UNICON

eLux SDK VM für eLux RP 6

Virtuelle Maschine zur Software-Entwicklung auf der Basis von eLux und zur Erstellung von eLux-Paketen mit dem eLux Builder Kit

Stand: 2023-07-21

0. Rechtliche Hinweise

Unicon GmbH
Ludwig-Erhard-Allee 26
76131 Karlsruhe
+49 (0)721 96451-0

1. Über die eLux SDK VM

Die **eLux SDK VM** ist eine Virtuelle Maschine, die wir mit Software Development Kit-Tools für eLux RP 6 (64-Bit) ausstatten und unseren Kunden auf Anfrage zur Verfügung stellen. Es handelt sich um ein Linux-System als VMware-Image und ist lauffähig unter allen VMware Server-Versionen.

Wir stellen eine Version der **eLux SDK VM** zur Verfügung, die der aktuellen eLux-Version entspricht. [1]

Version	Beschreibung	Bereitstellung
eLux SDK VM 6.2302.2	enthält das **eLux Builder Kit** für eLux RP 6 2302.2	als `.ova`-Vorlage (Open Virtual Appliance) Installieren Sie die Maschine innerhalb Ihrer VMware-Umgebung aus der Vorlage, beispielsweise über die Option **OVF-Vorlage bereitstellen**
eLux SDK VM 3.12	enthält das **eLux Builder Kit** für eLux RP 6 2209	
epkg für eLux SDK VM 6.x	Kommandozeilen-Werkzeug zur Erstellung von eLux RP 6-Paketen	Installieren Sie die **epkg**-Pakete auf einem Ubuntu-System.

Das **eLux-Builder-Kit** ist eine Entwicklungsumgebung, mit der erfahrene Linux-Entwickler oder -Administratoren beliebige Software als `.epm`- und `.fpm`-Dateien paketieren können. Dazu kann auch eigener Quellcode und/oder Kernel-Treiber kompiliert werden.

Mit dem **eLux Builder Kit** erstellte Pakete können Sie mit Hilfe von ELIAS in eine Imagedefinitions-Datei (`.idf`) einbinden.

Alternativ steht das Kommandozeilen-Werkzeug **epkg** zum Paketieren zur Verfügung. Mit **epkg** können Sie die Paket-Erstellung automatisieren.

Folgende Kenntnisse werden vorausgesetzt

- Netzwerk- und Kommunikations-Hardware
- Administration von Linux-Betriebssystemen
- Administration von eLux
- ELIAS-Funktionen

Anmeldung

Folgende Systemkonten sind angelegt:

root - elux
elux - elux

Das Home-Verzeichnis für das Konto **elux** ist `/home/elux`.

[1] Die eLux SDK VM 3.12 entspricht beispielsweise eLux RP 6 2209

- Melden Sie sich in der VMware-Konsole oder über **ssh** mit dem Konto **elux** an.

Hinweis
Alternativ wählen Sie die Menüoption **Applications > Internet > X11VNC Server** und verwenden einen VNC-Viewer.

Die virtuelle Maschine verwendet standardmäßig DHCP.

eLux-Pakete erstellen

Nach dem Systemstart wird der Mate-Desktop geladen und Sie werden automatisch mit dem Konto **elux** angemeldet.

- Verwenden Sie das **eLux Builder Kit**. Um mit dem Programm vertraut zu werden, erstellen Sie die Pakete des Beispiel-Workspaces. Für weitere Informationen siehe Beispiel-Anwendung.

Alternativ steht das Kommandozeilen-Programm **epkg** zur Verfügung. Für weitere Informationen siehe Das epkg-Werkzeug.

Windows Shared Folder (Samba)

Das **samba4**-Server-Paket ist auf dem Image installiert, aber noch nicht konfiguriert. Für weitere Informationen zur Konfiguration siehe https://help.ubuntu.com/lts/serverguide/samba-fileserver.html

oder lesen Sie die **smb.conf**-Anleitung in der Kommandozeile: `man smb.conf`.

1.1. Kernel-Entwicklung

Der Kernel ist gepatcht und vorkonfiguriert. Die kompilierten Kernel-Module finden Sie als Debian-Pakete unter `/usr/src/`. Für eLux RP 6 wird nur noch die 64 Bit-Version unterstützt.

1.2. Kompilieren eines Kernel-Moduls

- ab **eLux SDK VM 3.5** -

Im Verzeichnis `/home/elux/examples/compiling/kernel_module/` befindet sich ein **make** file. Verwenden Sie den Befehl **make**, um ein Kernel-Modul zu bauen, das Sie in eLux laden können.

> Achtung Das Kernel-Modul steht in Abhängigkeit zum Kernel der eLux-Version. Beispielsweise funktioniert ein mit der **eLux SDK VM 3.11** erstelltes Kernel-Modul nur mit eLux RP 6 2204.

1.3. Systempakete

Das System für eLux RP 6 basiert auf Ubuntu 16.04 "Xenial".

Alle Pakete, die von Unicon modifiziert oder zurückportiert wurden, sind in diesem Image enthalten. Jedes zusätzliche Paket kann auf dem üblichen Debian-Weg (**apt-get** oder **aptitude**) einbezogen werden.

1.4. Kommandozeilen-Werkzeug

- nur für SDK VM 3.x -

epkg ist ein Kommandozeilen-Werkzeug, mit dem Sie EPM- und FPM-Pakete paketieren können und mit dem Sie den Prozess automatisieren können.

In unseren Beispielen finden Sie unter `/home/elux/workspace/examples/build` das Skript `build.sh`, mit dem Sie das Beispiel-Paket erstellen können.

Konvertierung von Paketen

Während des Build-Prozesses legt **epkg** einen **eLux Builder Kit**-Workspace an, so dass Sie die Metadaten der Pakete bequem bearbeiten können. Damit das eLux Builder Kit von **epkg** genutzt werden kann, muss der eLux Builder Kit-Paketbaum zunächst konvertiert werden. Diese Funktionalität bringt **epkg** mit: Im Beispiel-Projeket finden Sie einen **convert**-Ordner und ein Skript für die Konvertierung:

`/home/elux/workspace/examples/convert/convert.sh`

Nach dem Erstellen unseres Beispielpaketes finden Sie den erzeugten Workspace unter `/home/elux/workspace/examples/build/tcpdump/output/.ebkworkspace`.
Um den Workspace leicht öffnen zu können, haben wir auch einen *recently used*-Eintrag im **File**-Menü gesetzt.

Sie können nun die Metadaten zu den Paketen im eLux Builder Kit bearbeiten und dann den finalen Build-Prozess starten. Die erzeugten Paketdateien (`.epm`, `.fpm`) finden Sie in folgendem Ausgabe-Verzeichnis:

`/home/elux/workspace/examples/build/output/container/UC_RP6_X64-1.0-1/`

epkg bietet auch die Möglichkeit, Pakete zu installieren. Dazu werden sie vom Ausgabe-Verzeichnis in ein definiertes Container Verzeichnis kopiert (Beispiel: `/home/containers`).

Die Datei `/usr/share/epkg/settings.ini` enthält die Konfiguration für die Installation und für das Signieren von Paketen.

Für weitere Informationen siehe Das epkg-Werkzeug.

1.5. Beispiel-Pakete

Für das eLux Builder Kit haben wir einen Beispiel-Workspace angelegt, der ein EPM mit FPMs enthält:

---

debug: `/home/elux/examples/packaging/debug/output/devel`

Der Workspace enthält Beispiele für das Erstellen von eLux-Paketen aus Entwicklungswerkzeugen wie `gdb, file, strace, ltrace, valgrin, less, coredump, tcpdump, vim, libpython, systemd-debug-shell, eluxbuild`[1]

Beispiel-Pakete erstellen

1. Laden Sie den Beispiel-Workspace `.ebkworkspace` im angegebenen Verzeichnis. Sie finden den Workspace auch im Menü **File** als *recently used*-Eintrag.
2. Markieren Sie die oberste Ebene (das EPM).
3. Klicken Sie auf die Schaltfläche **Build**.

Alle Pakete im Workspace werden erstellt. Die fertigen Paket-Dateien (`.epm`, `.fpm`) finden Sie in folgendem Ausgabe-Verzeichnis:

`/home/elux/examples/packaging/devel/output/container/UC_RP6_X64-1.0-1/`

---

[1] ab eLux SDK VM 3.6

2. eLux Builder Kit starten

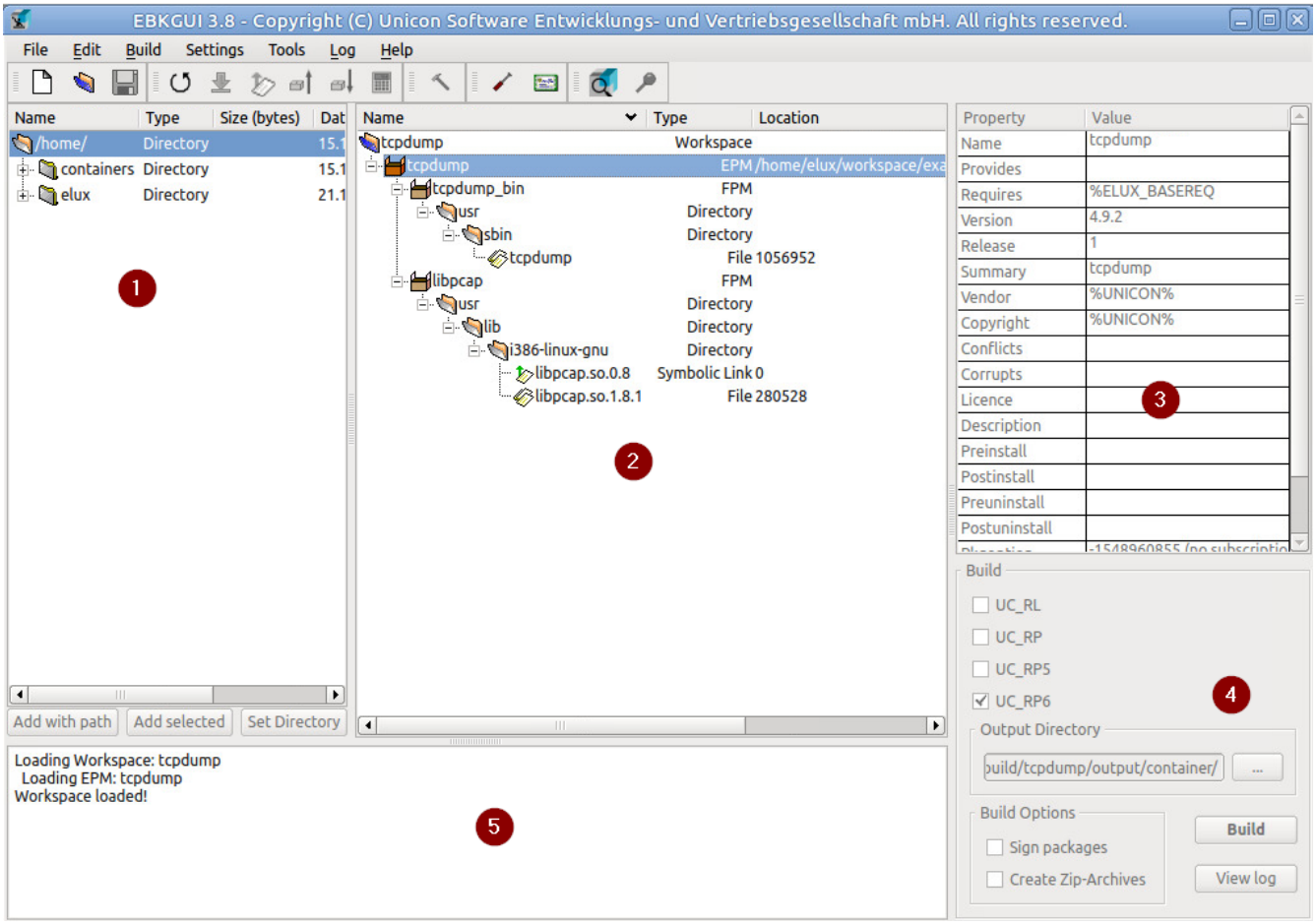
1. Doppelklicken Sie auf das Anwendungssymbol des eLux Builder Kit (EBKGUI) auf dem Desktop oder geben Sie im Kommandofenster den Befehl `ebkgui` ein.
2. Wenn Sie die Pakete signieren möchten, bearbeiten Sie die Zertifikatseinstellungen. Für weitere Informationen siehe eLux Builder Kit-Einstellungen.
3. Öffnen Sie einen vorhandenen oder erstellen Sie einen neuen Workspace.

Das eLux Builder Kit-Hauptfenster wird angezeigt.

- Machen Sie sich mit der Oberfläche vertraut (siehe Hauptfenster) und spielen Sie mit der Beispiel-Anwendung.
- Beginnen Sie mit Ihrem eigenen Projekt, indem Sie einen neuen Workspace anlegen. Anschließend definieren Sie Ihre Pakete, bearbeiten die Paket-Eigenschaften und die Build-Einstellungen, bevor Sie den Build-Prozess starten.

3. eLux Builder Kit - Oberfläche

3.1. Hauptfenster



Legende	
1	File Browser
2	Workspace Browser Zeigt die Struktur des aktuell geöffneten Workspace mit EPMs, FPMs, Verzeichnissen, Dateien und Symbolischen Links
3	Properties Editor Zeigt die Eigenschaften des im Workspace-Browser markierten Objektes mit **Eigenschaft** und **Wert** Die meisten Eigenschaften können Sie direkt im Feld oder über die Schaltfläche **...** im **Value**-Feld bearbeiten. Abgeblendete Eigenschaften können nicht verändert werden.
4	Build-Einstellungen
5	Ausgabe-Fenster

3.2. Menüfunktionen

Menü **File**

Option	Beschreibung
New workspace	Erzeugt einen neuen Workspace
Open workspace	Öffnet einen vorhandenen Workspace
Save	Speichert den aktuell geöffneten Workspace
Import EPM folder	Importiert ein EPM aus einem anderen Workspace in den geöffneten Workspace
Export selected EPM	Exportiert ein EPM aus dem Workspace in ein beliebiges Verzeichnis
Link to EPM folder	Verlinkt ein EPM aus einem anderen Workspace in den geöffneten Workspace
Exit	Schließt die eLux Builder Kit-Anwendung
Zuletzt geöffnete Workspaces	Die zuletzt geöffneten Workspaces werden zum schnellen Wieder-Öffnen angezeigt. Zusätzlich wird ein Beispiel-Workspace angezeigt.

Menü **Edit**

Option	Beschreibung
Reload Workspace	Lädt den aktuell geöffneten Workspace erneut
Strip File	Führt den Befehl **strip** auf die markierte Datei aus
Calculate FPM-Size	Berechnet die Größe und Anzahl der Dateien des markierten FPMs
Calculate all sizes	Berechnet die Größe und Anzahl der Dateien aller FPMs
Create Symbolic Link	Erstellt einen neuen Symbolischen Link für das markierte FPM oder Unterverzeichnis
Move FPM up	Verschiebt das markierte FPM um eine Position nach oben
Move FPM down	Verschiebt das markierte FPM um eine Position nach unten

Menü **Build**

Option	Beschreibung
Build Workspace	Erzeugt alle Pakete des aktuellen Workspace
Build selected	Erzeugt nur das Paket für das aktuell markierte Objekt Wenn der gesamte Workspace markiert ist, entspricht das dem Befehl **Build Workspace**. Wenn ein EPM markiert ist, werden das EPM und dessen FPMs erzeugt .

Menü **Settings**

Option	Beschreibung
Preferences	Öffnet den Dialog **Einstellungen**
Sign Settings	Öffnet den Dialog **Einstellungen** und springt zu den Einstellungen für die Signierung von Paketen

Menü **Tools**

Option	Beschreibung
EBKViewer	Startet den EBK-Viewer, wenn installiert
Fix FPM SORT_ ID	Behebt Probleme bei der Sortierreihenfolge von FPMs

Menü **Log**

Option	Beschreibung
Clear output window	Löscht den Inhalt des Ausgabe-Fensters
View build log file	Zeigt die Build-Protokolldatei des aktuell geöffneten Workspace
Delete build log file	Löscht die Build-Protokolldatei des aktuell geöffneten Workspace

3.3. Kontextmenüs

Im Workspace-Browser und im File-Browser sind Kontextmenüs verfügbar, die Ihnen den schnellen Zugriff auf häufig genutzte Befehle ermöglichen.

Kontextmenüs im Workspace Browser

Im Workspace-Browser ist jeweils ein Kontextmenü für EPMs, FPMs, Verzeichnisse und Dateien verfügbar.

Option (EPM)	Beschreibung
Add new FPM	Fügt ein neues FPM für das EPM an
Delete EPM	Löscht das EPM
Build EPM - but not its FPMs	Erzeugt nur das EPM **ohne** seine FPMs Diese Funktion ist nur im Kontextmenü verfügbar.
Hand down ...	Die Werte in den Eigenschaften **Version** und **Release** der untergeordneten FPMs werden mit den Werten des EPM überschrieben. Diese Funktion ist nur im Kontextmenü verfügbar.

Option (FPM)	Beschreibung
Add new Folder	Fügt ein neues Verzeichnis ein
Delete FPM	Löscht das FPM
Create Symbolic Link	Erstellt einen Symbolischen Link auf eine Datei oder ein Verzeichnis
Calculate size and files	Berechnet die Größe und Anzahl der Dateien
Move down/up	Verschiebt das markierte FPM um eine Position nach oben/unten

Option (Datei)	Beschreibung
Delete File	Löscht die Datei aus dem FPM
Strip	Führt den Befehl **strip** auf die markierte Datei aus
Edit	Öffnet die Datei zur Bearbeitung in einem externen Editor (siehe Voreinstellungen). Diese Option steht für jede Datei zur Verfügung, auch auf Binär-Dateien kann diese Funktion angewandt werden! Diese Funktion ist nur im Kontextmenü verfügbar.

Kontextmenüs im File Browser

Im File Browser ist jeweils ein Kontextmenü für Verzeichnisse und für Dateien verfügbar.

Option	Beschreibung	Verzeichnis	Datei
Add selected	Fügt das Objekt mit seinem Inhalt dem im Workspace-Browser markierten FPM/Verzeichnis hinzu	✓	✓
Add with path	Fügt das Objekt mit seinen übergeordneten Verzeichnissen dem im Workspace-Browser markierten FPM/Verzeichnis hinzu	✓	✓
Set as base dir	Definiert die oberste Ebene des File Browsers	✓	
Edit	Öffnet die Datei zur Bearbeitung in einem externen Editor		✓

3.4. File Browser

Mit Hilfe des File Browsers können Sie Dateien und Verzeichnisse zu einem FPM hinzufügen. Neben den Dateien und Verzeichnissen zeigt der Browser die Größe in Bytes, Datum/Zeit der letzten Änderung und die Rechte (im oktalen Zahlensystem) an.

Datei oder Verzeichnis zu einem FPM hinzufügen

Hinweis
Die maximale Größe für ein FPM beträgt 200 MB.[1] Wenn Sie mehr Plattenspeicher für Ihre Installation benötigen, erstellen Sie ein weiteres FPM.

1. Markieren Sie das relevante FPM im Workspace-Browser oder ein Verzeichnis, das sich bereits im FPM befindet
2. Um ein Objekt mit seinem Inhalt hinzuzufügen, wählen Sie im File Browser, im Kontextmenü der relevanten Datei oder des Verzeichnisses die Option **Add selected**.
3. Um ein Objekt mit den übergeordneten Verzeichnissen hinzuzufügen, wählen Sie im File Browser, im Kontextmenü der relevanten Datei oder des Verzeichnisses die Option **Add with path**.

Dateien und Verzeichnisse, die hinzugefügt werden, behalten die folgenden Eigenschaften:

- Datum und Zeit der letzten Änderung
- Besitzer
- Gruppe
- Dateirechte

Oberste Ebene des File Browsers festlegen

- Wählen Sie im Kontextmenü des relevanten Verzeichnisses die Option **Set as base dir**.

[1] ab eLux RP 6 2302 (vorher: 100 MB)

3.5. Paket-Eigenschaften im Properties Editor

Manche Paket-Eigenschaften (Metadaten) können direkt in den Feldern des **Properties Editors** bearbeitet werden. Andere Eigenschaften können nur mit speziellen Editoren angepasst werden. Für die markierte Eigenschaft wird dann eine Schaltfläche im **Value**-Feld angezeigt.

- Um eine Eigenschaft eines der EPMs oder FPMs zu bearbeiten, für die ein spezieller Editor hinterlegt ist, klicken Sie auf die Schaltfläche [...].

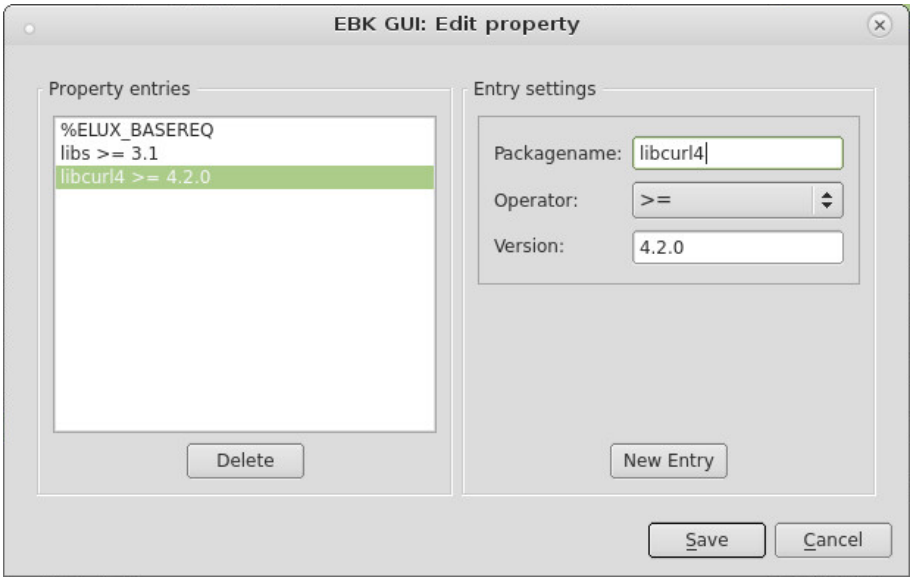
Description, Pre-/Postinstall, Pre-/Postuninstall und Comment

Für diese Eigenschaften ist ein einfacher Texteditor hinterlegt.

- Fügen Sie Text ein und bestätigen Sie mit **Save**.

Requires und Conflicts

Hinweis
Die Eigenschaft **Requires** eines EPMs enthält immer einen Eintrag, der festlegt, dass das EPM ein installiertes eLux BaseOS voraussetzt. Der Eintrag lautet: `%ELUX_BASEREQ`



Option	Beschreibung
Listenfeld **Property entries**	Listet die bisher definierten Einträge für **Requires** oder **Conflicts** auf. Format: *Paketname Operator Version* Ein markierter Eintrag kann im Bereich **Entry settings** bearbeitet werden.
Delete	Der markierte Eintrag wird gelöscht.
New Entry	Erstellt einen neuen Eintrag `NewEntry`

Option	Beschreibung
Packagename	Name des Eintrags Der Name muss einem anderen existierenden Paket entsprechen.
Operator	Operator, der auf die Version angewendet wird Wenn die Version des Pakets irrelevant ist, wählen Sie den Operator <none>. Der Eintrag wird dann auf alle Versionen des Paketes in **Packagename** angewendet.
Version	Legt die Version des in **Packagename** angegebenen Paketes fest

License

Um eine gültige Lizenz für Ihr Paket zu erhalten, kontaktieren Sie bitte Unicon per E-Mail: sales@unicon.com.

Option	Beschreibung
Save	Schaltfläche wird erst aktiviert, wenn alle drei Eingabefelder korrekt ausgefüllt sind.
Clear	Löscht die aktuelle Lizenz Um die Lizenz endgültig zu entfernen, klicken Sie auf **Save**.

Containers-Editor

Mit diesem Editor legen Sie fest, für welche Container das EPM oder FPM erstellt werden soll. Die Einstellung betrifft nur das aktuelle Paket.

Installoption

- nur für FPMs -

Option	Beschreibung
Mandatory	FPM muss installiert werden
Install	Sie können zwischen zwei Unter-Optionen wählen: **as default** - Das FPM wird standardmäßig installiert, kann aber über ELIAS abgewählt werden. **optional** - Das FPM wird standardmäßig **nicht** installiert, kann aber über ELIAS zur Installation angewählt werden.
Force uninstall instead of update	- unabhängig von **Mandatory** oder **Install** - Vor der Installation dieses FPM-Paketes, wird das alte FPM zunächst deinstalliert (falls vorhanden) statt eine Update-Installation des Pakets durchzuführen.

3.6. Build-Bereich

Im Build-Bereich legen Sie Optionen fest, beispielsweise ob die Pakete signiert und in ein ZIP-Archiv gepackt werden. Diese Einstellungen werden im Workspace selbst gespeichert und beim Laden eines Workspace gesetzt. Die Pakete werden in die entsprechenden Container-Unterverzeichnisse des Ausgabeverzeichnisses kopiert.

Option	Beschreibung
Build – Container	Ziel-Container, für die die Pakete des Workspaces erstellt werden
Output Directory	Zielverzeichnis, in das die fertig erstellten Pakete, die Signaturen und das ZIP-Archiv kopiert werden. eLux Builder Kit erzeugt im Ausgabeverzeichnis automatisch Unterverzeichnisse mit dem Container als Namen.
Build Options	**Sign packages** – Die erstellten Pakete sollen signiert werden und werden in ELIAS auf eine gültige Signatur geprüft. nur verfügbar, wenn in den Einstellungen der Pfad zum Signierungsprogramm **eluxsign** gesetzt ist und Schlüssel und Zertifikat zum Signieren angegeben sind **Create ZIP archives** – Für jedes EPM wird in ein separates ZIP-Archiv erzeugt. Das Archiv enthält das EPM, alle FPMs und die Signaturen für das EPM und die FPMs.
Build	Startet den Build-Prozess für das markierte EPM oder für das markierte FPM. Wenn im Workspace-Browser der gesamte Workspace markiert ist, wird der Build für alle Pakete durchgeführt.
View Log	Zeigt die Protokolldatei für den Workspace an Die Datei enthält Meldungen über die Erstellung der Pakete in ausführlicher Form.

3.7. Ausgabefenster

Das Ausgabefenster zeigt Statusmeldungen an, beispielsweise

- Fortschritt des aktuellen Erstellungsvorgangs
- Erzeugen eines neuen Workspace
- Laden/Speichern eines Workspace

Meldungen im Ausgabefenster löschen

- Wählen Sie die Menüfunktion **Log > Clear output window**.

4. Eigene Pakete im eLux Builder Kit erstellen

Um Ihre eigenen Pakete zu erstellen, legen Sie zunächst einen neuen Workspace an, in dem Sie anschließend Ihre EPM- und FPM-Pakete definieren und Metadaten festlegen.

4.1. Neuen Workspace anlegen

1. Wählen Sie **File > New Workspace**.
2. Bearbeiten Sie folgende Felder:

Option	Beschreibung
Workspace Name	Name für den neuen Workspace
Workspace Directory	Zielverzeichnis für den neuen Workspace
Use workspace name to create a subdirectory	Im ausgewählten Zielverzeichnis wird ein Unterverzeichnis mit dem Namen des Workspace angelegt. Der Workspace wird dann in das Unterverzeichnis gespeichert. Achtung: Wenn Sie die Option deaktivieren, muss das Zielverzeichnis leer sein.

3. Bestätigen Sie mit **OK**.

Der neue Workspace wird angelegt und in den Workspace-Browser geladen.

- Sie können nun EPMs und FPMs erstellen, siehe Neue Pakete definieren.

4.2. Neues Verzeichnis anlegen

1. Markieren Sie im Workspace-Browser das relevante FPM oder ein Unterverzeichnis des FPM.
2. Klicken Sie auf **Add**.
   Oder: Wählen Sie im Kontextmenü die Option **Add new folder**.
3. Geben Sie einen Namen für das Verzeichnis ein und bestätigen Sie mit **Finish**.

- Sie können nun weitere Verzeichnisse anlegen oder mit Hilfe des File Browsers Dateien zu diesem Verzeichnis hinzufügen.

4.3. Neue Pakete definieren

EPMs und FPMs werden einem bestehenden Workspace als Objekte hinzugefügt.

Regeln zur Benamung von Paketen

- Erlaubte Zeichen sind `a-z`, `A-Z`, `0-9`, `_`
  Wir empfehlen, Großbuchstaben zu vermeiden.
- Paketnamen müssen eindeutig sein. Ein Paketname darf nur einmal vorkommen, unabhängig davon ob es sich um ein EPM oder FPM handelt.
  Beachten Sie, dass ein eindeutiger Paketname auch relevant ist für **provides**, **conflicts** and **requires**.
- Paketnamen setzen sich aus folgenden Elementen zusammen:
  <name>-<version>-<release>.<container>.[epm|fpm]
  Beispiel: `baseosrp-6.2302.0-3.UC_RP6_X64.epm`
- Nach dem Build-Prozess dürfen Paket-Dateien nicht mehr umbenannt werden.

Regeln zur Benamung von udev-Regeln

- udev-Regeln werden in Dateien mit Dateierweiterung `.rules` gespeichert.
- Um Konflikte zu vermeiden, müssen Dateinamen für udev-Regeln eindeutig sein. Wir empfehlen, dass Sie den zugehörigen Paketnamen im Dateinamen verwenden.
- Um die Reihenfolge der Abarbeitung zu steuern, beginnen Sie den Dateinamen mit einer zweistelligen Zahl, maximal mit `98`.
  Die Regeldateien werden in alphanumerischer Reihenfolge abgearbeitet (höhere Zahlen später als niedrige).

  Beispiel für den Aufbau eines Dateinamens:
  <number>-<origin>-<package name>.rules
  Eine udev-Regeldatei von Unicon ist beispielsweise `91-unicon-keyboard.rules`

Für weitere Informationen über udev-Regeln siehe https://linux.die.net/man/7/udev.

4.3.1. EPM hinzufügen

1. Markieren Sie im Workspace-Browser das Workspace-Objekt (oberste Ebene).
2. Klicken Sie auf **Add**.
   Oder: Wählen Sie im Kontextmenü die Option **Add new EPM**.
3. Geben Sie einen Namen für das EPM ein und bestätigen Sie mit **Finish** ein.

- Bearbeiten Sie anschließend die Eigenschaften des EPM im Properties-Editor, siehe Paket-Eigenschaften im Properties Editor.

Sie können auch weitere EPMs anlegen oder FPMs für dieses EPM erstellen.

4.3.2. FPM hinzufügen

1. Markieren Sie im Workspace-Browser das relevante EPM.
2. Klicken Sie auf **Add**.
   Oder: Wählen Sie im Kontextmenü die Option **Add new FPM**.
3. Geben Sie einen Namen für das FPM ein und bestätigen Sie mit **Finish**.

- Um dem FPM Objekte hinzuzufügen, verwenden Sie den File Browser. Für weitere Informationen siehe File Browser.
- Bearbeiten Sie anschließend die Eigenschaften des FPM im Properties-Editor, siehe Paket-Eigenschaften im Properties Editor.

Sie können aber auch weitere FPMs anlegen oder dem FPM einen Symbolischen Link hinzufügen.

4.4. Symbolischen Link erzeugen

Ein **Symbolischer Link** ist ein Zeiger auf eine Datei oder ein Verzeichnis, das an einem anderen Ort liegt. Einen Symbolischen Link können Sie nur direkt in einem FPM oder in einem seiner Unterverzeichnisse erzeugen.

1. Markieren Sie das relevante FPM oder eines seiner Unterverzeichnisse.
2. Klicken Sie in der Symbolleiste auf das Symbol **Symbolischer Link**.
   Oder: Wählen Sie im Kontextmenü die Option **Create symbolic link**.
3. Bearbeiten Sie die folgenden Felder:

Option	Beschreibung
Name of the symbolic link	Name für den symbolischen Link Standardmäßig wird der Name automatisch aus der Eingabe in [2] bzw. [3] übernommen. Um selbst einen Namen einzugeben oder den automatisch vergebenen anzupassen, deaktivieren Sie die Option **Generate symbolic name automatically**.
Enter the target to which the symbolic link will point	Geben Sie das Linkziel ein. Wenn der Link auf eine Datei oder auf ein Verzeichnis innerhalb des FPMs verweisen soll, verwenden Sie **Select link target**.
Select link target	Zeigt eine Liste aller Dateien und Verzeichnisse des FPM an

4. Bestätigen Sie mit **OK**.

Der Link wird erzeugt und im relevanten Unterverzeichnis angezeigt. Wenn Sie den Link markieren, können Sie im Properties-Editor das Ziel (Target) überprüfen.

5. eLux Builder Kit-Einstellungen

Im Dialog **Settings > Preferences** können Sie verschiedene Voreinstellungen vornehmen. Beispielsweise können Sie die Zertifikate zur Signierung von Paketen konfigurieren (Register **Build**).

Wenn Sie ungültige Einstellungen vornehmen, erhalten Sie eine Meldung. Für weitere Informationen siehe Fehlerhafte Einstellungen.

5.1. Register Global

Add a comment to the Workspace, EPM or FPM when adding or deleting items

eLux Builder Kit fügt automatisch einen Kommentar im übergeordneten Element (Workspace, EPM oder FPM) hinzu, wenn Sie eine der folgenden Aktionen ausführen:

Aktion	
Hinzufügen	...einer Datei zu einem FPM
	...eines Verzeichnisses zu einem FPM
Löschen	...eines EPM von einem Workspace
	...eines FPM von einem EPM
	...einer Datei aus einem FPM
	...eines Verzeichnisses aus einem FPM
Anlegen	...eines EPM in einem Workspace
	...eines FPM in einem EPM
Importieren	...eines EPM aus einem anderen Workspace in den aktuellen Workspace
Linken	...eines EPM aus einem anderen Workspace in den aktuellen Workspace

Die Kommentare haben das folgende Format:

DATE - TIME: [ACTION] ELEMENTTYPE <ELEMENTNAME> (USER)

Parameter	Beschreibung
DATE	aktuelles Datum
TIME	aktuelle Zeit
ACTION	Art der Aktion (ADD, CREATE, DELETE, IMPORT, LINK)
ELEMENTTYPE	Typ des Elements auf dem die Aktion ausgeführt wurde (EPM, FPM, FILE, DIRECTORY)
ELEMENTNAME	Name des Elements
USER	Name des Benutzers, der die Aktion durchgeführt

Weitere Optionen

Option	Beschreibung
Increment release of EPMs/FPMs when they were changed	Die Release-Nummer eines EPM oder FPM wird automatisch erhöht, wenn Sie mindestens eine Eigenschaft des EPM/FPM geändert haben. Die Erhöhung der Release-Nummer wird nicht sofort durchgeführt, sondern beim ersten Speichern des Workspace bzw. beim Anklicken der **Build**-Schaltfläche. Bevor das eLux Builder Kit die Release-Nummer erhöht, wird eine Meldung angezeigt. Wenn Sie mit **Yes** bestätigen, werden die Release-Nummern aller geänderter EPM/FPM um einen Zähler erhöht. **Enschränkungen** Das Hinzufügen von Dateien oder Verzeichnissen zu einem FPM wird vom eLux Builder Kit nicht als eine Änderung registriert. Die vorherige Release-Nummer des geänderten EPM/FPM muss eine Zahl sein. Wenn die vorherige Release-Nummer keine Zahl ist, setzt das eLux Builder Kit eine `0` ein.
Open last workspace on startup	Beim Starten von eLux Builder Kit wird automatisch der zuletzt geöffnete Workspace geladen.
Use external text editor	Textdateien werden in einem externen Editor bearbeitet. Standardmäßig öffnet eLux Builder Kit eine **Xterm**-Shell und lässt den **vi**-Editor aufrufen. Alternativ können Sie über die Browse-Schaltfläche einen Editor auswählen oder in der Kommandozeile einen anderen Editors aufrufen. **Einschränkungen:** eLux Builder Kit kann nicht zwischen Text-Dateien und Binär-Dateien unterscheiden. Dadurch ist es möglich, alle Dateien im angegebenen Editor zu öffnen. Der verwendete Editor muss als Übergabeparameter die zu bearbeitende Datei akzeptieren.
Defaults	Setzt Einstellungen auf die Standardwerte für das aktuelle Register zurück

5.2. Register Build

Option	Beschreibung
Strip command	Parameter für den **strip**-Befehl auf eine Datei
Zip location	Pfad zum **zip**-Programm
eluxbuildrc location	Pfad zur Datei `eluxbuildrc` Beachten Sie, dass eLux Builder Kit auf diese Datei Schreibrechte benötigt!
Temp folder	Verzeichnis für temporäre Dateien

Option	Beschreibung
eluxsign location	Pfad zum **eluxsign**-Programm **eluxsign** signiert die von Ihnen erstellten Pakete.
Signature Key	Pfad zur Schlüssel-Datei
Signature Certificate	Pfad zur Zertifikat-Datei

Hinweis
Für die Paket-Signierung müssen alle drei Signatur-Einstellungen (**eluxsign location**, **Signature Key**, **Signature Certificate**) korrekt angegeben werden. Wenn eine Angabe fehlerhaft ist, wird die Signierung deaktiviert. Verwenden Sie den PKCS#6-Standard mit PEM-Kodierung (Dateiformate `.pem`/`.crt`). Für weitere Informationen siehe auch Zertifikate zur Paketverifizierung vorbereiten im **ELIAS 18**-Handbuch.

5.3. Register EPM/FPM

Option	Beschreibung
Use default properties	eLux Builder Kit übernimmt die unter **Copyright** und **Vendor** eingetragenen Werte in die entsprechenden Eigenschaften beim Anlegen neuer EPMs oder FPMs.
Copyright	Text, der für jedes neue EPM oder FPM als **Copyright**-Eigenschaft eingetragen wird
Vendor	Text der für jedes neue EPM oder FPM als **Vendor**-Eigenschaft eingetragen wird

5.4. Fehlerhafte Einstellungen

Wenn Sie fehlerhafte Einstellungen vornehmen und mit **OK** bestätigen, wird eine Meldung angezeigt und die fehlerhaften Einstellungen werden rot eingefärbt.

> **Achtung** Wenn Sie auf **Cancel** klicken, werden die fehlerhaften Einstellungen verworfen und die alten Einstellungen verwendet.

1. Bestätigen Sie mit **OK**.
2. Korrigieren Sie die fehlerhaften Einstellungen.
3. Klicken Sie auf **OK**, um die Einstellungen zu speichern.

Option	Wert muss gültig sein	kein Wert erlaubt
Strip command	nein	ja
Zip location	ja	nein
elxubuildrc	ja	nein
Temp folder	ja	nein
eluxsign location	ja	ja
Signature Key	ja	ja
Signature Certificate	ja	ja

6. Das epkg-Werkzeug

- für SDK VM 3.x -

epkg ist ein Kommandozeilen-Werkzeug, mit dem Sie eLux-Pakete im `.fpm`- und `.epm`-Format paketieren können. Die folgenden Module werden unterstützt:

Build	EPM- und FPM-Pakete erstellen
Convert	Alte Metadatei-Version in die aktuelle Version konvertieren
ConvertFrom	Paketbaum in die aktuelle Metadatei-Version konvertieren
Install	EPM und FPMs in das Container-Verzeichnis kopieren
New	Leere Meta-Repository-Dateien für gegebene EPM- und FPM-Namen erstellen
NewFrom	Leere Meta-Repository-Dateien aus einem Paketbaum erstellen
Info	Paket-Metadaten beziehen
Size	Unkomprimierte Größe berechnen oder eine `eluxmanlog`-Datei analysieren, um die komprimierte Größe zu erhalten
UpdateCopyrightInfo	Copyright-md5-Summe aktualisieren

6.1. epkg verwenden

epkg bietet verschiedene Funktionen zur Erstellung und Verwaltung von eLux-Paketen. Die Funktionen werden in mehreren Modulen zur Verfügung gestellt.

- Um beispielsweise ein EPM- oder FPM-Paket zu erstellen, verwenden Sie das **Build**-Modul. Beispiel:

  ```
  epkg Build --container=rp6_x64
  ```

Optionen für das Erstellen eines Builds

- Um eine Option einzuleiten, verwenden Sie zwei Minuszeichen:

  ```
  epkg Build --<option>
  ```

Option	Beschreibung
`container`	Typ des Ziel-Containers, für den das Paket gebaut wird (eLux Hauptversion) Erlaubte Werte: `rp6`, `rp6_x64`, `elux7`
`skipCheck`	Aktualisierung der md5-Prüfsummen überspringen
`skipCreateTree`	Erstellen des package tree überspringen
`skipCreateEpm`	Erstellen von EPMs und FPMs überspringen

Option	Beschreibung
`skipSign`	Signieren des Paketes überspringen
`autoIncrementVersion`	Release-Version geänderter EPMs und FPMs automatisch hochzählen
`releaseType`	Release-Typ eines Builds (Test oder Release), siehe unten
`testVersion`	Versionsbezeichnung, die für Test-Builds verwendet wird Erlaubte Werte: `'[a-z]+(+[0-9]+])?'`
`debug`	Debug-Modus aktivieren

Hilfe und weitere Befehle

`epkg --help`	Allgemeine Hilfe-Seite über alle Module
`epkg <ModuleName> --help`	Hilfe-Seite eines bestimmten Moduls Erlaubte Werte: `Build, Convert, ConvertFrom, Install, Upload, New, NewFrom, Info, Sign, UpdateCopyrightInfo`
`epkg --<ModuleName>`	epkg-Modul, das ausgeführt werden soll Erlaubte Werte: `Build, Convert, ConvertFrom, Install, Upload, New, NewFrom, Info, Sign, UpdateCopyrightInfo`
`epkg --log`	Protokollierung aktivieren

Test- versus Release-Builds

Mit **epkg** können Sie sowohl Release-Pakete als auch Test-Pakete erstellen. Die Art des Builds wird mit dem Parameter `--releaseType` gesteuert.

`epkg ---releaseType=test`	Test-Build (Standard)
`epkg ---releaseType=release`	Release-Build

Ein Test-Build vergibt in der Paketbezeichnung die mit `testVersion` definierte Bezeichnung. An den Paketnamen wird die zugrundeliegende Release-Nummer und danach die durch `testVersion` definierte Zeichenfolge angehängt, standardmäßig `testing`.[1] Frühere Versionen haben die vorhandene Release-Nummer standardmäßig durch die Zeichenfolge `9999` ersetzt.

Test-Builds unterscheiden sich von Release-Builds außerdem in folgenden Merkmalen:

- Einige Schritte wie die Aktualisierung der md5-Prüfsumme werden übersprungen.
- Pakete, die durch Test-Builds erstellt werden, können durch die Installation im Container überschrieben werden.

Für Beispiele siehe Anwendungsbeispiele.

[1] ab eLux SDK VM 3.12 (eLux RP 6 2209)

6.2. epkg konfigurieren

Voreinstellungen für **epkg** werden in der Konfigurationsdatei `/etc/epkg/settings.ini`[1] definiert.

Beispiel:

```
[Signing]
keyPath=/etc/epkg/code-signing-key.pem
certificatePath=/etc/epkg/code-signing-cert.pem

[Global]
DevelopmentContainer=/home/containers/
Blocksize=1024
SizeOffsetInPercent=2
UniconCopyright=Copyright (C) %currentYear% SampleTec. All rights reserved.
Proxy=http://proxy.sampletec-01.com:3800

[WebElias]
Url=https://elias.test.sampletec-01.com
```

DevelopmentContainer

Definieren Sie unbedingt den `DevelopmentContainer`. Dieser Container wird als Ziel für das **Install**-Modul verwendet. Wenn ein Paket unverändert bleibt, wird das alte FPM aus diesem Container zur Erstellung der `.zip`-Datei verwendet.

- Setzen Sie folgenden Eintrag in der `settings.ini`-Datei:

```
[Global]
DevelopmentContainer=<Container-Pfad>
```

Signierung

Bei der Installation wird automatisch ein Self-signed-Zertifikat erstellt und in der Konfigurationsdatei angegeben.[2] Damit Sie Pakete in älteren SDK-Versionen[3] signieren können, müssen Sie selbst für das Zertifikat sorgen und Zertifikatdatei und Schlüsseldatei angeben.

- Setzen Sie folgende Einträge in der `settings.ini`-Datei:

```
[Signing]
keyPath=<Pfad und Dateiname>
certificatePath=<Pfad und Dateiname>
```

---

[1] ab eLux SDK VM 3.12 (eLux RP 6 2209). In früheren Versionen `/usr/share/epkg/settings.ini`
[2] ab eLux SDK VM 3.12 (eLux RP 6 2209)
[3] bis eLux SDK VM 3.11

Verwenden von Debian Repositories

Hinweis
Die Verwendung von Debian-Repositories ist standardmäßig deaktiviert.

Wenn Sie möchten, dass **epkg** Pakete aus Debian Repositories als Quellen für EPMs und FPMs verwendet werden, müssen die Repositories in einer speziellen Konfigurationsdatei definiert werden.

- Bearbeiten Sie die Datei `/usr/share/unicon/suiteCPP.ini` und setzen Sie die relevanten Einträge.

Die `.ini`-Datei hat die folgende Struktur:

```
[source<i>]
name=
base=
architecture=
source<j>=
```

source<i>	Jeder Abschnitt ist ein Repository, das referenziert werden kann. `<i>` muss mit jedem neuen Abschnitt hochgezählt werden, der Startwert ist `0`.
name	Der Name eines Repository wird als Schlüssel für andere Referenzen verwendet.
base	Ein Repository kann auf einem anderen Repository aufbauen. Das bedeutet, dass Ihr Repository die Debian-Quellen des Base-Repository einschließt. Verwenden Sie den Namen des Base Repository als Wert.
architecture	Debian-Architektur im Format einer Komma-separierten Liste Beispiel: `amd64`
source<j>	Kann beliebig oft innerhalb eines Abschnittes verwendet werden <j> muss mit jeder neuen Quelle hochgezählt werden, der Startwert ist `1`. Jede Quelle repräsentiert eine Debian-Quelle, um nach Paketen zu suchen, und hat das folgende Format: `<priority> deb <repository URL> <distribution> <component>`

priority	apt Pin-Priority, kann verwendet werden, um einer Quelle Priorität über eine andere zu geben
repository URL	URL des Debian Repository
distribution	Name der Distribution
component	Durch Leerzeichen getrennte Liste der Komponenten

Beispiel:

```
[source0]
name=common
architecture=amd64
source1=501 deb http://internalrepo.sampletec-01.com/common rp6-mirror main restricted
universe multiverse
source2=502 deb http://internalrepo.sampletec-01.com/common

[source1]
name=internaltool
architecture=amd64
base=common
source1=503 deb http://internalrepo.sampletec-01.com/tool rp6 main
```

6.3. Metadatei-Struktur

epkg ist in zwei Verzeichnisse aufgeteilt:

elux	enthält alle Metadateien mit der Beschreibung, wie die EPM/FPMs gepackt werden sollen
input	enthält alle Dateien, die für die eLux-Paketkonfiguration bearbeitet oder erstellt wurden

Die Metadateien selbst teilen sich in zwei Gruppen:

Global meta files steuern das Verhalten aller EPM/FPMs, während **Per FPM meta files** für jedes FPM erforderlich sind.

Global meta files

Datei	Beschreibung
elux/control	Wenn Sie Debian Repositories einsetzen, enthält diese Datei den Schlüssel für das verwendete Debian Repository.
elux/version	Nur zur internen Verwendung - nicht verändern! Definiert die Format-Version der Metadaten
elux/variables	Definiert Variablen, die in allen anderen Metadateien verwendet werden können Jede Zeile enthält eine Variablendefinition im Format `<name>=<value>` Variable können in anderen Dateien referenziert werden mit `%<name>%`
elux/stripExcludes	Standardmäßig werden alle strippbaren Dateien nach dem Kopieren in den Paketbaum entfernt. Jede Zeile enthält einen regulären PERL-Ausdruck mit Dateien, die beim Strippen ausgeschlossen werden sollen. Für weitere Informationen siehe http://www.boost.org/doc/libs/1_57_0/libs/regex/doc/html/boost_regex/syntax/perl_syntax.html
elux/preCommands	Enthält Befehle, die ausgeführt werden sollen, bevor die Dateien in die Paketbaumstruktur kopiert werden Die `%root%`-Variable kann um Referenzieren des `tmp/`-Verzeichnisses verwendet werden. Die Datei wird ausgeführt mit `sh -c`
elux/<epmName>_epm.md5	Nur zur internen Verwendung - nicht verändern! Enthält alle md5-Prüfsummen der vom EPM verwendeten Daten. Die Datei ist erforderlich, um zu überprüfen, ob das EPM geändert wurde.

Ab der **epkg**-Version 1.0.50 (enthalten ab eLux SDK VM 3.3) werden die Pre- und Post-(Un-)Install-Skripte als separate Dateien direkt an die jeweiligen Pakete gebunden:[1]

elux/<epmName>_epm.preInst	Skript, das vor der Paket-Installation ausgeführt wird
elux/<epmName>_epm.postInst	Skript, das nach der Paket-Installation ausgeführt wird
elux/<epmName>_epm.preUninst	Skript, das vor der Paket-Deinstallation ausgeführt wird
elux/<epmName>_epm.postUninst	Skript, das nach der Paket-Deinstallation ausgeführt wird

In alle Install-Skripte können Sie weitere Dateien einbeziehen,[2] um beispielsweise Code in anderen Install-Skripten wiederzuverwenden. Verwenden Sie dazu die Anweisung `#include`, gefolgt von dem relevanten Pfad relativ zum `elux`-Verzeichnis. Die `#include`-Anweisung wird durch den Inhalt der angegebenen Datei ersetzt.

Beispiel: `#include scripts/test`

Per FPM meta files

Datei	Beschreibung

elux/<fpmName>.debs	enthält alle Debian-Pakete, aus denen Dateien stammen, die das FPM beinhaltet
	
	Jede Zeile enthält den Namen eines Debian-Paketes. Während des Build-Vorgangs versucht das System, die im Debian-Paket enthaltene Lizenzinformation zu analysieren (parsen). Wenn die Analyse fehlschlägt, wird eine Meldung angezeigt und die Lizenz muss manuell eingegeben werden. Die Lizenz kann durch Hinzufügen einer neuen Zeile direkt unter dem Namen des Debian-Paketes in folgendem Format angegeben werden: `Licence: <Licence1>,<Licence2>`
	
	Beispiel:
	`Xauth`
	`Licence: MIT`
	`xfonts-utils`
	`Licence: BSD-2-Clause,MIT`

[1] in vorherigen Versionen nur über die GUI (ebkworkspace) verfügbar
[2] ab eLux SDK VM 3.7

Datei	Beschreibung
elux/<fpmName>.dirs	Jede Zeile dieser Datei enthält ein Verzeichnis, das erstellt wurde. Dies wird notwendig, wenn ein FPM leere Verzeichnisse enthält.
elux/<fpmName>.excludes	Jede Zeile enthält einen regulären PERL-Ausdruck mit Dateien, die für das Paketieren ausgeschlossen werden sollen. Dies kann hilfreich sein, wenn Sie Wildcard-Zeichen in `elux/<fpmName>.install` verwenden.
elux/<fpmName>.install	enthält alle Dateien, die das FPM beinhaltet Jede Zeile enthält eine Zeichenkette in einem der folgenden Formate: `<sourcePath>` (wenn Quelle und Ziel identisch sind) `<sourcePath> -> <destinationPath>` Der `<sourcePath>`darf Wildcard-Zeichen (`*`) enthalten. Der `<sourcePath>` ist relativ zum `tmp/`-Verzeichnis, und der `<destinationPath>` ist relativ zum Paketbaum-Wurzelverzeichnis.
elux/<fpmName>.md5	Nur zur internen Verwendung - nicht verändern! Enthält alle md5-Prüfsummen der vom FPM verwendeten Daten. Die Datei ist erforderlich, um zu überprüfen, ob das FPM geändert wurde.
elux/<fpmName>.postCommands	Contains commands to be executed after the files are copied to the package tree structure. The `%root%` variable can be used to reference the package tree root. Die Datei wird ausgeführt mit `sh -c`
elux/<fpmName>.size	Enthält die unkomprimierte und die komprimierte Größe des FPM Die unkomprimierte Größe wird während des Build-Vorgangs automatisch berechnet oder kann mit dem Befehl `epkg Size --calculate` berechnet werden. Die komprimierte Größe muss aus der `eluxmanlog`-Datei bezogen werden. Dazu verwenden Sie das Kommando: `epkg Size --fromEluxManLog <pathToEluxManLog>`

Datei	Beschreibung

elux/<fpmName>.thirdparty	Enthält all Thirdparty-Pakete, aus denen Dateien stammen, die das FPM beinhaltet
	
	Jede (zweite) Zeile enthält den Pfad zu einem Thirdparty-Paket, gefolgt von einer Zeile, die die Lizenzinformation im folgenden Format enthält:
	
	`Licence: <Licence1>,<Licence2>`
	
	Zusätzlich können Sie Optionen definieren, indem Sie eine weitere Zeile hinzufügen, die mit der Zeichenkette `Options` beginnt:
	
	`Options` ist eine Komma-separierte Liste aus folgenden Optionen:
	
	`NoExtract`: Die Dateien sollen nicht ausgepackt, sondern nur ins `tmp`-Verzeichnis kopiert werden.
	
	Unterstützte Dateierweiterungen:
	
	`.zip`
	`.tar.*`
	`.tar`
	`.tgz`
	`.rpm`
	`.deb`
	`.xpi`
	`.bz2`
	`.bin` (erforderlich fürJava 1.6, funktioniert nur für Java 1.6)
	
	Beispiel:
	
	`/home/mirror/import/mozilla/ESR//-38.5.2esr/firefox.tar.bz2`
	`Licence: MPL-2.0`

Ab der **epkg**-Version 1.0.50 (enthalten ab eLux SDK VM 3.3) werden die Pre- und Post-(Un-)Install-Skripte als separate Dateien direkt an die jeweiligen Pakete gebunden:[1]

elux/<fpmName>.preInst	Skript, das vor der Paket-Installation ausgeführt wird
elux/<fpmName>.postInst	Skript, das nach der Paket-Installation ausgeführt wird
elux/<fpmName>.preUninst	Skript, das vor der Paket-Deinstallation ausgeführt wird
elux/<fpmName>.postUninst	Skript, das nach der Paket-Deinstallation ausgeführt wird

[1] in vorherigen Versionen nur über die GUI (ebkworkspace) verfügbar

6.4. Anwendungsbeispiele

Paket-Kategorie setzen

Mit ELIAS 18 können eLux-Pakete nach Kategorie sortiert angezeigt werden. Ordnen Sie benutzerdefinierte Pakete einer Kategorie zu.

1. Bearbeiten Sie die Datei `/input/epkepm`:
   `edit ./input/epkepm`
2. Setzen Sie für Ihre Pakete im Abschnitt `epm` den Parameter `category`. Beispiel:
   ```
   [EPM]
   category=Communication
   ```

Folgende Kategorien können gesetzt werden:

Kategorie	Beschreibung	Beispiel
System	Betriebssystem-Komponenten	eLux OS, Linux-Kernel, Deskop environment
Anwendung	Backend- und lokale Anwendungen, Web-Browser	Citrix Workspace app, VMware Horizon-Client, Firefox
Sicherheit	VPN, Smartcard-Middleware	Cisco AnyConnect, cryptovision sc/interface
Kommunikation	Unified Communication, VoIP	Cisco JVDI, Citrix HDX RTME, Zoom Citrix-Plugin, Zoom-Client für Linux, Ekiga SoftPhone
Netzwerk	Proxy, Network Access Control	Squid Proxy-Server, IEEE802.1X
Multimedia	Audio, Video	GStreamer, Fluendo Codec Pack
Treiber	Unterstützung für neue Hardware	WLAN, WWAN, Drucker
Werkzeuge	Bibliotheken	BIOS-Update, XInput Calibrator
Vermischtes	Benutzerdefinierte Pakete, Pakete ohne Kategorie	

Wenn Sie für Ihre Pakete keine Kategorie setzen, wird das Paket unter **Miscellaneous** angezeigt.

Paketbaum-Struktur konvertieren in Metadatei-Struktur

- Um einen Paketbaum in das neue Meta-Format zu konvertieren, verwenden Sie folgenden Befehle:

  ```
  epkg NewFrom --pkgTree=<path to package tree>
  ```

  Alle erforderlichen Dateien werden unter `elux/` erstellt. Die Dateien sind jedoch leer.

- Füllen Sie `*.thirdparty` und `*.debs` mit den Input-Daten.

```
epkg ConvertFrom --pkgTree=<path to package tree>
                 --container =rp6_x64
```

Die Metadateien `*.install` und `*.dirs` werden gefüllt und die erforderlichen Verzeichnisse werden unter `input/` erstellt.

Paket im Test-Modus erstellen

```
epkg Build --releaseType=test --container=rp6_x64
```

Hinweis
Der Test-Modus ist standardmäßig aktiv.

Paket im Release-Modus erstellen

```
epkg Build --releaseType=release --container=rp6_x64
```

Paket ohne Signierung und ohne Zippen erstellen

```
epkg Build --skipSign --skipCreateZip --container=rp6_x64
```

Paket installieren

```
epkg Install
```

Alle EPM/FPMs, die Signaturdateien und die `.zip`-Dateien werden in das konfigurierte Container-Verzeichnis kopiert.

7. Sonstiges

7.1. Citrix Virtual Channels

Mit Ihren individuellen eLux-Paketen können Sie virtuelle Kanäle für die Citrix Workspace-App bereitstellen (Citrix Virtual Channel).

Die notwendigen Informationen zum Aktivieren eines virtuellen Kanals werden dem relevanten FPM in Form einer `.ini`-Datei mitgegeben. Die `.ini`-Datei muss im Verzeichnis `/usr/share/citrix/vd.d/` abgelegt sein. Ein Skript erzeugt aus den hier abgelegten `.ini`-Dateien und der standardmäßig verwendeten `module.ini.orig` die Ziel-Datei `module.ini`

Die Struktur einer `.ini`-Datei kann folgendermaßen aussehen:

```
[WFClient]
DesktopApplianceMode=True

[ICA 3.0]
GenericUSB=On
VirtualDriver=GenericUSB

[ClientDrive]
MaxRequestSize2=4116

[GenericUSB]
DriverName=VDGUSB.DLL
```

Hinweis
Wenn ein Wert in der Datei `module.ini.orig` bereits existiert, wird er durch den neuen Wert überschrieben. Ausnahme: `[ICA 3.0] VirtualDriver`, hier wird der neue Wert an die Liste der vorhandenen Treiber angehängt.

7.2. Skripte nach Konfig-Aktualisierung

Skripte werden normalerweise nach dem Gerätestart ausgeführt. In manchen Situationen ist es hilfreich, Anweisungen nach dem Abgleich der lokalen Konfiguration mit der Systemkonfiguration ausführen zu lassen. Dafür hinterlegen Sie eine Skript-Datei in folgendem Verzeichnis:

`/etc/elux/eluxd.d/`

Das Skript wird dann nach jeder Konfigurations-Aktualisierung mit dem Parameter `config` aufgerufen.