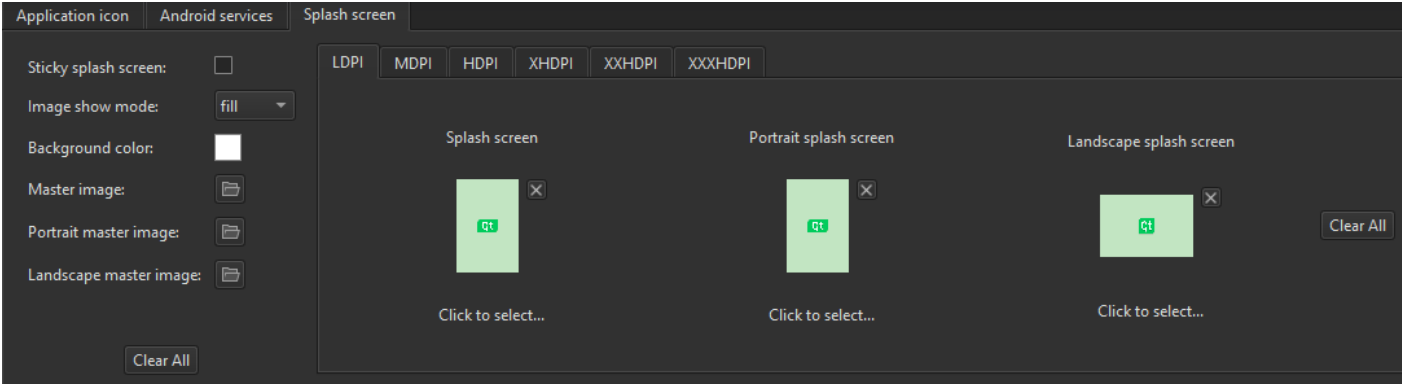
图标和初始屏幕

您可以将不同的图像设置为在低、中、高和超高 DPI 显示器上显示为应用程序图标和初始屏幕。以下列表汇总了通常与每个类别关联的 DPI 值：

- › 低密度：~120dpi
- › 中密度（MDPI）：~160dpi
- › 高密度（HDPI）：~240dpi
- › 超高密度（XHDPI）：~320dpi
- › 超高密度（XXHDPI）：~480dpi
- › 超高密度（XXXHDPI）：~640dpi

在应用程序图标选项卡中指定**图标**的设置。选择分辨率最高的图像作为**主图标**。Qt Creator根据需要调整图标大小，并将其版本设置为在低，中，高和超高DPI显示器上显示。或者，分别设置每个分辨率的图标。





默认情况下，绘制活动时自动隐藏初始屏幕。若要使其在调用`QNativeInterface::QAndroidApplication::hideSplashScreen()`之前保持可见，请选中“**粘滞初始屏幕**”复选框。

在**图像显示模式**下，选择是将初始屏幕居中放在设备显示屏上，还是缩放初始屏幕以填充显示屏。

在背景颜色中设置**背景颜色**。

选择分辨率最高的图像作为主图像，纵向主图像和**横向主图像**。

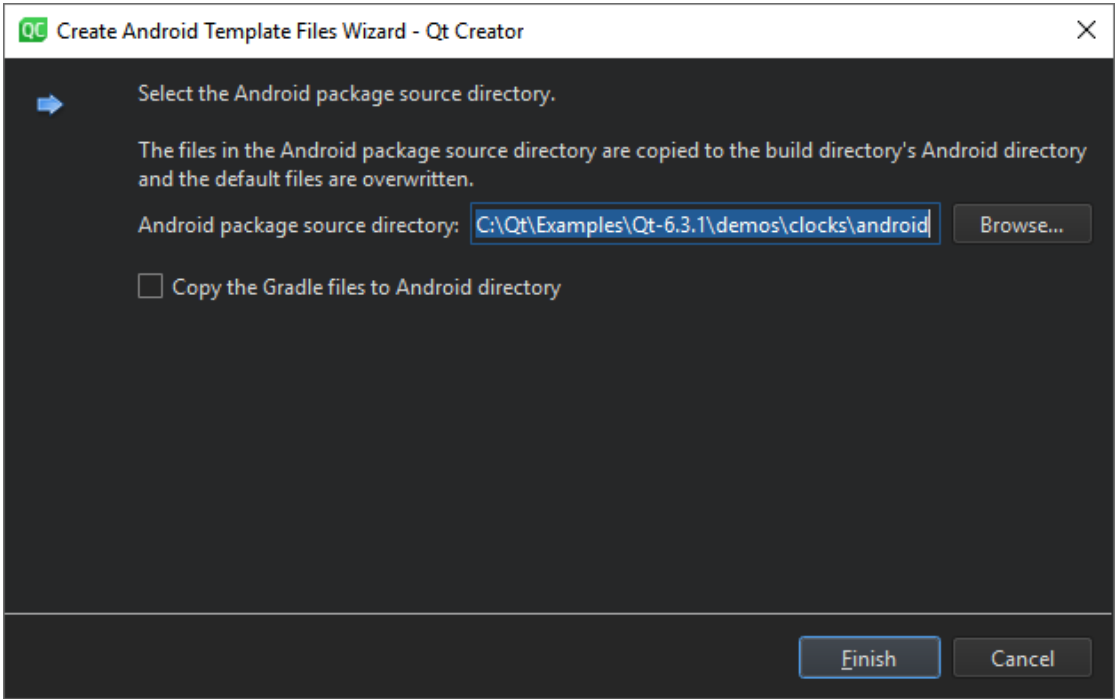
选择“**全部清除**”以重置所有设置或删除所有图像。

安卓清单编辑器

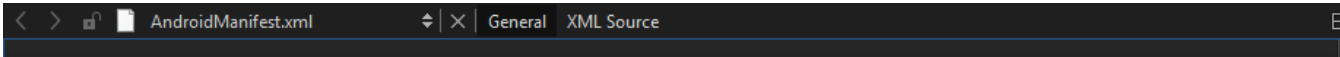
如果您使用 qmake 作为构建系统，则可以创建一个 Android 清单文件并在 Qt Creator 中进行编辑。

要创建 Android 清单文件并在 Android 清单编辑器中打开它，请执行以下操作：

1. 选择**项目>构建>构建安卓APK>创建模板**。
2. 检查Android **包源目录**中的路径。



3. 选择将 **Gradle 文件复制到 Android 目录**，如果您计划扩展 Qt 应用程序的 Java 部分。
4. 选择“**完成**”以将模板文件复制到目录并打开清单文件进行编辑。android
5. 在**包名称** 字段中，输入应用程序的有效**包名称**。例如。该应用程序由自动生成的 Java 启动器启动，该启动器与应用程序一起打包到 Android 包（.apk）中。org.example.myapplication



Version code: -- %%INSERT_VERSION_CODE%% --	Activity name: -- %%INSERT_APP_NAME%% --
Version name: -- %%INSERT_VERSION_NAME%% --	Style extraction: minimal
Minimum required SDK: Not set	Screen orientation: portrait
Target SDK: Not set	

- 您可以在**版本代码**字段中指定包的内部版本号。它用于确定应用程序的一个版本是否比另一个版本更新。在**版本名称**字段中，指定向用户显示的版本号。
- 在**“所需的最低 SDK”**字段中，选择运行应用程序所需的最低 API 级别。Qt Creator 支持的最低 API 级别是 android-9。但是，Qt 版本可能具有不同的最低 API 级别，因此 Qt Creator 不允许您选择为工具包指定的 Qt 版本不支持的 API 级别。
- 在**“目标 SDK”**字段中，选择应用程序的目标 API 级别。这会影响操作系统中某些兼容性功能的激活。该工具默认使用的值为 14，这意味着默认情况下不会启用系统导航栏中的溢出按钮。androiddeployqt
- 在**应用程序名称**字段中，设置应用程序的名称。
- 在**活动名称**字段中，设置**活动名称**。
- 在**样式提取**字段中，设置 Qt 用于**确定要使用的 UI 样式**的方法。
- 在**屏幕方向**字段中，选择用于确定**屏幕方向**的选项。
- 在**“应用程序图标”**中，根据屏幕分辨率指定要用作**应用程序图标**的图像。
- 在**“初始屏幕”**中，根据屏幕方向和分辨率选择要显示为**初始屏幕**的图像。
- 在**Android 服务**中，选择**“添加”**以**添加服务**。必须至少为新服务输入服务类名称。如果选择**“在外部进程中运行”**，则还需要输入进程名称。如果选择**“在外部库中运行”**，则需要输入库名称。服务参数对于未在外库中运行的服务是必需的。有关编写服务代码和服务结构的更多信息，请参阅[Android 服务](#)。

Service class name	Run in external process	Process name	Run in external library	Library name	Service arguments	
⚠	<input type="checkbox"/> Run in external process		<input type="checkbox"/> Run in external library			<div>Add</div> <div>Remove</div>

- 在**“权限”**字段中，可以指定应用程序所需的权限。从 Android 6.0 (API 23) 开始，必须在运行时请求权限（参见 [QtAndroidPrivate::requestPermission\(\)](#)）。对于较低的 Android API 级别，系统会要求用户在安装应用程序时授予权限。然后，Android 操作系统授予应用程序访问相应数据和功能的权限。

Permissions

☒ Include default permissions for Qt modules.
☒ Include default features for Qt modules.

android.permission.ACCESS_CHECKIN_PROPERTIES

Add

Remove

- 选中**“包括Qt模块的默认权限”**和**“包括Qt模块的默认功能”**复选框，以添加Qt库所需的权限。这可以用于Qt Core或Qt Location。android.permission.WRITE_EXTERNAL_STORAGE android.permission.ACCESS_COARSE_LOCATION
- 若要添加权限，请从列表中选择该权限，然后单击**“添加”**。

在顶部标题上，选择**“XML 源”**选项卡以编辑XML格式的文件。

< 部署到设备

将应用程序部署到引导2Qt设备 >

©2022 Qt有限公司 此处包含的文档贡献的版权归 他们各自的所有者。此处提供的文档根据自由软件基金会发布的GNU 自由文档许可证版本1.3的条款进行许可。Qt和相应的徽标是Qt有限公司在芬兰和/或全球其他国家的商标。所有其他商标均为财产 其各自所有者。



联系我们

公司

- 关于我们
- 投资者
- 编辑部
- 职业
- 办公地点

支持

- 支持服务
- 专业服务
- 合作 伙伴
- 训练

社区

- 为Qt做贡献
- 论坛
- 维基
- 下载
- 市场

发牌

- 条款和条件
- 开源
- 常见问题

对于客户

- 支持中心
- 下载
- Qt登录
- 联系我们
- 客户成功案例