11.10.3 Gambas-AppImages

**Inhaltsverzeichnis**



Die Gambas-IDE bietet zwar die Erzeugung von Installationspaketen für verschiedene Distributionen an. Jedoch werden Sie schnell merken, dass die Installation einer Gambas-Applikation auf einer Zielplattform linux-typische Hürden bereit hält, mit denen ein Anwender schnell überfordert ist. Das gilt beispielsweise für Installationspakete, die auf Plattformen geschnürt wurden, auf denen Gambas per PPA installiert wurde. In so einem Fall kommt man oft nicht umhin, auf der Zielplattform ebenfalls das PPA einzubinden. Informationen dazu finden Sie im Gambas-Buch im Kapitel 2.3 Installation aus alternativen Quellen – PPA. Außerdem ist es mühsam, für jede Distribution ein spezifisches Installationspaket zu erzeugen und oft fehlt dafür auch das notwendige Hintergrundwissen.

AppImages bieten hier einen alternativen Weg und haben den großen Vorteil, dass das Programm ohne spezielle Anpassung auf unterschiedlichen Linux-Betriebssystemen läuft, da es alle notwendigen Komponenten in seinem Image enthält. AppImages sind zudem portabel und lassen sich von jeder Quelle aus starten, u.a. auch von einem USB-Stick. Sie können sogar in einer Sandbox wie Firejail laufen. Diese Vorteile erkaufen Sie sich durch eine erhöhte Startzeit und eine wesentlich größere Datei, die u.U. nicht mehr per EMail verschickt werden kann. Wenn Sie sich von einem AppImage trennen möchten, dann löschen Sie es ganz einfach.

### 11.10.3.1 Anforderungen an den Gambas-Quelltext

#### 11.10.3.1.1 Programm-Konfiguration durch gb.settings

Gambas-Applikationen speichern Konfigurationen typischerweise im Verzeichnis /home/user/.config/gambas3 ab. Denken Sie daran, dass das Verzeichnis gambas3 auf Zielplattformen fehlt, wenn kein Gambas installiert ist. Es empfiehlt sich deshalb beim Start des Programms zu prüfen, ob das Verzeichnis gambas3 existiert. Falls nicht, sollten Sie es unmittelbar anlegen.

#### 11.10.3.1.2 Nutzung von Kommandozeilen-Tools mit dem Shell-Kommando

Für ein Gambas-Programm, das als AppImage verteilt werden soll, müssen Sie berücksichtigen, dass die Zielplattformen recht unterschiedlich sein können. So kann es beispielsweise passieren, dass notwendige Programme auf der Zielplattform fehlen. So sollten Sie zum Beispiel bei der Nutzung von Gambas-Shell-Kommandos geeignete Prüfroutinen in den Quelltext einfügen, die gegebenenfalls auf notwendige Installationen von fehlenden Paketen hinweisen. Abhängigkeiten von unterschiedlichen Desktops sollten vermieden oder aber berücksichtigt werden.

#### 11.10.3.1.3 Einbetten von Kommandozeilen-Tools in das AppImage

Um die Abhängigkeit von der Verfügbarkeit von Kommandozeilen-Tools auf der Zielplattform zu elimi-

nieren, können Sie die erforderlichen Tools auch als innere Ressource in das AppImage einbetten. Geben Sie dazu im Dialog zur Erstellung des Installationspakets unter "Extra Abhängigkeiten" das gewünschte Tool als Abhängigkeit an. Diese Tools sind dann aufgrund der veränderten Umgebung allerdings nicht mehr für Shell-Kommandos erreichbar. Verwenden Sie deshalb in so einem Fall das Exec-Kommando. Beachten Sie, dass mit der Angabe einer Abhängigkeit die Größe des AppImage um die Größe der eingebundenen Pakets wächst.

#### 11.10.3.1.4 Probleme bei der Verwendung von curl oder gb.net.curl

Bei der Verwendung von curl als Kommandozeilen-Tool oder des Gambas HttpClients (gb.net.curl) stelllte sich heraus, dass AppImages auf einigen Plattformen einwandfrei funktionierten, während sie auf anderen – wie Opensuse – versagten. Als Grund werden unterschiedlich Speicherorte für SSL-Zertifikate vermutet. Es kann deshalb erforderlich sein, solche Besonderheiten einer Distribution zu berücksichtigen. Konkrete Lösungen wurden bisher nicht erarbeitet und können somit noch nicht angeboten werden. Deshalb empfiehlt es sich vorläufig, curl als Shell- oder Exec-Kommando zu verwenden, ohne curl als "Extra Abhängigkeit" ins AppImage einzubetten.

### 11.10.3.2 Voraussetzungen für die Erzeugung von AppImages

#### 11.10.3.2.1 AppImages und Ubuntu

Die Erzeugung von AppImages ist gegenwärtig (Juni 2024) nur auf Plattformen möglich, die auf dem Betriebssystem Ubuntu basieren. Das schließt Linux Mint mit ein. Wer andere Betriebssysteme verwendet, kann sich beispielsweise mit einer ubuntu-basierten virtuellen Maschine weiter helfen, auf der Sie auch gern eine ältere Variante des Betriebssystems installieren können, während die Gambas-Version möglichst aktuell sein sollte. Die folgende Beschreibung wurde unter Verwendung von Mint 20.3 Una erzeugt und ist erfahrungsgemäß nicht auf Ubuntu 20.4 übertragbar.

#### 11.10.3.2.2 Installation des Appimage-Builders

Um AppImages mit der Gambas-IDE erzeugen zu können, ist der AppImage-Builder von Simon Peter erforderlich. Dieser Builder kann entweder direkt installiert werden oder über ein AppImage verwendet werden. Installations-Anweisungen für die verschiedenen Distributionen sind hier zu finden:

```
https://appimage-builder.readthedocs.io/en/latest/intro/install.html
```

Für die Installation auf Mint-Plattformen 20.3 haben sich diese Methoden bewährt:

Installation der erforderlichen Abhängigkeiten:

```
sudo apt install -y libgdk-pixbuf2.0-dev patchelf python3-pip python3-setuptools zsync
```

Installation des AppImage-Builders:

```
sudo pip3 install appimage-builder
```

#### 11.10.3.2.3 Erzeugung eines AppImages in der IDE

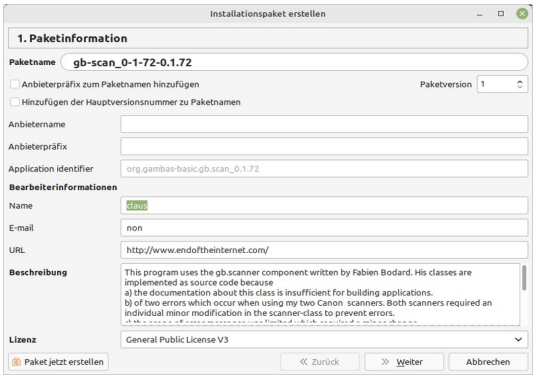


Abbildung 11.10.3.2.1: 1. Paketinformation

Laden Sie zunächst den Quellcode des Gambas-Programms, aus dem Sie ein AppImage erzeugen wollen, in die Gambas-Entwicklungsumgebung (IDE). Erzeugen Sie zunächst aus dem Projekt eine ausführbare Datei. Wählen Sie dann den Menüeintrag "Projekt/Installationspaket erzeugen". Es erscheint der Dialog zur Erzeugung des Installationspakets.

Wenn Sie auf '» Weiter' klicken erscheint dieser Dialog:

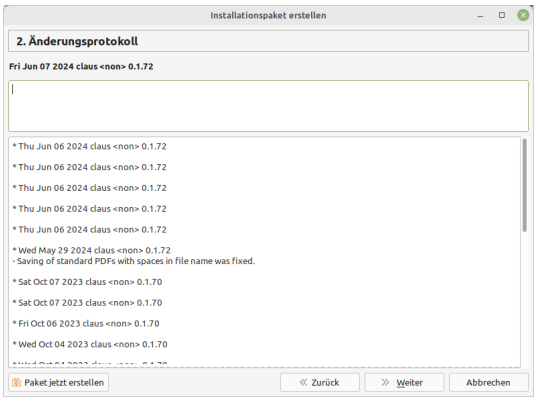


Abbildung 11.10.3.2.2: 2. Änderungsprotokoll

Fügen Sie hier optional einen kurzen Kommentar zu den Änderungen ein oder klicken wieder einfach auf '» Weiter' – es erscheint dieser Dialog:

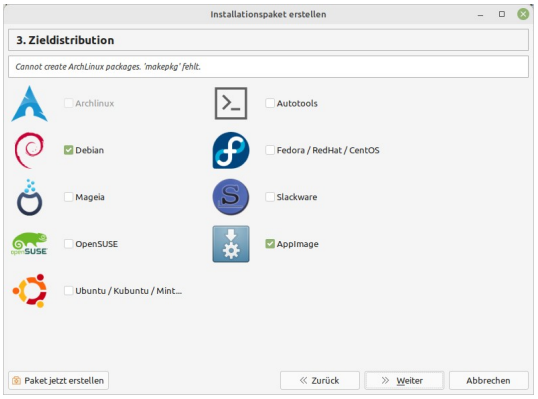


Abbildung 11.10.3.2.3: 3. Zieldistribution

Wählen Sie hier "AppImage" aus und klicken Sie auf '» Weiter'. Bitte beachten Sie, dass Sie alle hier aufgeführten Dialoge grundsätzlich überspringen können, indem Sie "Paket jetzt erstellen" anklicken. Damit wird der Paketierungsprozess unmittelbar gestartet. Diese Taste werden Sie als "Abkürzung" auf allen Dialog-Fenstern finden. Falls Sie ein Ubuntu-Derivat wie Linux Mint verwenden, so wählen Sie im folgenden Dialog zunächst die Ubuntu-Version aus, die Ihrem Mint 20.3 entspricht (Ubuntu 20.04). Dann tragen Sie das zu nutzende Ubuntu-Repository "http://archive.ubuntu.com/ubuntu" ein und verzichten zunächst auf die Option zum Erzeugen einer Signatur.

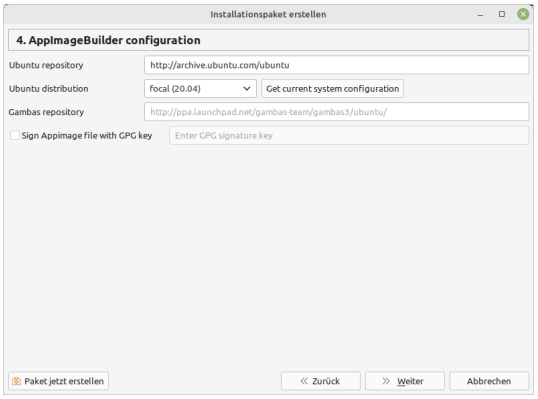


Abbildung 11.10.3.2.4: 4. AppImageBuilder-Konfiguration

Klicken Sie auf '» Weiter'.

Wählen Sie im folgenden Dialog die Paketgruppe aus (Option):

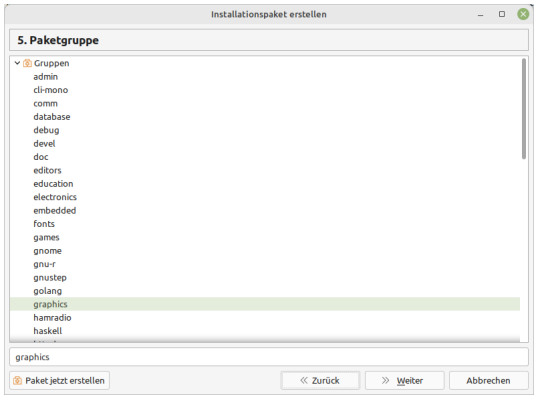


Abbildung 11.10.3.2.5: 5. Auswahl der Paketgruppe

Klicken Sie auf '» Weiter'.

Wählen Sie im nächsten Dialog den Menüeintrag aus (Option):

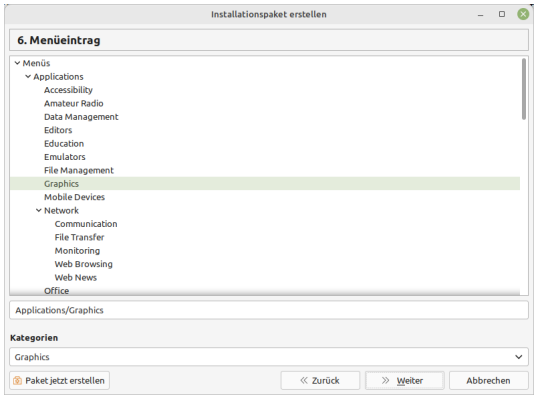


Abbildung 11.10.3.2.6: 6. Menüeintrag

Nachdem Sie auf '» Weiter' geklickt haben, können Sie im folgenden Dialog die verwendeten Mime-Typen und eine Desktop-Konfiguration angeben (Option):

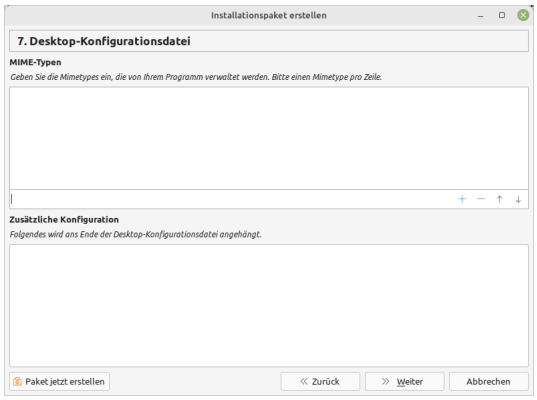


Abbildung 11.10.3.2.7: 7. Desktop-Konfigurationsdatei

Klicken Sie auf '» Weiter'.

Im folgenden Dialog müssen Sie u.U. wichtige Angaben zu Abhängigkeiten machen.

Diese werden dann bei der Paketbildung mit berücksichtigt:

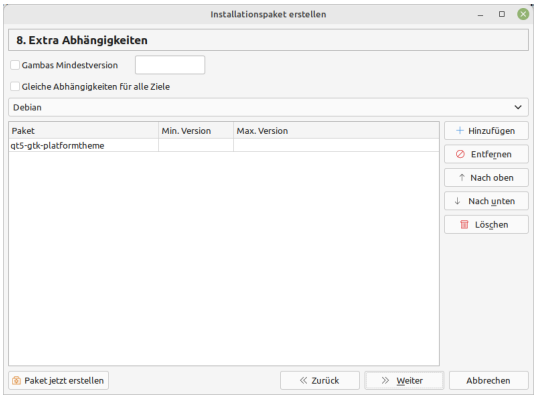


Abbildung 11.10.3.2.8: 8. Festlegung von Abhängigkeiten

An dieser Stelle kann es erforderlich sein, Pakete in die Abhängigkeitsliste aufzunehmen.

Achtung:

Bei Programmen, die die Komponente gb.media nutzen, müssen alle erforderlichen GStreamer-Plug-ins als Abhängigkeiten angeben werden. Für die Audio-Wiedergabe sind das die folgenden Plugins:

- gstreamer1.0-pulseaudio,
- gstreamer1.0-plugins.base,
- gtsreamer1.0-plugins-good.

Für X11-Visualisierungen bei Verwendung des ximagesink-Elements verwenden Sie:

- gstreamer1.0-x

und für die Wiedergabe von Videos in verschiedenen Formaten :

- gstreamer1.0-libav .

In einem weiteren Fall musste die Gambas-Komponente gb.net als Abhängigkeit angegeben werden, weil die Komponente im Programm erst zur Laufzeit geladen wurde.

Durch einen Klick auf '» Weiter' kommen Sie in den nächsten Dialog, in dem Sie zusätzliche Dateien mit angeben können:

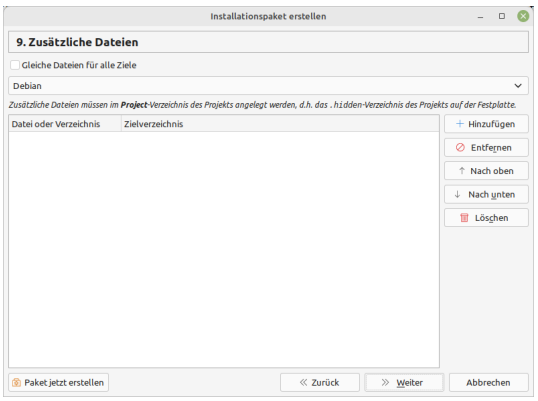


Abbildung 11.10.3.2.9: 9. Angabe zusätzlicher Dateien

Durch einen Klick auf '» Weiter' kommen Sie zum nächsten Dialog, in dem Sie das Verzeichnis vorgeben können, in dem das erzeugte AppImage abgelegt wird:

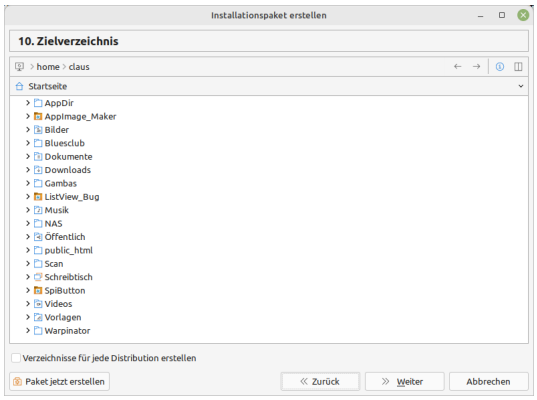


Abbildung 11.10.3.2.10: 10. Angabe des Zielverzeichnisses

Per Klick auf '» Weiter' kommen Sie auf die letzte Dialogseite.

Auf dieser Seite brauchen Sie nur noch auf "Paket erstellen" klicken, um den Builder-Prozess zu starten, dessen Fortschritt Ihnen angezeigt wird. Nach einigen Sekunden sollte das Fortschrittsjournal Ihnen am Ende die Meldung "Die Pakete wurden erfolgreich erstellt" anzeigen.

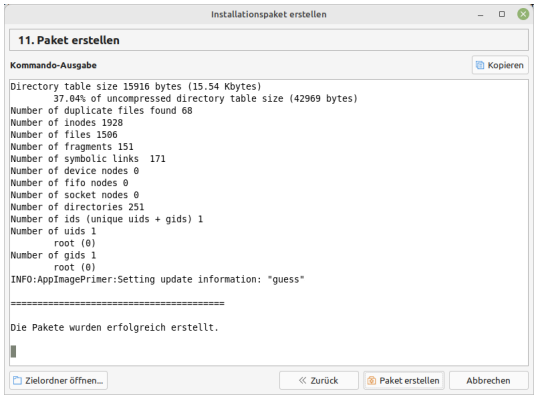


Abbildung 11.10.3.2.11: 11. Paket estellen

Abschließend finden Sie das neue AppImage im angegebenen Zielverzeichnis. Sie können die einwandfreie Funktion der Applikation sofort überprüfen, indem Sie diese mit einem Doppelklick starten.

### 11.10.3.3 AppImage auf einer Fremdplattform ausführen

#### 11.10.3.3.1 AppImage ausführbar machen

Damit ein AppImage auf einer anderen Plattform gestartet werden kann, muss es dort zunächst ausführbar gemacht werden. Das erreichen Sie mit Hilfe des Dateimanagers oder in der Konsole:

```
chmod +x TV_recorder-x86_64.AppImage
```

#### 11.10.3.3.2 Starten eines AppImages

Starten können Sie ein AppImages per Doppelklick mit der Maus oder in der Konsole:

```
./TV_recorder-x86_64.AppImage
```

### 11.10.3.4 Signierung eines Gambas-AppImages

Bei Gambas-Versionen >= 3.20.0 ist die Signierung von AppImages möglich.

Damit kann die Authentizität eines AppImages durch Überprüfung der verwendeten Signatur sichergestellt werden. Hierfür ist ein GnuPG-Schlüssel erforderlich.

#### 11.10.3.4.1 Nutzung existierender GnuGPG-Schlüssel

Wer die GnuPG-Verschlüsselung beim EMail-Programm Thunderbird verwendet, hat bereits einen geeigneten Schlüssel, der im Appimage-Builder verwendet werden kann. Rufen Sie im GUI des Thunderbirds den Menü-Punkt "Extras/OpenPGP-Schlüssel verwalten" auf. In dem erscheinenden Dialog markieren Sie den eigenen Schlüssel und klicken dann auf die rechte Maustaste. Es erscheint ein Kontext-Menü. In diesem Menü wählen Sie den Eintrag 'Kopieren/Fingerabdruck'. Dadurch wird der Schlüssel in die Zwischenablage kopiert. Diesen Schlüssel können Sie in den o.a. Dialog '4. AppImageBuilder configuration' eingeben.

#### 11.10.3.4.2 Erzeugung eines neuen Signatur-Schlüssels

Alternativ zur Verwendung eines existierenden GnuPG-Schlüssels können Sie sich in der Konsole einen neuen Schlüssel erzeugen. Die beispielhaften Angaben und der generierte Schlüssel sind nicht real und nicht reproduzierbar:

```
claus@claus-VirtualBox:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Bitte wählen Sie, welche Art von Schlüssel Sie möchten:
   (1) RSA und RSA (voreingestellt)
   (2) DSA und Elgamal
   (3) DSA (nur signieren/beglaubigen)
   (4) RSA (nur signieren/beglaubigen)
  (14) Vorhandener Schlüssel auf der Karte
Ihre Auswahl? 4
RSA-Schlüssel können zwischen 1024 und 4096 Bit lang sein.
Welche Schlüssellänge wünschen Sie? (3072)
Die verlangte Schlüssellänge beträgt 3072 Bit
Bitte wählen Sie, wie lange der Schlüssel gültig bleiben soll.
         0 = Schlüssel verfällt nie
      <n>  = Schlüssel verfällt nach n Tagen
      <n>w = Schlüssel verfällt nach n Wochen
      <n>m = Schlüssel verfällt nach n Monaten
      <n>y = Schlüssel verfällt nach n Jahren
Wie lange bleibt der Schlüssel gültig? (0)
Schlüssel verfällt nie
Ist dies richtig? (j/N) j

GnuPG erstellt eine User-ID, um Ihren Schlüssel identifizierbar zu machen.

Ihr Name: Claus Dietrich
Email-Adresse: claus.dietrich@exampledomain.de
Kommentar: For signing AppImages
Sie haben diese User-ID gewählt:
    "Claus Dietrich (For signing AppImages) <claus.dietrich@exampledomain.de>"

Ändern: (N)ame, (K)ommentar, (E)-Mail oder (F)ertig/(A)bbrechen? F
Wir müssen eine ganze Menge Zufallswerte erzeugen.  Sie können dies
unterstützen, indem Sie z.B. in einem anderen Fenster/Konsole irgendetwas
tippen, die Maus verwenden oder irgendwelche anderen Programme benutzen.
gpg: Schlüssel B3603E9243F6E126 ist als ultimativ vertrauenswürdig gekennzeichnet
gpg: Widerrufzertifikat wurde als '/home/claus/.gnupg/openpgp-revocs.d/84A01D03A8E60CB0493A6C4F-
B3603E9243F6E126.rev' gespeichert.
Öffentlichen und geheimen Schlüssel erzeugt und signiert.

Bitte beachten Sie, daß dieser Schlüssel nicht zum Verschlüsseln benutzt werden kann.  Sie können aber mit
dem Befehl "--edit-key" einen Unterschlüssel für diesem Zweck erzeugen.
pub   rsa3072 2023-06-23 [SC]
      CB0493A6C4FB3603E9243F6E12684A01D03A8E60
uid                      Claus Dietrich (For signing AppImages) <claus.dietrich@exampledomain.de>
claus@claus-VirtualBox:~$
```

Merken Sie sich das verwendete Passwort, denn es wird noch benötigt.

Tragen Sie den Schlüssel 'CB0493A6C4FB3603E9243F6E12684A01D03A8E60' jetzt – wie nachfolgend beschrieben – in den Dialog ein.

#### 11.10.3.4.3 Signieren eines AppImages (Gambas >= 3.20.0)

Um Ihr AppImage zu signieren, müssen Sie es so erzeugen, wie bereits beschrieben. Lediglich im Dialog Nr. 4 müssen Sie die Option "Sign AppImage file with GPG key" markieren und den vorhandenen oder neu generierten GnuPG-Schlüssel angeben:

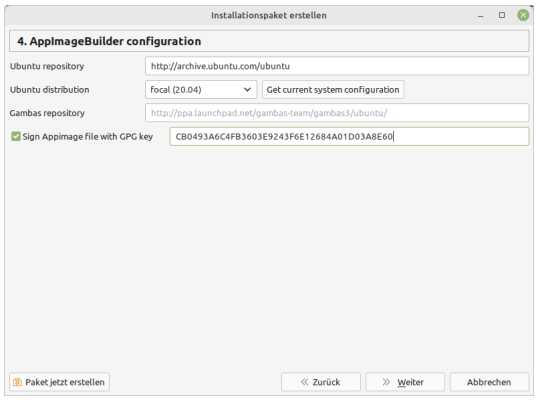


Abbildung 11.10.3.4.1: Eingabe des GnuPG-Schlüssels

Der Rest des Paket-Erzeugungsdialogs ist identisch. Allerdings werden Sie am Ende des Erzeugungsprozesses gebeten, das Passwort einzugeben, das bei der Erzeugung des Schlüssels verwendet wurde.

#### 11.10.3.4.4 Veröffentlichung der Signatur

Die Signatur des AppImages kann man sich mit folgendem Kommando anzeigen lassen und zusammen mit dem Programm veröffentlichen.

```
./TV_Recorder-x86_64.AppImage --appimage-signature
```

Die Signatur sieht beispielhaft so aus und sollte in dieser Form mit dem Programm veröffentlicht werden:

```
-----BEGIN PGP SIGNATURE-----

iQGzBAABCgAdFiEESKAeYDpsT7NgPpJD9uEmc6sjUo4FAmSbM9QACgkQ9uEmc6sj
Uo7AXAwAgaBVkxJA7ZOvY3WgrbiWyFVqkCbCGsrI/ztlzG1ZmGoM1mxK1gSpzrKY
HJQQ+OFq9HQQgCeIn5gPRCumCiOw3qGPjhrABFVbzkQUZ8g8oR14bkQOH4vkKreC
eYVAskgVfgJ6FvJwwwwCICoHZAAOjjBrirGh5J/MLggQFv5gGLcIzj2Ak2nOXmEP
XTwRNDfj+TULvVb2UhmmppKjbVKnbudnx0WWovczbzsgoO6tvC6T1y2LILgchUxZ
hS0xsgyn0BpjusZmjGrffkfm4EoMRkuEH5Jj6tadMmw69VhxpEIevDbmUes8HZs0
pZ3DTtFnnzpCHiRvWAIAiH9SwoZr4780/bnedJFbI4ThGmI83chJudvfSIr3mwTu
a47Q+hGHWow8/ouBt0O3QiiEObhRw78W4vjNL6dhNjfC6a3hXOGvq+pfb3kRZuZT
kiMUVR0lDtx6aFBvSQasgKkBzgLnxn8gJrR9N2l3VWqpNQzMqb1GcAuTWH7Ie9Xg
gW+00mBX
=TC2c
-----END PGP SIGNATURE-----
```

#### 11.10.3.4.5 Überprüfung der Signatur

Als Anwender des AppImage können Sie nun mit Hilfe der Signatur die Authentizität des AppImages überprüfen.

Zu diesem Zweck führen Sie das gleiche Kommando aus wie der Erzeuger des AppImages:

```
./TV_Recorder-x86_64.AppImage --appimage-signature
```