エッジゲートウェイ
デベロッパーズマニュアル

AG10

AG20

IM AMD03A01-51JA　第 6 版

ライセンスおよび商標

ライセンス

- 本製品では、GPL（GNU General Public License）等のオープンソースライセンスに基づくソフトウェアを使用しています。
  詳細については、当社 Web サイトに記載しております。
  URL：　https://support.amnimo.com/

- 保証の範囲と対応責任
  本製品において、GPL 等の条項に従いオープンソースソフトウェアそのものの動作に関しては、保証を行いません。

商標

- 本文中に使われている商品名、会社名などの固有名詞は各社の商標または登録商標です。

はじめに

このたびは当社のエッジゲートウェイ amnimo G series（以下「エッジゲートウェイ」）をご採用いただき、誠にありがとうございます。

amnimo デベロッパーズマニュアル（以下「本書」）は、エッジゲートウェイ上で動作するアプリケーションを作成する「開発者」に向けた情報を記載しています。また、「ユーザーズマニュアル（IM AMD03A01-01JA）」の内容を理解されていることを前提としています。あわせてご参照ください。

本書について

本書に関するご注意

- 本書の内容は、将来予告なしに変更することがあります。
- 本書の内容の全体または一部を無断で転載、複製することは禁止されています。
- 本書の内容に関しては万全を期していますが、万一ご不審の点や誤りなどお気づきのことがありましたら、当社カスタマーサポートまでご連絡ください。

  連絡先： アムニモカスタマーサポート
  E-mail： support@amnimo.com
  URL： https://support.amnimo.com

- 機能／性能上とくに支障がないと思われる仕様変更、構造変更、および使用部品の変更については、その都度の改訂はしない場合がありますのでご了承ください。

マニュアル一覧

- エッジゲートウェイデベロッパーズマニュアル（本書）
  https://amnimo.com/manual/edge_gw/doc/IM_AMD03A01-51JA_Edge_Gateway_Indoor_amnimo_G_series_Developers_manual.pdf
- エッジゲートウェイシリーズ CLI ユーザーズマニュアル
  https://amnimo.com/manual/edge_gw/cli/ja/cli_users_manual.pdf
- エッジゲートウェイシリーズ GUI ユーザーズマニュアル
  https://amnimo.com/manual/edge_gw/gui/ja/gui_users_manual.pdf
- エッジゲートウェイユーザーズマニュアル
  https://amnimo.com/manual/edge_gw/ja/edge_gw_users_manual.pdf
- エッジゲートウェイスタートアップガイド
  https://amnimo.com/manual/edge_gw/sg/ja/edge_gw_sg.pdf
- デバイス管理システムマニュアル
  https://amnimo.com/manual/edge_gw/alpine/dms/ja/index.htm
- オープンソースソフトウェア使用許諾条件書
  https://amnimo.com/manual/edge_gw/doc/IM_AMD03A01-12JA_amnimo_GW_series_OSS_license.pdf

本書で使用しているアイコンと記号

本書のアイコンと記号には、以下の意味があります。

	機能や操作に関して、特に注意する情報を記載しています。
	機能や操作に関して、補足的な情報を記載しています。
	本書内や他の文書への参照情報を記載しています。
	一般ユーザーモードでコマンド操作できることを示しています。
	管理者モードでコマンド操作できることを示しています。
	設定モードでコマンド操作できることを示しています。

スタイルガイド

本書のコマンド書式は、以下のように記述しています。

表記	説明
VALUE	● 太字の場合は、固定値です。 ● 太字斜体の場合は、設定パラメーターまたはキーワードです。省略不可です。
[A | B]	A または B を選択します。省略可能です。
< A | B >	A または B を選択します。省略不可です。
[0 – 9]	0 から 9 までのいずれかの値を選択します。省略可能です。
< 0 – 9 >	0 から 9 までのいずれかの値を選択します。省略不可です。
↵	改行（Enter キー入力）を表します。

コマンドの実行例

本書のコマンドの実行例は、実行する環境に応じて、以下のように色分けして記述しています。

エッジゲートウェイ

エッジゲートウェイ（ネイティブ開発環境）のコンソール操作を示しています。

```
gateway
```

ホストマシン

ホストマシン（クロス開発環境）のコンソール操作を示しています。

```
host
```

その他

共通のコマンド、書式、ソースコードなどを示しています。

```
```

安全および改造に関するご注意

- 人体および本エッジゲートウェイまたは本エッジゲートウェイを含むシステムの保護・安全のため、本エッジゲートウェイを取り扱う際は、本書の安全に関する指示事項に従ってください。なお、これらの指示事項に反する扱いをされた場合、当社は安全性を保証いたしかねます。
- 本エッジゲートウェイを無断で改造することは固くお断りします。
- その他、無線通信を含む、安全に関する注意事項の詳細については、『エッジゲートウェイユーザーズマニュアル』に記載しています。あわせてご参照ください。

目次

第1章　エッジゲートウェイの開発環境

本章では、エッジゲートウェイの開発環境について説明します。

1.1　概要

エッジゲートウェイには ARMv8 64bit の CPU が搭載されています。エッジゲートウェイ上で動作するアプリケーションを開発するためには、エッジゲートウェイ上でプログラムのビルドと実行を行うためのネイティブ開発環境を構築する必要があります。ホスト上で本 CPU アーキテクチャに沿ってプログラムをビルドする場合は、クロス開発環境を構築する必要があります。

本書では、C 言語をベースとしたエッジゲートウェイのアプリケーション開発をクロスおよびネイティブ開発環境で実現する方法について説明しています。

1.1.1　エッジゲートウェイのハードウェア仕様

カテゴリー		屋内タイプ	屋外タイプ
CPU		ARM Cortex-A53 1000MHz（2 コア）	
RAM		2Gbyte	
NOR-FLASH		4MByte	
NAND-FLASH		512MByte	
eMMC		32Gbyte	
SSD		64GByte～2TByte※1	64GByte～4TByte※1
モバイル回線※2		3G／4G	
アンテナ端子		3G／4G 用 SMA × 2, GPS 用 SMA × 1	（アンテナ内蔵）
GNSS		GPS（QZSS）／GLONASS／Galileo／BeiDou	
PoE 給電		IEEE 802.3at	
インターフェイス			
	Ethernet	Gigabit Ethernet 5 ポート （内 4 ポート PoE 給電機能付きスイッチ）	
	コンソール	RJ45	USB Type-C
	RS-232	DB9	-
	RS-485	-	専用端子台
	SD カード	1 スロット（SDXC）	
	SIM スロット	Micro SIM（3FF）×2 スロット、eSIM × 2※3	
	USB	USB2.0 ホスト 1 ポート、Type-A	
	DI DO	デジタル IN フォトカプラー 4ch デジタル OUT フォトモスリレー 2ch 端子台（2 ピース型スクリュータイプ）	
スイッチ		● Push スイッチ × 1（シャットダウンおよび工場出荷時設定用） ● DIP スイッチ × 4（設定用）	● Push スイッチ × 1（シャットダウンおよび工場出荷時設定用） ● DIP スイッチ × 4（設定用） ● Rocker スイッチ× 1（電源用）
LED		2 色 LED（PWR、ANT、MOB、ANT、ST1、ST2、ST3）	
サイズ		177（W）×110（D）×44（H）mm （ただし突起含まず）	200（W）×151.5（D）×300（H）mm （ただし突起含まず）
ケース		-	防水・防塵開閉式 PC プラボックス
防塵防水		-	IP65

カテゴリー	屋内タイプ	屋外タイプ
重量	約 820g	約 2.8kg
電源仕様	電源電圧：10.8VDC（12VDC-10%） ～32VDC（24VDC+20%） 消費電力：最大 50W（内 PoE 40W） 絶対最大定格 60W	定格入力電圧：100VAC-240VAC 周波数：50/60Hz 定格入力電力・用量：70W/ 70-75VA
動作温度	-20°C～60°C	
保存温度	-20°C～70°C	
相対湿度	10%～90% (結露なきこと)	10%～90%

※1 オプションとして提供しています。

※2 搭載する通信モジュールによって帯域が変わります。

※3 eSIM の使用を検討されているお客様はご相談ください。

1.1.2 エッジゲートウェイのソフトウェア仕様

カテゴリー		説明
カーネル		Linux（4.19）
OS		Ubuntu 18.04 LTS
基本プロトコル		ARP／IPv4／IPv6※／UDP／TCP
接続プロトコル		IPCP／PPP／PPPoE （IPv4／IPv6※）
動的 IP アドレス		DHCP サーバー／クライアント（IPv4／IPv6※） DHCP サーバーはインターフェイスごとに設定することが可能。
ドメイン名解決		DNS リレー／クライアント
ルーティング		静的ルーティング
アドレス変換・ポート変換		NAT・NAPT（宛先／送信先）
VPN		IPsec／remote.it による VPN
時刻同期		NTP／GPS
セキュリティ		パケットフィルター
運用管理		
	設定手段	amsh amnimo ゲートウェイ専用 CLI（Command Line Interface）
	ファームウェア更新	apt（差分更新）／amfirm（全体更新）
	ログ管理	syslog
	デバイスマネジメント	amnimo DMS
開発ツールチェイン		● gcc-7 7.5.0-3ubuntu1～18.04 ● binutils 2.30-21ubuntu1～18.04.2

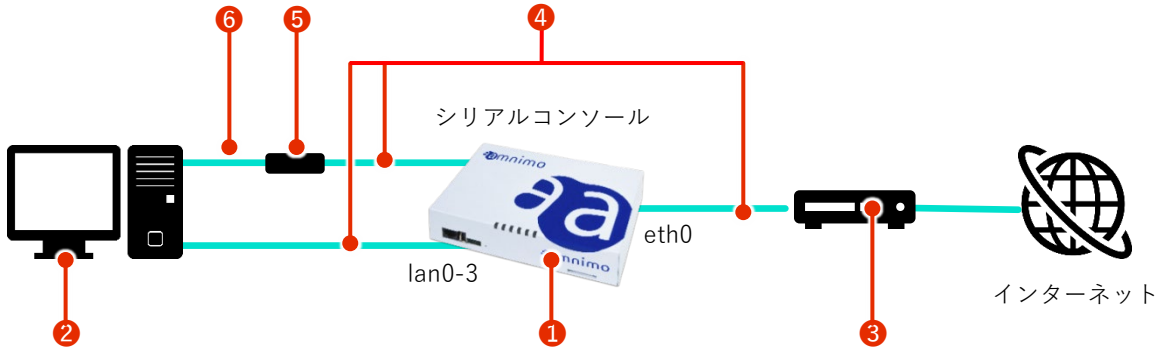
※ IPv6 は、Ver.2 以降で対応する予定です。

1.2　事前準備

エッジゲートウェイの開発環境を構築するにあたって、事前に準備するべき内容について説明します。

1.2.1　ハードウェア構成

エッジゲートウェイの開発環境を構築するためのハードウェア構成（屋内タイプのエッジゲートウェイ）を以下に示します。



必要なハードウェアに関する説明

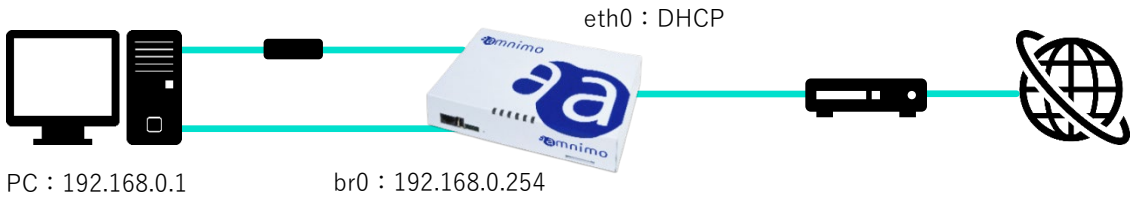
No.	ハードウェア	説明
❶	屋内タイプエッジゲートウェイ	開発したアプリケーションが動作する環境です。また、ネイティブ開発環境の場合は、ビルドする環境になります。『1.2.2』説明する設定が必要となります。
❷	PC	クロス開発環境の場合は、アプリケーションをビルドする環境になります。 初期状態では、SSH が無効になっているため、設定変更するために、RS232C インターフェイスが必要となります。USB ポートが存在する場合は、USB-RS232C 変換アダプター等で代用できます。
❸	有線ルーター	ゲートウェイをインターネットへ接続するために用意します。 DHCP サーバー機能を設定します。
❹	LAN ケーブル	シリアルコンソール eth0／lan0-3 に接続するために使用します。
❺	RJ45-RS232C 変換コネクター	RJ45（LAN ケーブル）から RS232C（RS232C ケーブル）に変換するために使用します。 屋外タイプのエッジゲートウェイでは不要です。
❻	RS232C ケーブル	PC の RS232C コネクターと RJ45-RS232C 変換コネクターを接続します。 屋外タイプのエッジゲートウェイでは USB Type-C のケーブルを使用します。

- エッジゲートウェイ以外は開発するユーザー側で用意する必要があります。ハードウェアの詳細については、『エッジゲートウェイユーザーズマニュアル』を参照してください。
- 本書内の説明は屋内タイプのエッジゲートウェイで説明していますが、屋外タイプのエッジゲートウェイもシリアルコンソールに接続するハードウェアが異なる以外基本構成は同じです。

社内 LAN には接続せず、必ず独立したネットワークとして構成するようにしてください。

1.2.2　エッジゲートウェイの設定

『1.2.1』で説明したハードウェア構成を開発環境として動作可能にするための設定例について説明します。以下に、本章で使用しているエッジゲートウェイの設定例を示します。



エッジゲートウェイの設定例

設定項目	設定内容
インターフェイス	eth0：　dhcp※ br0：　192.168.0.254※
SSH	有効※
ホスト名	amnimo
タイムゾーン	Asia/Tokyo

※　工場出荷時の初期設定です。

amsh の設定モードで設定一覧を表示して、設定内容を確認します。

設定モード

gateway

```
amnimo(cfg)# show config ↵
# ---- transition to configure mode ----
configure
# ---- hostname configure ----
hostname amnimo
# ---- timezone configure ----
timezone Asia Tokyo
# ---- interface eth0 configure ----
interface eth0
enable
pmtu auto
dhcp4
dhcp4 dns 30
dhcp4 ntp
dhcp4 mtu
dhcp4 route 30
mtu 1500
mode 100baseT-Auto
proxy-arp
no optional
exit
# ---- interface lan0 configure ----
interface lan0
enable
pmtu auto
mtu 1500
mode 100baseT-Auto
proxy-arp
no optional
```

```
exit
# ---- interface lan1 configure ----
interface lan1
enable
pmtu auto
mtu 1500
mode 100baseT-Auto
proxy-arp
no optional
exit
# ---- interface lan2 configure ----
interface lan2
enable
pmtu auto
mtu 1500
mode 100baseT-Auto
proxy-arp
no optional
exit
# ---- interface lan3 configure ----
interface lan3
enable
pmtu auto
mtu 1500
mode 100baseT-Auto
proxy-arp
no optional
exit
# ---- interface br0 configure ----
interface br0
enable
bridge lan0
bridge lan1
bridge lan2
bridge lan3
mac lan0
pmtu auto
address 192.168.0.254/24
mtu 1500
proxy-arp
no optional
exit
# ---- filter input configure ----
filter input default-policy accept
# ---- filter output configure ----
filter output default-policy accept
# ---- filter forward configure ----
filter forward default-policy accept
# ---- rule 100 ----
filter forward 100
enable
policy drop
match protocol udp dst-port 137:138
exit
# ---- rule 110 ----
filter forward 110
enable
policy drop
match protocol udp src-port 137:138
exit
```

```
# ---- rule 120 ----
filter forward 120
enable
policy drop
match protocol tcp dst-port 137
exit
# ---- rule 130 ----
filter forward 130
enable
policy drop
match protocol tcp src-port 137
exit
# ---- rule 140 ----
filter forward 140
enable
policy drop
match protocol tcp dst-port 139
exit
# ---- rule 150 ----
filter forward 150
enable
policy drop
match protocol tcp src-port 139
exit
# ---- rule 160 ----
filter forward 160
enable
policy drop
match protocol tcp dst-port 445
exit
# ---- rule 170 ----
filter forward 170
enable
policy drop
match protocol tcp src-port 445
exit
# ---- dns configure ----
dns
no enable
exit
# ---- ipsec log-level configure ----
ipsec log-level
asn control
cfg control
chd control
dmn control
enc control
esp control
ike control
imc control
imv control
job control
knl control
lib control
mgr control
net control
pts control
tls control
tnc control
exit
```

```
# ---- ntp configure ----
ntp
no enable
exit
# ---- ssh configure ----
ssh
no enable
exit
# ---- syslog local configure ----
syslog local
enable
rotate-size 10240
rotate-count 8
level informational
exit
# ---- syslog remote configure ----
syslog remote
no enable
server-port 514
level informational
exit
# ---- storage sda1 configure ----
storage mount sda1 /media/ssd type ext4 options defaults
storage fsck sda1 preen
storage monitor sda1 retry 3 interval 10m reboot 3
storage failsafe sda1 retry 3 interval 10 reboot 3
# ---- poe lan0 configure ----
poe lan0
enable
limit-current auto
ondelay 0
exit
# ---- poe lan1 configure ----
poe lan1
enable
limit-current auto
ondelay 0
exit
# ---- poe lan2 configure ----
poe lan2
enable
limit-current auto
ondelay 0
exit
# ---- poe lan3 configure ----
poe lan3
enable
limit-current auto
ondelay 0
exit
# ---- cpufreq configure ----
cpufreq ondemand
# ---- thermal polling configure ----
thermal polling 1000
# ---- thermal cpufreq high configure ----
thermal cpufreq high
enable
mode high
temperature 100.0
hysteresis 10.0
```

```
log detection warnings
log restoration notifications
state 200MHZ
exit
# ---- thermal cpufreq low configure ----
thermal cpufreq low
enable
mode low
temperature -10.0
hysteresis 5.0
log detection warnings
log restoration notifications
state 1000MHZ
exit
# ---- thermal mobile high configure ----
thermal mobile high
enable
mode high
temperature 100.0
hysteresis 10.0
log detection warnings
log restoration notifications
state disable
exit
# ---- thermal interface high configure ----
thermal interface high
enable
mode high
temperature 100.0
hysteresis 10.0
log detection warnings
log restoration notifications
state 100baseT-Auto
exit
# ---- dms configure ----
dms
no enable
exit
# ---- nxwitness configure ----
nxwitness
enable
port 7001
database /mnt/share/nxwitness/database/file.db
password secret 1sxWjNj/NBbdEfGFmP6vrw==
exit
# ---- remoteit configure ----
remoteit
no enable
exit
# ---- gui configure ----
gui
enable
protocol http
port 80
exit
```

1.2.3 PC 環境の準備

PC の OS には、エッジゲートウェイと同じ Ubuntu 18.04 LTS 版をインストールします。

ダウンロード先 URL： https://releases.ubuntu.com/18.04/

Windows 環境で開発する場合は、仮想化技術ソフトウェア Virtual Box や VMware などの仮想環境を構築し、その上で開発することをおすすめします（本マニュアルでの説明は割愛します）。

Ubuntu のバージョン確認方法

host

```
$ lsb_release -a ↵
No LSB modules are available.
Distributor ID:  Ubuntu
Description:     Ubuntu 18.04.4 LTS
Release: 18.04
Codename:        bionic
```

Ubuntu のバージョンは、ユーザー環境のアップデートに伴って更新される可能性があります。

IP アドレスの設定／確認

以下の例では、PC のインターフェイス（ens36）がエッジゲートウェイと同じネットワーク（192.168.0.0/24）に接続しています。

host

```
$ sudo ip addr add 192.168.0.1 dev ens36 ↵
$ ip addr show dev ens36 ↵
3: ens36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defa
ult qlen 1000
    link/ether 00:0c:29:6d:95:a9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.1/24 brd 192.168.0.255 scope global ens36
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe6d:95a9/64 scope link
       valid_lft forever preferred_lft forever
```

1.3　クロス開発環境の作成

エッジゲートウェイのアプリケーションをクロス開発環境で開発する方法について説明します。

1.3.1　PC 環境の準備

apt パッケージのインストール

PC 側の Ubuntu 環境で、apt パッケージリストの更新とアップグレードをして、以下の apt パッケージをインストールしてください。

host

```
$ sudo apt update ↵
$ sudo apt upgrade ↵
$ sudo apt install build-essential crossbuild-essential-arm64 device-tree-compiler libs
sl-dev gcc-arm-linux-gnueabi file tree ↵
```

ツールチェインのバージョン確認

ツールチェインのバージョンを確認してください。

host

```
$ aarch64-linux-gnu-gcc --version ↵
aarch64-linux-gnu-gcc (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

ツールチェインコマンド一覧

apt パッケージ	コマンド
binutils-aarch64-linux-gnu	aarch64-linux-gnu-addr2line
	aarch64-linux-gnu-ar
	aarch64-linux-gnu-as
	aarch64-linux-gnu-c++filt
	aarch64-linux-gnu-dwp
	aarch64-linux-gnu-elfedit
	aarch64-linux-gnu-gprof
	aarch64-linux-gnu-ld
	aarch64-linux-gnu-ld.bfd
	aarch64-linux-gnu-ld.gold
	aarch64-linux-gnu-nm
	aarch64-linux-gnu-objcopy
	aarch64-linux-gnu-objdump
	aarch64-linux-gnu-ranlib
	aarch64-linux-gnu-readelf
	aarch64-linux-gnu-size
	aarch64-linux-gnu-strings
	aarch64-linux-gnu-strip
cpp-aarch64-linux-gnu	aarch64-linux-gnu-cpp
g++-aarch64-linux-gnu	aarch64-linux-gnu-g++
g++-aarch64-linux-gnu	aarch64-linux-gnu-gcc
	aarch64-linux-gnu-gcc-ar
	aarch64-linux-gnu-gcc-nm
	aarch64-linux-gnu-gcc-ranlib

apt パッケージ	コマンド
	aarch64-linux-gnu-gcov
	aarch64-linux-gnu-gcov-dump
	aarch64-linux-gnu-gcov-tool

1.3.2 クロスビルドの実行

サンプルコードを用いてクロスビルドを実行する手順について説明します。

サンプルコード（main.c、Makefile）の作成

以下に示すのは、10 秒ごとに文字列を出力し続けるプログラムのソースコードです。

main.c

host

```
#include <stdio.h>
#include <unistd.h>

int main(void)
{
    while(1){
        printf("Hello World!¥n");
        sleep(10);
    }
}
```

Makefile

```
CC = aarch64-linux-gnu-gcc
CFLAGS = -O3 -Wall
DESTDIR = /usr/bin
LIBS =
OBJS = main.o
PROG = amnimo-sample-app

all: $(PROG)

$(PROG): $(OBJS)
        $(CC) $(OBJS) $(LIBS) -o $(PROG)

clean:
        rm -f *.o $(PROG)

install: $(PROG)
        sudo install -s $(PROG) $(DESTDIR)
```

ディレクトリ構造とファイル配置を以下に示します。

```
~/sample$ tree ↵
.
├── main.c
└── Makefile
```

ビルド

host

```
~/sample$ make all ↵
```

ビルド結果の実行ファイルの種別の確認

host

```
~/sample$ file amnimo-sample-app ↵
amnimo-sample-app: ELF 64-bit LSB shared object, ARM aarch64, version 1 (SYSV), dynamic
ally linked, interpreter /lib/ld-linux-aarch64.so.1, for GNU/Linux 3.7.0, BuildID[sha1]
=1e9a9503efe79366a8c4f51bba5d20cc65f9a7fe, not stripped
```

実行ファイル転送

生成された実行ファイル amnimo-sample-app を、scp でエッジゲートウェイに送信します。

host

```
~/sample$ scp amnimo-sample-app admin@192.168.0.254:/tmp/ ↵
The authenticity of host '192.168.0.254 (192.168.0.254)' can't be established.
ECDSA key fingerprint is SHA256:AJ0j48/CzTC8mETZnRjwHyGegbpq00vQOg6/8sB9npg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.254' (ECDSA) to the list of known hosts.
admin@192.168.0.254's password:
amnimo-sample-app                                     100% 9080     3.0MB/s   00:00
```

青字部分（fingerprint の確認）は、以下の場合に出力されます。

- 初回の ssh ログイン時
- scp によるデータ送受信時

実行ファイルの実行

エッジゲートウェイにログインし、実行ファイルを実行します。

gateway

```
admin@amnimo:~$ /tmp/amnimo-sample-app ↵
Hello World!
Hello World!
…
```

1.4　ネイティブ開発環境

エッジゲートウェイのアプリケーションをネイティブ開発環境で開発する方法について説明します。

1.4.1　エッジゲートウェイの準備

インストール済みツールチェイン

工場出荷時のエッジゲートウェイには、以下のツールチェインが実装されています。

apt パッケージ	コマンド
binutils-aarch64-linux-gnu	aarch64-linux-gnu-addr2line
	aarch64-linux-gnu-ar
	aarch64-linux-gnu-as
	aarch64-linux-gnu-c++filt
	aarch64-linux-gnu-dwp
	aarch64-linux-gnu-elfedit
	aarch64-linux-gnu-gprof
	aarch64-linux-gnu-ld
	aarch64-linux-gnu-ld.bfd
	aarch64-linux-gnu-ld.gold
	aarch64-linux-gnu-nm
	aarch64-linux-gnu-objcopy
	aarch64-linux-gnu-objdump
	aarch64-linux-gnu-ranlib
	aarch64-linux-gnu-readelf
	aarch64-linux-gnu-size
	aarch64-linux-gnu-strings
	aarch64-linux-gnu-strip

ビルドに必要な apt パッケージのインストール

エッジゲートウェイにログインし、エッジゲートウェイ側の Ubuntu 環境において、apt パッケージリストの更新とアップグレードをして、以下の apt パッケージをインストールしてください。

なお、事前に、『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『2.5　パッケージリポジトリの操作』を確認してください。

gateway

```
$ sudo apt update ↵
$ sudo apt upgrade ↵
$ sudo apt install gcc file tree ↵
```

C++（g++-aarch64-linux-gnu）や cmake などを利用したい場合は、あらかじめ準備しておく必要があります。

ツールチェインのバージョン確認

ツールチェインのバージョンを確認します。

gateway

```
admin@amnimo:~$ gcc --version ↵
gcc (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOS
```

1.4.2　ネイティブビルドの実行

サンプルコードを用いてネイティブビルドを実行する手順について説明します。

サンプルコード（main.c、Makefile）の作成

『1.3.2』のサンプルコードを利用します。

main.c

gateway

```
#include <stdio.h>
#include <unistd.h>

int main(void)
{
    while(1){
        printf("Hello World!¥n");
        sleep(10);
    }
}
```

Makefile

```
CC = gcc
CFLAGS = -O3 -Wall
DESTDIR := /
LIBS =
OBJS = main.o
PROG = amnimo-sample-app

all: $(PROG)

$(PROG): $(OBJS)
        $(CC) $(OBJS) $(LIBS) -o $(PROG)

clean:
        rm -f *.o $(PROG)

install: $(PROG)
        install -D -s $(PROG) $(DESTDIR)/usr/sbin/$(PROG)
```

ディレクトリ構造とファイル配置を以下に示します。

```
admin@amnimo:~/sample$ tree ↵
.
├── main.c
└── Makefile
```

ネイティブビルドの実行

gateway

```
admin@amnimo:~$ make all ↵
```

ビルド結果の実行ファイルの種別の確認

gateway

```
$ file amnimo-sample-app ↵
amnimo-sample-app: ELF 64-bit LSB shared object, ARM aarch64, version 1 (SYSV), dynamic
ally linked, interpreter /lib/ld-linux-aarch64.so.1, for GNU/Linux 3.7.0, BuildID[sha1]
=1436db4cc909f72d8565e5f8d14a2b4be47c0515, not stripped
```

実行ファイルの実行

エッジゲートウェイにログインし、実行ファイルを実行します。

gateway

```
admin@amnimo:~$ /tmp/amnimo-sample-app ↵
Hello World!
Hello World!
…
```

1.5 分散ビルドの環境構築

ネイティブ開発環境では、汎用 PC と比較すると、エッジゲートウェイの CPU 処理能力のパフォーマンスが劣ります。そのため、クロス開発環境のビルド時間と比較すると、多くの時間が費やされます。プログラムの規模が大きくなると、この差は顕著になります。

以下では、distcc を使った分散ビルド環境を構築することで、ネイティブ開発環境のビルド処理を高速化する方法について説明します。

1.5.1 PC 環境の準備

ビルド処理の分散先として、ホスト側を利用します。

distcc/ccache パッケージのインストール

PC 側の Ubuntu 環境で、apt パッケージリストの更新とアップグレードをして、以下の apt パッケージをインストールしてください。

host

```
$ sudo apt install distcc ccache ↵
```

distcc の設定ファイルの修正

/etc/default/distcc の以下の箇所を編集します。

host

```
# Defaults for distcc initscript
# sourced by /etc/init.d/distcc

#
# should distcc be started on boot?
#
STARTDISTCC="true" # STARTDISTCC="false"を"true"にする

#
# Which networks/hosts should be allowed to connect to the daemon?
# You can list multiple hosts/networks separated by spaces.
# Networks have to be in CIDR notation, f.e. 192.168.0.0/24
# Hosts are represented by a single IP Adress
#
ALLOWEDNETS="192.168.0.0/24" # ALLOWEDNETS=に依頼元（エッジゲートウェイ）の IP アドレスを含むサブネットを指定する

#
# Which interface should distccd listen on?
# You can specify a single interface, identified by it's IP address, here.
#
LISTENER="192.168.0.1" # LISTENER=に PC の IP アドレス（エッジゲートウェイからアクセス可能なインターフェイスの IP アドレス）を指定する

#
# You can specify a (positive) nice level for the distcc process here
#
NICE="10"   # NICE（優先度高-20～19 優先度低）を指定する。ご使用の開発環境に合わせて設定してください。

#
# You can specify a maximum number of jobs, the server will accept concurrently
#
JOBS="8" # 分散ビルドする JOB 数を指定。ご使用の開発環境に合わせて設定してください。

#
```

```
# Enable Zeroconf support?
# If enabled, distccd will register via mDNS/DNS-SD.
# It can then automatically be found by zeroconf enabled distcc clients
# without the need of a manually configured host list.
#
# ZEROCONF="true"

ZEROCONF="false"
```

distcc デーモンの再起動

host

```
$ sudo systemctl daemon-reload ↵
$ sudo systemctl enable distcc ↵
$ sudo systemctl restart distcc ↵
$ netstat -a | grep distcc
tcp        0      0 localhost:distcc        0.0.0.0:*               LISTEN      ←出力される
ことを確認
```

1.5.2 エッジゲートウェイの準備

ホスト側と同様に、distcc/ccache のインストール、設定ファイルの編集、デーモンの再起動をします。

ただし、設定ファイルは、STARTDISTCC のみを変更します。

distcc/ccache パッケージのインストール

gateway

```
admin@amnimo:~$ sudo apt install distcc ccache ↵
```

distcc の設定ファイルの修正

/etc/default/distcc の以下の箇所を編集します。

gateway

```
# Defaults for distcc initscript
# sourced by /etc/init.d/distcc

#
# should distcc be started on boot?
#
STARTDISTCC="true" # STARTDISTCC="false"を"true"にする

# 以降はデフォルト

…
```

distcc デーモンの再起動

gateway

```
admin@amnimo:~$ sudo systemctl daemon-reload ↵
admin@amnimo:~$ sudo systemctl enable distcc ↵
admin@amnimo:~$ sudo systemctl restart distcc ↵
admin@amnimo:~$ netstat -a | grep distcc ↵
    tcp        0      0 localhost:distcc        0.0.0.0:*               LISTEN   ←出力される
ことを確認
```

環境変数の設定

分散ビルドの環境変数を設定します。

gateway

```
export CCACHE_PREFIX=distcc
export CC="ccache /usr/bin/aarch64-linux-gnu-gcc"
export CXX="ccache /usr/bin/aarch64-linux-gnu-g++"
export AS="/usr/bin/aarch64-linux-gnu-as"
export AR="/usr/bin/aarch64-linux-gnu-ar"
export CPP="/usr/bin/aarch64-linux-gnu-cpp"
export LD="/usr/bin/aarch64-linux-gnu-ld"
export PKG_CONFIG="/usr/bin/aarch64-linux-gnu-pkg-config"
export DISTCC_HOSTS="192.168.0.1/8 127.0.0.1"
```

DISTCC_HOSTS

- 複数の IP アドレスを追加することができます。左から右方向に、能力の高いサーバーから低いサーバーの順に並べてください。
- DISTCC_HOSTS に設定した順で処理を振り分けます。対象の IP アドレスの distcc と接続できない場合、次に指定した IP アドレスに処理が振り分けられます。
- 192.168.0.1/8 の 8 は同時に実行するスレッド数を示します。省略すると、デフォルト値として 4 が設定されます。分散先 distcc の設定ファイルの JOBS と同じ数値を指定してください。

1.5.3 分散ビルドの実行

make を用いる場合の並行処理指定

Makefile があるビルド環境を前提に説明します。-j に続けて数値を指定することで、並行処理の数を指定することが可能です。

gateway

```
admin@amnimo:~$ make -j8 ↵
```

分散ビルド状態の確認

distcc の動作状況は、以下のコマンドで確認することができます。

gateway

```
admin@amnimo:~$ watch distccmon-text ↵
```

正常動作

```
  8999  Compile     yyy.c           127.0.0.1[1]
  9100  Compile     zzz.c           192.168.0.1[0]
  9094  Compile     aaa.c           192.168.0.1[1]
  (Ctrl＋C で終了)
```

第2章　エッジゲートウェイのファームウェア

本章では、ユーザーが作成した独自のエッジゲートウェイ用アプリケーションをファームウェアとして提供する方法や、パッケージとして提供する方法について説明します。

2.1　エッジゲートウェイのファイルシステム構成と冗長化

エッジゲートウェイでは、設定、Linux カーネル、rootfs、userfs が冗長化されており、ファームアップ実施中の電源断などで中断されても、ファームアップ実施前の状態を保持できるようになっています。

2.1.1　ファイルシステム構成について

以下にエッジゲートウェイのメモリデバイスとファイルシステムの対応について記載します。

メモリデバイス	デバイスファイル	マウントポイント	容量	ファイルシステム	冗長領域※1	内容
eMMC	/dev/mmcblk0boot0	/mnt/config/area0	2.9MB	ext4	AREA0	Linux カーネル、設定保存領域
	/dev/mmcblk1boot0	/mnt/config/area1	2.9MB	ext4	AREA1	
	/dev/mmcblk0p1	/	8.3GB	ext4	AREA0	rootfs 領域
	/dev/mmcblk0p2	/	8.3GB	ext4	AREA1	
	/dev/mmcblk0p3	/opt/local	4.1GB	ext4	AREA0	userfs 領域
	/dev/mmcblk0p4	/opt/local	4.1GB	ext4	AREA1	
	/dev/mmcblk0p5	/mnt/share※2	3.5GB	ext4	-	共有領域
SSD※3	/dev/sda1	Ex. /media/ssd	64GB ～ 4TB※4	ext4	-	NxWitness 用データ保存領域

※1 エッジゲートウェイは冗長領域を保有しています。冗長領域の管理方法については、『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『2.4　ファームウェアを操作する』を参照してください。

※2 /mnt/share/log は/var/log, /mnt/share/common は/opt/common としてマウントされています。

※3 SSD はオプションです。またマウントポイントやファイルシステムはユーザー様で選択が可能です。上記記載内容は１つの例になります。SSD を有効にする方法については、『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『第 4 章　ストレージの操作』を参照してください。

※4 屋内タイプのエッジゲートウェイでは SSD オプションの最大容量は 2TB になります。

2.1.2　冗長領域の切り替え条件

/etc/rc.local 起動完了前にリブートした場合は、エッジゲートウェイの管理アプリケーションが起動不良と判断し、冗長領域として現在起動している反対側の領域から起動します。

2.2 エッジゲートウェイのファームウェアイメージを作成する

まず、ゲートウェイ上にユーザーが作成したアプリケーションや必要なパッケージなどをインストールします。次に、その環境からファームウェアイメージを作成します。

2.2.3 アプリケーションやパッケージのインストール

作成したアプリケーションを、rootfs 領域や userfs 領域に保存します。

アプリケーションの動作に必要なパッケージは、以下のコマンドでインストールします。

gateway

```
admin@amnimo:~$ sudo apt update ↵
admin@amnimo:~$ sudo apt install <package name> ↵
```

2.2.4 ファームウェアイメージの作成

以下のコマンドで、指定した冗長領域からファームウェアイメージを作成します。コマンドを実行すると、シャットダウンが開始され、シャットダウン中に rootfs 領域および userfs 領域を含むファームウェアイメージが作成されます。

書式（バージョン 1.4.5 以前）

```
amfwgen snapshot --target=<target> --partition=<partition> <amf>
```

書式（バージョン 1.5.0 以降）

```
amfwgen snapshot --target=<target> --partition=<partition> [--exclude=<exclude>] [--de
vice=<device>] <amf>
```

設定項目

<table>
<tr><th>項目</th><th>内容</th></tr>
<tr><td>target</td><td>ファームウェアイメージを作成する冗長領域を指定します。
<table>
<tr><th>表示</th><th>内容</th></tr>
<tr><td>fore</td><td>現在起動している冗長領域を指定します。
target を省略した場合のデフォルト値として使用されます。</td></tr>
<tr><td>back</td><td>現在動作していない冗長領域を指定します。</td></tr>
</table></td></tr>
<tr><td>partition</td><td>ファームウェアイメージを作成する冗長領域を指定します。
<table>
<tr><th>表示</th><th>内容</th></tr>
<tr><td>rootfs</td><td>rootfs を指定します。</td></tr>
<tr><td>userfs</td><td>userfs を指定します。</td></tr>
</table></td></tr>
<tr><td>exclude</td><td>作成するファームウェアイメージから除外したいファイルやディレクトリを記載したファイルへのパスを指定します。ファイルには、一行毎に一つのファイルやディレクトリへのパスを記載したテキストファイルを指定してください。本オプションは、生成したファームウェアイメージを複数の機器にインストールする場合に、機器固有のファイルやパスワード/署名ファイルを除外するために使用することを想定しています。
  本オプションはバージョン 1.5.0 以降で利用可能です。</td></tr>
<tr><td>device</td><td>作成するファームウェアイメージを保存するストレージを示すデバイスファイルを指定します。指定しない場合は、デフォルト値として現在起動している冗長領域のデバイスファイルが指定されます。デバイスファイルは/dev/XXX の形式で指定してください。ファームウェアイメージは、指定したストレージ上の<amf>で指定したパスに保存されます。
  本オプションはバージョン 1.5.0 以降で利用可能です。</td></tr>
</table>

項目	内容
amf	ファームウェアイメージの出力先を指定します。再起動すると、ここで指定したパスに、ファームウェアイメージが生成されます。 バージョン 1.4.5 以前のファームウェアを使用している場合、出力先には rootfs 上のパスを指定してください。userfs/sharefs や SD カードのマウント先を指定することはできません。

- 本コマンドを実行すると、シャットダウンが開始されます。
- 本コマンドを実行してから再起動までにかかる時間は、冗長領域の使用サイズによって変化します。通常、 1 時間から数時間かかります。
- 事前に冗長領域の使用サイズを削減してからコマンドを実行することをおすすめします。

- 本コマンドで作成したファームウェアは CLI のファームウェア更新機能（『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『2.4 ファームウェアを操作する』）で利用可能ですが、バージョン 1.4.5 以前の場合 GUI のファームウェア更新機能では利用不可です。バージョン 1.5.0 以降に更新後、本コマンドで作成したファームウェアは、GUI のファームウェア更新機能でも利用可能です。

実行例（バージョン 1.5.0 以降）

現在起動している領域の rootfs と userfs を含むファームウェアイメージを生成します。ファームウェアイメージは、/dev/sda1 上のファイルシステムのトップディレクトリに保存されます。ファームウェアイメージには、/etc/amnimo/archive.list に記載されたファイルは含まれません。

gateway

```
admin@amnimo:~$ cat /etc/amnimo/archive.list ← 機器固有のファイルリスト
/etc/amnimo/archive.list
/etc/amnimo/config.yaml
/etc/machine-id
/etc/ssh/ssh_host_ecdsa_key
/etc/ssh/ssh_host_ecdsa_key.pub
/etc/ssh/ssh_host_ed25519_key
/etc/ssh/ssh_host_ed25519_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub
/etc/amnimo/dimoni.conf
/opt/networkoptix/mediaserver/var/ssl/cert.pem
/etc/remoteit/config.json
admin@amnimo:~$ sudo amfwgen snapshot --target=fore --partition=rootfs,userfs --exclude
=/etc/amnimo/archive.list --device=/dev/sda1 firmware.amf ↵
```

第3章　エッジゲートウェイの活用

本章では、エッジゲートウェイのアプリケーションを組み込む上での活用方法について紹介します。

3.1　フック処理の活用

エッジゲートウェイには、特定のケースにおいてユーザー独自の処理をフックする仕組みがあります。以下のディレクトリ配下に存在するスクリプトファイルが実行されます。

フック処理一覧

機能	ディレクトリ	実行条件
イーサネット インターフェイス	/etc/amnimo/if-up.d	インターフェイスのリンクアップ
	/etc/amnimo/if-down.d	インターフェイスのリンクダウン
PoE	/etc/poe/hook/temp-alert-occur.d	PoE コントローラの温度異常発生
	/etc/poe/hook/temp-alert-clear.d	PoE コントローラの温度異常復旧
PPP	/etc/ppp/ip-up.d	IP アドレス割当
	/etc/ppp/ip-down.d	IP アドレス解放
通信モジュール	/etc/ecm/if-up.d	回線接続
	/etc/ecm/if-down.d	回線切断
低電圧監視機能	/etc/amnimo/uvol-detecion.d	低電圧検出
	/etc/amnimo/uvol-recovery.d	低電圧復旧
温度監視	/etc/amnimo/thermal-detection.d	温度異常発生
	/etc/amnimo/thermal-restoration.d	温度異常復旧
デジタル入力 (D IN)	/etc/amnimo/dimoni.d	各デジタル入力端子の変化 ➔ 詳細は『3.4.3 Nx Witness 以外のアプリとの連携』を参照ください。

スクリプトファイルの名称は、ASCII の大文字と小文字、ASCII の数字、および ASCII のアンダースコア（_）と ASCII のハイフンマイナス（-）で構成されている必要があります。

3.2 systemd を利用したサービス管理

Ubuntu では、サービス管理機能として systemd が採用されています。そのため、ユーザーが用意したアプリケーションについても systemd のサービスタイプのユニットファイルを用意することで起動管理することが可能です。

ここでは、『1.3.2』のサンプルプログラム amnimo-sample-app をサービスとする場合に、systemd のユニットファイルを作成する方法について説明します。

3.2.1 ユニットファイルの作成

amnimo-sample-app のユニットファイル（amnimo-sample-app.service）を、/lib/systemd/system 配下に作成します。

amnimo-sample-app.service

```
[Unit]
Description= amnimo sample application
After=syslog.target network.target
RequiresMountsFor＝/media/ssd

[Service]
Type=simple
ExecStart=/usr/sbin/amnimo-sample-app
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Unit セクション

ユニットの説明、依存関係、順序関係など、ユニットのタイプに依存しない項目を記載します。

パラメーター	内容	サンプルの設定例
Description	プログラムに関する説明を記載します。	amnimo sample application
After	指定したユニットリストが起動したあとに、本ユニットを実行します。	● syslog.target syslog にログを保存するために、syslog のあとに本サンプルプログラムを起動します。 ● network.target ネットワーク接続を前提とする場合は、追記します。
RequiresMountsFor	本ユニットの動作に必要なマウントポイントを指定します。	/media/ssd のマウント完了を待ちます。

Service セクション

サービスユニット固有の設定を記載します。

パラメーター	内容	サンプルの設定例
Type	サービスユニットの起動方法を指定します。	simple プロセスが開始した時点で本ユニットサービスが開始したと判断します。
ExecStart	サービスを起動するコマンドを指定します。	/usr/sbin/amnimo-sample-app （本サンプルコマンド）
Restart	サービスのプロセスが停止した場合の再起動条件を設定します。	always 終了理由に関わらず再起動します。
RestartSec	サービスを再起動する場合、サービス停止後、再起動するまでの時間を指定します。	5 秒

Install セクション

指定したターゲットの自動起動を有効化したときにユニットが起動するように、依存関係を記載します。

パラメーター	内容	サンプルの設定例
WantedBy	サービスユニットを enable にしたときに、設定したユニットの.wants ディレクトリにユニットファイルのリンクが作成されます。	multi-user.target （複数ユーザー環境用のターゲット、systinit のランレベル 2〜4 に相当）

ユニットファイルをカスタマイズする場合は、以下のサイトを参照してください。

- ユニットファイル
  https://www.freedesktop.org/software/systemd/man/systemd.unit.html
- サービスユニット
  https://www.freedesktop.org/software/systemd/man/systemd.service.html

3.2.2　サービスの状態制御と状態管理

作成したサービスユニットを起動し、動作確認します。

ユニットファイル修正後の操作

ユニットファイルを修正した場合は、以下のコマンドを実行し、設定内容を反映する必要があります。

gateway

```
admin@amnimo:~$ sudo systemctl daemon-reload ↵
```

サービスユニットの起動と状態確認

サービスを起動する場合は、以下のように実行します。

gateway

```
admin@amnimo:~$ sudo systemctl start amnimo-sample-app ↵
```

サービスユニットが正常に起動しているかどうかは、以下のように確認することができます。

gateway

```
admin@amnimo:~$ sudo systemctl status amnimo-sample-app ↵
● amnimo-sample-app.service - amnimo sample application
   Loaded: loaded (/lib/systemd/system/amnimo-sample-app.service; disabled; vendor pres
et: enabl
   Active: active (running) since Wed 2020-08-19 18:22:05 JST; 10s ago
 Main PID: 30823 (amnimo-sample-a)
   CGroup: /system.slice/amnimo-sample-app.service
           └─30823 /usr/sbin/amnimo-sample-app
. . .
```

ユニットの有効化／無効化

システム起動時に自動起動したい場合は、ユニットを有効化します。

gateway

```
admin@amnimo:~$ sudo systemctl enable amnimo-sample-app ↵
```

ユニットを無効化する場合は、以下のように実行します。

gateway

```
admin@amnimo:~$ sudo systemctl disable amnimo-sample-app ↵
```

3.3　パッケージの作成をする

エッジゲートウェイではパッケージ管理システムを利用できます。ユーザーが作成したアプリケーションをパッケージとして管理することで、パッケージ同士の依存関係の管理することも可能です。

ここでは、ネイティブ開発環境でパッケージを作成する方法について説明します。

パッケージ管理システムの詳細については、Ubuntu のサイトで詳細を確認してください。
https://ubuntu.com/server/docs/package-management

3.3.1　準備

必要なアプリケーションをインストールします。

関連パッケージのインストール

gateway

```
admin@amnimo:~$ sudo apt update ↵
admin@amnimo:~$ sudo apt upgrade ↵
admin@amnimo:~$ sudo apt install devscripts cdbs debhelper dh-make ↵
```

サンプルアプリケーションのソースコード

サンプルプログラムには、『1.3.2』のサンプルコードを利用します。また、サービスのユニットファイルには、『3.2.1』で作成したユニットファイルを利用します。

main.c

```
#include <stdio.h>
#include <unistd.h>

int main(void)
{
    while(1){
        printf("Hello World!¥n");
        sleep(10);
    }
    return 0;
}
```

Makefile

『1.3.2』の Makefile に、サービスユニットのインストール処理を追記します。

```
CC = gcc
CFLAGS = -O3 -Wall
DESTDIR :=
LIBS =
OBJS = main.o
PROG = amnimo-sample-app
UNITFILE = amnimo-sample-app.service

all: $(PROG)
$(PROG): $(OBJS)
        $(CC) $(OBJS) $(LIBS) -o $(PROG)

clean:
        rm -f *.o $(PROG)

install: $(PROG)
        install -D -s $(PROG) $(DESTDIR)/usr/sbin/$(PROG)
        install -D -m 644 $(UNITFILE) $(DESTDIR)/lib/systemd/system/$(UNITFILE)
```

amnimo-sample-app.service

```
[Unit]
Description= amnimo sample application
After=syslog.target

[Service]
Type=simple
ExecStart=/usr/sbin/amnimo-sample-app

[Install]
WantedBy=multi-user.target
```

ディレクトリ構造とファイル配置を以下に示します。

gateway

```
admin@amnimo:~/sample$ tree ↵
.
├── amnimo-sample-app.service
├── main.c
└── Makefile
```

3.3.2　パッケージの作成

パッケージのテンプレートの作成

dh_make を使用して、サンプルアプリケーションのディレクトリ内にパッケージのテンプレートを作成します。

gateway

```
admin@amnimo:~$ cd sample ↵
admin@amnimo:~/sample$ dh_make --native --single --email support@amnimo.com --packagena
me amnimo-sample-app_1.0.0 ↵
Maintainer Name: unknown
Email-Address    : support@amnimo.com
Date             : Wed, 19 Aug 2020 10:33:40 +0000
Package Name     : amnimo-sample-app
Version          : 1.0.0
License          : gpl3
Package Type     : single
Are the details correct? [Y/n/q]          ← 「Y」キーに続けて Enter を入力し、一旦終了する
admin@amnimo:~/sample$
```

上記の実行例で使用している dh_make のオプションを以下に示します。

オプション	設定内容
--native	Ubuntu 準拠のパッケージの作成
--single	アプリケーションの作成
--email	各ファイル上のメールアドレスの設定
--packagename	各パッケージファイルのパッケージ名

また、上記の例で設定している各パラメーターを以下に示します。

パラメーター	設定内容
パッケージ名	amnimo-sample-app
バージョン	1.0.0  セマンティックバージョニング 2.0.0 に従って指定します。 https://semver.org/lang/ja/
メールアドレス	support@amnimo.com

パッケージのテンプレートを作成すると、以下に示す青字のファイルが出力されます。

gateway

```
admin@amnimo:~/sample$ tree ↵
.
├── amnimo-sample-app.service
├── debian
│   ├── amnimo-sample-app.cron.d.ex
│   ├── amnimo-sample-app.doc-base.EX
│   ├── amnimo-sample-app-docs.docs
│   ├── changelog
│   ├── compat
│   ├── control
│   ├── copyright
│   ├── manpage.1.ex
│   ├── manpage.sgml.ex
│   ├── manpage.xml.ex
│   ├── menu.ex
│   ├── postinst.ex
│   ├── postrm.ex
│   ├── preinst.ex
│   ├── prerm.ex
│   ├── README
│   ├── README.Debian
│   ├── README.source
│   ├── rules
│   └── source
│       └── format
├── main.c
└── Makefile
```

主な生成ファイル

ファイル名	内容
control	パッケージ管理ツールが使用する情報です。
copyright	パッケージの著作権やライセンスに関する情報です。 デフォルトは、GPLv3 です。 本ファイルは必須ではありません。
changelog	パッケージのバージョン番号、リビジョン、ディストリビューション、緊急度（urgency）を識別するために利用します。 以下のフォーマットに従って記載する必要があります。 http://www.debian.org/doc/debian-policy/ch-source.html#s-dpkgchangelog debchange コマンドで、フォーマットのテンプレートを取得できます。
rules	パッケージを作成するためのルールが記載されています。 基本的な Makefile の記述（all、clean、install など）の場合は、特に変更する必要はありません。

パッケージ情報をカスタマイズする場合は、Debian のサイトの詳細情報を確認してください。
https://www.debian.org/doc/manuals/maint-guide/dreq.ja.html

パッケージの作成

dh_make を実行したことで、パッケージの作成に必要なファイルは作成されています。続いて、debuild でパッケージを作成します。

gateway

```
admin@amnimo:~/sample$ debuild -us -uc -ui ↵
```

実行すると、以下に示す青字のファイルが出力されます。パッケージ関連のファイルは、実行ディレクトリの 1 つ上の階層に出力されます。

gateway

```
admin@amnimo:~/sample$ tree ../ ↵
../
├── amnimo-sample-app_1.0.0_arm64.build
├── amnimo-sample-app_1.0.0_arm64.buildinfo
├── amnimo-sample-app_1.0.0_arm64.changes
├── amnimo-sample-app_1.0.0_arm64.deb
├── amnimo-sample-app_1.0.0.dsc
├── amnimo-sample-app_1.0.0.tar.xz
├── sample
│   ├── amnimo-sample-app
│   ├── debian
│   │   ├── amnimo-sample-app
│   │   │   ├── DEBIAN
│   │   │   │   ├── control
│   │   │   │   └── md5sums
│   │   │   ├──lib
│   │   │   │   └── systemd
│   │   │   │     └──system
│   │   │   │       └── amnimo-sample-app.service
│   │   │   └── usr
│   │   │       ├── sbin
│   │   │       │   └── amnimo-sample-app
│   │   │       └── share
│   │   │          └── doc
│   │   │              └── amnimo-sample-app
│   │   │                  ├── changelog.gz
│   │   │                  ├── copyright
│   │   │                  └── README.Debian
│   │   ├── amnimo-sample-app.cron.d.ex
│   │   ├── amnimo-sample-app.doc-base.EX
│   │   ├── amnimo-sample-app-docs.docs
│   │   ├── amnimo-sample-app.substvars
│   │   ├── changelog
│   │   ├── compat
│   │   ├── control
│   │   ├── copyright
│   │   ├── debhelper-build-stamp
│   │   ├── files
│   │   ├── manpage.1.ex

（略）

│   │   └── source
│   │       └── format
│   ├── main.c
│   ├── main.o
│   └── Makefile
```

主な生成ファイル

ファイル名	内容
amnimo-sample-app_1.0.0_arm64.deb	バイナリパッケージです。 dpkg コマンドを使ってインストール／アンインストールすることができます。
amnimo-sample-app_1.0.0.dsc	ソースコード内容の概要です。
amnimo-sample-app_1.0.0_arm64.changes	リビジョンパッケージにおける変更点がすべて記載されているファイルです。
amnimo-sample-app_1.0.0_arm64.build	パッケージ生成時のログです。
amnimo-sample-app_1.0.0_arm64.buildinfo	パッケージの依存情報、ビルド時刻などが記載されています。
amnimo-sample-app_1.0.0.tar.xz	ソースコード tar アーカイブです。 sample ディレクトリ内にある debian ディレクトリの中身が含まれています。

作成したパッケージの内容確認

作成した deb パッケージのアーカイブの中身を確認します。

gateway

```
admin@amnimo:~$ dpkg -c amnimo-sample-app_1.0.0_arm64.deb
drwxr-xr-x root/root         0 2020-01-02 00:53 ./
drwxr-xr-x root/root         0 2020-01-02 00:53 ./lib/
drwxr-xr-x root/root         0 2020-01-02 00:53 ./lib/systemd/
drwxr-xr-x root/root         0 2020-01-02 00:53 ./lib/systemd/system/
-rw-r--r-- root/root       163 2020-01-02 00:53 ./lib/systemd/system/amnimo-sample-app.
service
drwxr-xr-x root/root         0 2020-01-02 00:53 ./
drwxr-xr-x root/root         0 2020-01-02 00:53 ./usr/
drwxr-xr-x root/root         0 2020-01-02 00:53 ./usr/sbin/
-rwxr-xr-x root/root      5864 2020-01-02 00:53 ./usr/sbin/amnimo-sample-app
drwxr-xr-x root/root         0 2020-01-02 00:53 ./usr/share/
drwxr-xr-x root/root         0 2020-01-02 00:53 ./usr/share/doc/
drwxr-xr-x root/root         0 2020-01-02 00:53 ./usr/share/doc/amnimo-sample-app/
-rw-r--r-- root/root       193 2020-01-02 00:53 ./usr/share/doc/amnimo-sample-app/READM
E.Debian
-rw-r--r-- root/root       145 2020-01-02 00:53 ./usr/share/doc/amnimo-sample-app/chang
elog.gz
-rw-r--r-- root/root      1412 2020-01-02 00:53 ./usr/share/doc/amnimo-sample-app/copyr
ight
```

3.3.3　パッケージのインストール／アンインストール

生成したパッケージは、dpkg コマンドでインストール／アンインストールすることができます。

パッケージのインストール

パッケージファイル名を指定します。

以下は、/home/admin 配下にあるパッケージをインストールする例です。

gateway

```
admin@amnimo:~$ sudo apt install /home/admin/amnimo-sample-app_1.0.0_arm64.deb ↵
```

インストールしたパッケージの確認

パッケージ名を指定します。

gateway

```
admin@amnimo:~$ apt list amnimo-sample-app ↵
Listing... Done
amnimo-sample-app/now 1.0.0 arm64 [installed,local]
```

パッケージのアンストール

パッケージ名を指定します。

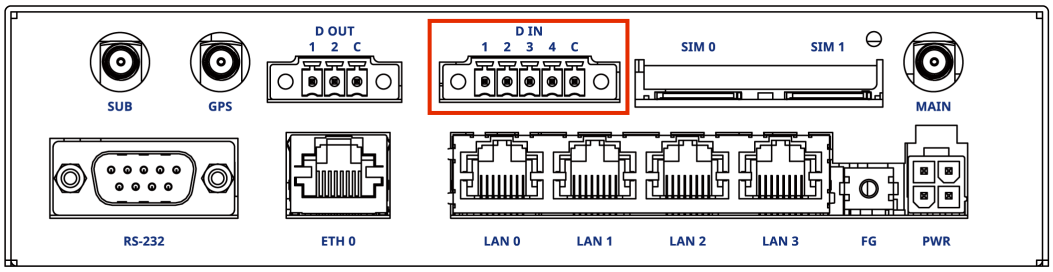
gateway

```
admin@amnimo:~$ sudo apt remove amnimo-sample-app  ↵
```

3.4　デジタル入力(D IN)と Nx Witness の連携

エッジゲートウェイには、背面のデジタル入力（D IN）端子の変化を監視し、Nx Witness にイベントとして通知するアプリケーション amnimo-dimoni がプリインストールされています。

デジタル入力（D IN）と Nx Witness と連携させることで、リアルタイムでイベントを知ることや、録画済みデータの中からイベント発生時の映像を素早く検索することができるようになります。



本機能はエッジゲートウェイのファームウェアのバージョンが V1.3.0 以降よりプリインストールされています。

➔ デジタル入力(D IN)の詳細については、『エッジゲートウェイユーザーズマニュアル』の『1.2.7　D IN／D OUT ポート』を参照してください。

3.4.1　amnimo-dimoni の設定

デジタル入力（D IN）の変化を Nx Witness に通知するための設定を行います。

amnimo-dimoni 設定ファイルの変更

/etc/amnimo/dimoni.conf ファイルを開き、NX_PASSWORD に Nx Witness の admin パスワードを設定します。

- Nx Witness の admin パスワードとは、PC の Nx Witness クライアントアプリから初めてエッジゲートウェイに接続する際に設定するパスワードです。
- Nx Witness で使用するポート番号に 7001 以外を指定した場合は NX_PORT の設定も変更してください。(デフォルトは 7001 です)

➔ 詳細については、『エッジゲートウェイ スタートアップガイド』の『Step6 VMS を設定する』を参照してください。

gateway

```
admin@amnimo:~$ sudo cat /etc/amnimo/dimoni.conf ↵

#
# Nx Witness Generic Events
#
NX_PASSWORD=PASSWORD        ←Nx Witness で使用するパスワードを設定します。
NX_PORT=7001                ←Nx Witness で使用するポート番号（デフォルト 7001）を設定します。
NX_DI_1_UP=EVENT_DI-1_UP    ↓↓↓↓↓　以下は変更する必要がありません。↓↓↓↓↓
NX_DI_1_DN=EVENT_DI-1_DN
NX_DI_2_UP=EVENT_DI-2_UP
NX_DI_2_DN=EVENT_DI-2_DN
NX_DI_3_UP=EVENT_DI-3_UP
NX_DI_3_DN=EVENT_DI-3_DN
NX_DI_4_UP=EVENT_DI-4_UP
NX_DI_4_DN=EVENT_DI-4_DN
：
```

サービスの再起動

設定を変更した場合は、amnimo-dimoni のサービスを再起動する必要があります。

gateway

```
admin@amnimo:~$ sudo systemctl restart amnimo-dimoni ↵
```

連携設定の無効化

システム起動時に amnimo-dimoni を自動起動しますが、NxWitness を無効化する場合は、下記のように NX_PASSWOARD に設定したパスワードを削除し、amnimo-dimoni のサービスを再起動する必要があります。

gateway

```
admin@amnimo:~$ sudo vi /etc/amnimo/dimoni.conf ↵

#
# Nx Witness Generic Events
#
NX_PASSWORD=                ←パスワードの設定内容を削除します。
NX_PORT=7001
NX_DI_1_UP=EVENT_DI-1_UP
NX_DI_1_DN=EVENT_DI-1_DN
NX_DI_2_UP=EVENT_DI-2_UP
NX_DI_2_DN=EVENT_DI-2_DN
NX_DI_3_UP=EVENT_DI-3_UP
NX_DI_3_DN=EVENT_DI-3_DN
NX_DI_4_UP=EVENT_DI-4_UP
NX_DI_4_DN=EVENT_DI-4_DN
 :
admin@amnimo:~$ sudo systemctl restart amnimo-dimoni ↵ ←サービスの再起動
```

設定ファイルの保存

amsh にて amnimo-dimoni 設定ファイル(/etc/amnimo/dimoni.conf)の内容をエッジゲートウェイ設定ファイルに書き込みます。この操作により、firmware area update コマンドによるファームウェアの全体更新時や現在起動していない冗長領域側で起動した場合でも設定内容を保持できます。

管理者モード 設定モード

gateway

```
amnimo# config file save startup-config ↵
```

3.4.2 Nx Witness のイベント設定

デジタル入力（D IN）の変化時に LIVE 映像画面上へのテキスト表示、録画映像へのブックマーク付与、メール送信、などのアクションを指定することができます。

Nx Witness クライアントアプリの操作

PC から Nx Witness クライアントアプリを起動し、イベントを設定します。エッジゲートウェイ接続後、画面左上の ☰ をクリックし、[システムアドミニストレーション] -> [イベントルール]の順にクリックし、イベントルール設定画面を表示します。本画面で以下の設定を行います。

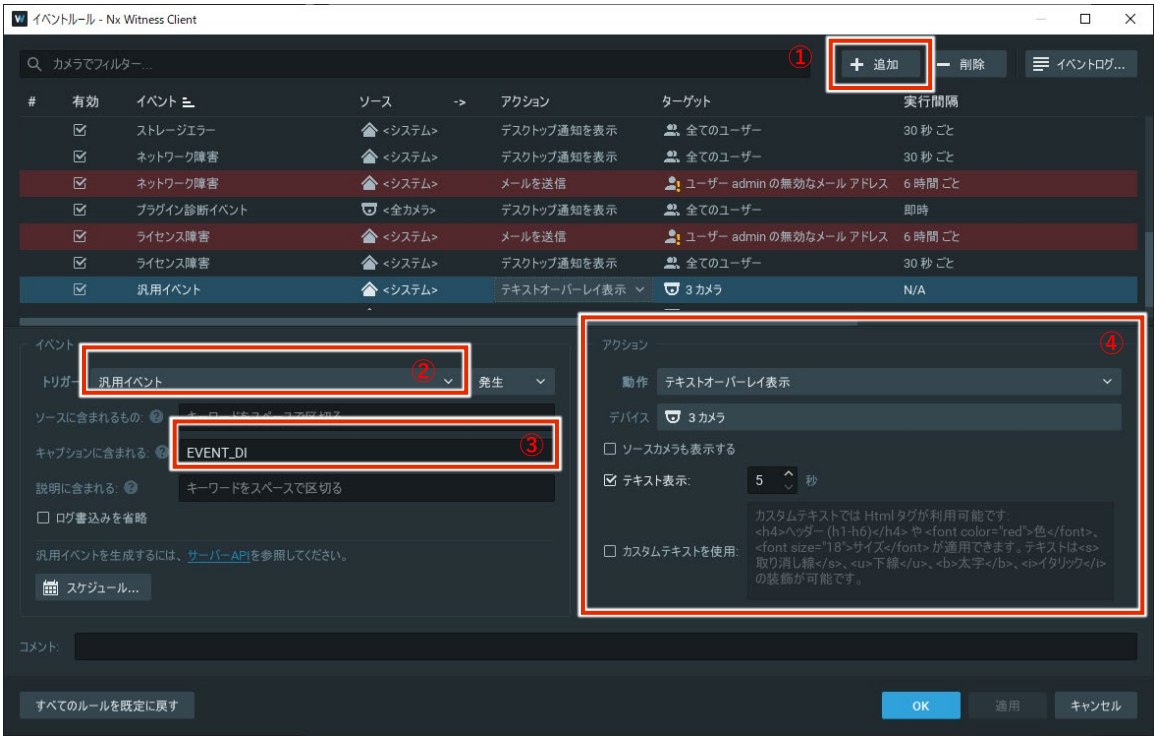
① [追加]ボタンを押し、新たにイベントを作成します。

② [トリガー]に[汎用イベント]を設定します。

③ [キャプションに含まれる]に文字列(下記一覧表)を設定し、イベントのトリガーを指定します。

④ [アクション]にはイベント発生時に実行したい動作を設定します。

イベントルール設定画面



[キャプションに含まれる]に設定可能な文字列とイベント発生のタイミングの一覧

[キャプションに含まれる]に設定可能な文字列	イベント発生のタイミング
EVENT_DI-1_UP	DI-1 が Low から High に変化
EVENT_DI-1_DN	DI-1 が High から Low に変化
EVENT_DI-2_UP	DI-2 が Low から High に変化
EVENT_DI-2_DN	DI-2 が High から Low に変化
EVENT_DI-3_UP	DI-3 が Low から High に変化
EVENT_DI-3_DN	DI-3 が High から Low に変化
EVENT_DI-4_UP	DI-4 が Low から High に変化
EVENT_DI-4_DN	DI-4 が High から Low に変化
EVENT_DI-1	DI-1 の変化（High/Low 両方）
EVENT_DI-2	DI-2 の変化（High/Low 両方）
EVENT_DI-3	DI-3 の変化（High/Low 両方）
EVENT_DI-4	DI-4 の変化（High/Low 両方）
EVENT_DI	いずれかの DI 変化（High/Low 両方）

イベントオプションの追加

Nx Witness に通知するイベントにオプションを追加することができます。イベントオプションを追加することで、より高度なイベントルールを設定することができるようになります。

- イベントオプションの追加は FW 1.5.0 以降でご利用になれます。
- 設定可能なイベントオプションの詳細については、http://<エッジゲートウェイの IP アドレス>:7001/static/index.html#/developers/events を参照ください。

イベントオプションを追加する際は/etc/amnimo/dimoni.conf ファイルを開き、各イベントのオプションを設定します。設定を有効にする場合は、amnimo-dimoni のサービスを再起動する必要があります。

例えば、以下のように設定するとデジタル入力（D IN）のイベントはレベルが High の期間中継続されるようになります。

gateway

```
admin@amnimo:~$ sudo vi /etc/amnimo/dimoni.conf ↵
 :
NX_DI_1_UP_OPT=&state=Active
NX_DI_1_DN_OPT=&state=Inactive
NX_DI_2_UP_OPT=&state=Active
NX_DI_2_DN_OPT=&state=Inactive
NX_DI_3_UP_OPT=&state=Active
NX_DI_3_DN_OPT=&state=Inactive
NX_DI_4_UP_OPT=&state=Active
NX_DI_4_DN_OPT=&state=Inactive
 :
admin@amnimo:~$ sudo systemctl restart amnimo-dimoni ↵ ←サービスの再起動
```

イベントルール設定画面

Nx Witness クライアントアプリの[アクション]設定で、秒数の指定のチェックボックスを外すことで、イベント期間中はそのアクションを維持します。

下記のように設定した場合、デジタル入力 D IN1 のレベルが High の期間中、常にテキスト表示が行われます。

Nx Witness の設定保存

amsh の設定モードで前述のイベント設定を含む Nx Witness の設定を保存します。この操作により、firmware area update コマンドによるファームウェアの全体更新時や現在起動していない冗長領域側で起動した場合でも設定内容を保持できます。

- password には Nx Witness の admin パスワードを設定してください。
- Nx Witness で使用するポート番号に 7001 以外を指定した場合は port の設定も変更してください。

設定モード

gateway

```
amnimo(cfg)# nxwitness ↵
amnimo(cfg-nxwitness)# password ↵
Enter new password:
Retype new password:
passwd: password updated successfully.
amnimo(cfg-nxwitness)# port 7001 ↵
amnimo(cfg-nxwitness)# exit ↵
amnimo(cfg)# config nxwitness save ↵
amnimo(cfg)# config file save startup-config ↵
```

3.4.3 Nx Witness 以外のアプリとの連携

amnimo-dimoni のアプリケーションフック機能を使うことで、Nx Witness 以外のアプリともデジタル入力（D IN）を連携させることが可能です。

アプリケーションフックを追加する際は/etc/amnimo/dimoni.conf ファイルを開き、D IN 端子に対応するパラメーターにコマンドやスクリプトファイルを設定します。設定を有効化するには、amnimo-dimoni のサービスを再起動する必要があります。

例えば、以下のように設定するとデジタル入力 D IN1 のレベルが High になった際に「amctrl do --on 1」が実行され、Low になった際に「amctrl do --off 1」が実行されます。(sudo は不要です)

gateway

```
admin@amnimo:~$ sudo vi /etc/amnimo/dimoni.conf ↵
 :
#
# Hooks for other applications
#
HK_DI_1_UP=amctrl do --on 1
HK_DI_1_DN=amctrl do --off 1
HK_DI_2_UP=
HK_DI_2_DN=
HK_DI_3_UP=
HK_DI_3_DN=
HK_DI_4_UP=
HK_DI_4_DN=
 :
admin@amnimo:~$ sudo systemctl restart amnimo-dimoni ↵ ←サービスの再起動
```

第4章　トラブルシューティング

本章では、エッジゲートウェイのアプリケーション開発において正常に動作しない問題が生じた場合の対処方法について説明します。

4.1　エッジゲートウェイの設定機能の活用

エッジゲートウェイの設定機能は、以下のケースで活用することができます。

- 本来動いていた動作に戻したい
- 同じエッジゲートウェイの設定を流用したい
- 動作確認のため、一部異なるエッジゲートウェイの設定を複数作成したい

エッジゲートウェイの設定には、CLI（amsh）を使用する方法と通常の Linux コンソールを使用する方法があります。本書では、CLI（amsh）を利用した方法で説明しています。
設定方法については、以下を参照してください。

➔ CLI（amsh）を使用する方法
『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『第 3 章　設定ファイルの操作』

➔ Linux コンソールを使用する方法
『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『11.1　設定ファイルを制御する』

4.1.1　デフォルトの設定ファイル

エッジゲートウェイのデフォルトの設定ファイルは、「startup-config」です。エッジゲートウェイを起動すると、このファイルが設定ファイルとして読み込まれます。

設定ファイル（拡張子：dat）は、以下のように SHA-2 のハッシュ値が記載されたファイル（拡張子：sha256）とともに保存されます。冗長管理されているため、2 か所に保存されます。

gateway

```
admin@amnimo:~$ ls -l /mnt/config/area0/ ↵
total 22
drwx------ 2 root root 12288 Feb 15  1974 lost+found
-rw-r--r-- 1 root root  3588 Aug  7 10:57 startup-config.dat
-rw-r--r-- 1 root root    89 Aug  7 10:57 startup-config.sha256
admin@amnimo:~$ ls -l /mnt/config/area1/ ↵
total 22
drwx------ 2 root root 12288 Feb 15  1974 lost+found
-rw-r--r-- 1 root root  3588 Aug  7 10:57 startup-config.dat
-rw-r--r-- 1 root root    89 Aug  7 10:57 startup-config.sha256
```

4.1.2　設定ファイルのバックアップ

設定ファイルの名前を変更する機能を利用して、バックアップを保存します。

➔ 詳しくは、『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『3.6　設定ファイルの名前を変更する』を参照してください。

実行例

管理者モードで、backup-config という名称で startup-config を保存します。

管理者モード

gateway

```
amnimo# config file copy startup-config backup-config ↵
```

バックアップされた設定ファイルを確認します。

gateway

```
admin@amnimo:~$ ls -l /mnt/config/area0/ ↵
total 22
-rw-rw-rw- 1 root root  3588 Aug  7 11:44 backup-config.dat
-rw-rw-rw- 1 root root    89 Aug  7 11:44 backup-config.sha256
drwx------ 2 root root 12288 Feb 15  1974 lost+found
-rw-r--r-- 1 root root  3588 Aug  7 10:57 startup-config.dat
-rw-r--r-- 1 root root    89 Aug  7 10:57 startup-config.sha256
admin@amnimo:~$ ls -l /mnt/config/area1/ ↵
total 22
-rw-rw-rw- 1 root root  3588 Aug  7 11:44 backup-config.dat
-rw-rw-rw- 1 root root    89 Aug  7 11:44 backup-config.sha256
drwx------ 2 root root 12288 Feb 15  1974 lost+found
-rw-r--r-- 1 root root  3588 Aug  7 10:57 startup-config.dat
-rw-r--r-- 1 root root    89 Aug  7 10:57 startup-config.sha256
```

4.1.3 設定ファイルの世代管理

任意のファイル名で保存することで、ユーザーが設定ファイルを世代管理することが可能です。

実行例

管理者モードで、設定を変えつつ、「pattern-A-config」「pattern-B-config」「pattern-C-config」という名称で各設定を保存します。最後に、設定ファイルをリスト表示します。

管理者モード

gateway

```
各種設定を終えたあとに以下を実行
amnimo# config file save pattern-A-config ↵

一部設定を終えたあとに以下を実行
amnimo# config file save pattern-B-config ↵

一部設定を終えたあとに以下を実行
amnimo# config file save pattern-C-config ↵
amnimo# show config file ↵
startup-config 2020-01-02T00:00:00+09:00
pattern-A-config 2020-01-01T00:00:05+09:00
pattern-B-config 2020-01-01T00:00:10+09:00
pattern-C-config 2020-01-01T00:00:12+09:00
```

4.1.4　設定ファイルの反映

保存したファイルを読み込んで、エッジゲートウェイに反映することができます。

同一機器内に設定ファイルが複数存在する場合

実行例

『4.1.3』で保存した 1 つの設定ファイル（pattern-A-config）を管理者モードで読み込んで、設定を反映します。

管理者モード

gateway

```
amnimo# config file load pattern-A-config ↵          ←設定を読み込む
amnimo# config file save ↵                           ←読み込んだ設定を startup-config として保存し、次の起動時の設定として反映
startup-config file already exists. Do you want to overwrite? (y/N): y ↵
amnimo# reboot type soft ↵                           ←ソフトウェアリブートの実施
```

別のエッジゲートウェイの設定ファイルを利用する場合

実行例

amsh ではなく、外部コマンドの amcfg を利用して、別のエッジゲートウェイから移行した設定ファイル（/media/sdacard/pattern-D-config）を読み込みます。

gateway

```
admin@amnimo:~$ sudo amcfg load /media/sdacard/pattern-D-config ↵    ←設定を読み込む
admin@amnimo:~$ sudo amcfg save ↵ ←読み込んだ設定を startup-config として保存し、次の起動時の設定として反映
startup-config file already exists. Do you want to overwrite? (y/N): y
admin@amnimo:~$ sudo amctrl reboot -t soft ↵         ←ソフトウェアリブートの実施
```

4.2 正常に起動しない場合の対処

エッジゲートウェイのアプリケーションを開発する上で、ユーザーがインストールしたアプリケーションが必要となる場合があります。状況によっては Linux が起動しない状態になることも考えられます。

エッジゲートウェイには冗長領域があるため、万が一起動しない状態に陥った場合は、ブートローダーで起動エリアを切り替えて起動することが可能です。

4.2.1 サンプルケースについて

この節で想定しているサンプルケースでは、以下の流れで問題に対処しています。

- エリア 0 とエリア 1 は、当初、同じ構成で運用されていた。
- エリア 0 上でアプリケーションを開発する過程で、エリア 0 側が正常に起動しなくなった。
- エリア 0 をエリア 1 と同じ構成に戻すことで復旧させる。

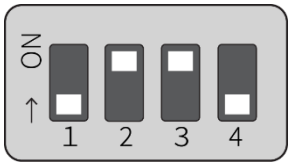
4.2.2 Step1.ブートローダー上の起動エリアの切り替え

ブートローダーで起動エリアを切り替えることができます。シリアルコンソール上で設定してください。

U-Boot コマンドモードで起動する

電源を接続する前に、DIP スイッチで「U-Boot コマンドモード」に設定し、電源を接続します。

U-Boot コマンドモードの DIP スイッチ設定



No.4 の DIP スイッチは、ON でも OFF でも構いません。ブートローダーでは参照しません（アプリケーション側でのみ使用します）。

電源を入れると、以下のパスワード入力画面（入力期間：10 秒）になりますので、パスワードを入力し、Enter キーを押してログインしてください。

gateway

```
TIM-1.0
WTMI-devel-18.12.1-118f0bd
WTMI: system early-init
SVC REV: 5, CPU VDD voltage: 1.108V
（略）
STATUS:SN=[300002],MAC0=[E8:1B:4B:00:30:02],BS=[a:0  b:385  h:0  s:0],DIPBM=[ubootcommand]
am_show_board_status: CNTFRQ_EL0=12500000 Hz

Please enter password - autoboot in 10 sec...       ←パスワード入力後、Enter キーを押す
Return to boot status(0x55) for login
Amnimo>> run stopwdt ↵            ←ウォッチドッグ IC により数分経過するとリセットするため、操作に時間がかかる場合は、停止処理を実行してください。
Amnimo>>
```

- 初期パスワードについては、弊社サポートにご確認ください。
- パスワードの入力の失敗は 3 回までです。3 回以上失敗すると、Linux 起動モードで起動します。

U-Boot で起動エリアを切り替える

ambootsw コマンドを利用して、起動エリアを切り替えることができます。起動エリアを切り替えたあとに、reset コマンドで再起動します。

書式

```
ambootsw < get | set <0|1> >
```

設定項目

<table>
<tr><th>項目</th><th>内容</th></tr>
<tr><td>get</td><td>起動エリアを確認します。

出力フォーマット

Boot Area: AREA_NO
Boot Count: BOOT_COUNT

<table>
<tr><th>項目</th><th>表示</th><th>内容</th></tr>
<tr><td>AREA_NO</td><td>0 (1st Area)</td><td>エリア 0
設定保存領域： /dev/mmcblk0boot0
rootfs： /dev/mmcblk0p1
userfs： /dev/mmcblk0p3</td></tr>
<tr><td></td><td>1 （ 2nd Area）</td><td>エリア 0
設定保存領域： /dev/mmcblk0boot0
rootfs： /dev/mmcblk0p1
userfs： /dev/mmcblk0p3</td></tr>
<tr><td>BOOT_COUNT</td><td>数値</td><td>起動回数</td></tr>
</table>
</td></tr>
<tr><td>set AREA_NO</td><td>起動エリアを切り替えます。

<table>
<tr><th>項目</th><th>内容</th></tr>
<tr><td>AREA_NO</td><td>起動エリアの番号を指定します。</td></tr>
</table>
</td></tr>
</table>

実行例

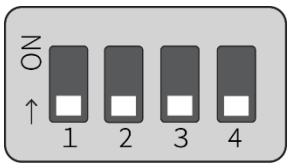
gateway

```
Amnimo>> ambootsw get ↵          ←現在の起動エリアを確認
Boot Area : 0 (1st Area)         ←エリア 0
Boot Count : 385
Amnimo>> ambootsw set 1 ↵        ←起動エリアをエリア 1 に変更
Amnimo>> ambootsw get ↵          ←再度の起動エリアを確認
Boot Area : 1 (2nd Area)         ←エリア 1
Boot Count : 385
```

Linux 起動モードで起動する

再起動する前に、DIP スイッチで「Linux 起動コマンドモード」に設定し、再起動処理を行います。

Linux 起動モードの DIP スイッチ設定



No.4 の DIP スイッチは、ON でも OFF でも構いません。ブートローダーでは参照しません（アプリケーション側でのみ使用します）。

gateway

```
Amnimo>> reset ↵          ←再起動
```

4.2.3 Step2. Linux 起動後の起動エリアの確認

起動エリアは、Linux 起動後に確認することができます。再起動したあとに、U-Boot 側で設定した起動エリアになっていることを確認してください。

➔ 詳細については、『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『2.4.6 起動する冗長領域を設定する』もしくは『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『11.2.5 起動エリアを制御する』を参照してください。

実行例

amctrl コマンドの場合は、以下のように実行します。

gateway

```
admin@amnimo:~$ sudo amctrl boot ↵
AREA: 1
```

amsh の場合は、以下のように実行します。以下は、一般ユーザーモードでの実行例です。

ユーザーモード 管理者モード

gateway

```
amnimo$ show device boot ↵
1
```

4.2.4 Step3.ファームエリアの同期

起動エリアのエリア 1 側で正常起動したことを確認できたため、エリア 1 側のファームウェアの内容をエリア 0 側に同期し、エリア 0 側を正常に起動するようにします。

➔ 詳細については、『エッジゲートウェイシリーズ CLI ユーザーズマニュアル』の『2.4.5 ファームウェアの冗長エリアを同期する』を参照してください。

実行例

管理者モード

gateway

```
amnimo# firmware area sync ↵
reboot to sync? (y/N):          ←「y」キーに続けて Enter を入力
amnimo# device boot 0 ↵         ←起動エリアを 0 に戻す
amnimo# reboot type soft↵       ←ソフトウェアリブートを行う
```

firmware area sync コマンドの実行をキャンセルするには、Enter を入力するか、「n」キーに続けて Enter を入力します。

改訂履歴

第 1 版	2020 年 11 月発行
第 2 版	2020 年 12 月発行
第 3 版	2021 年 3 月発行
第 4 版	2021 年 9 月発行
第 5 版	2022 年 1 月発行
第 6 版	2022 年 5 月発行

amnimo™

エッジゲートウェイデベロッパーズマニュアル
2022 年 5 月 6 日　第 6 版

IM AMD03A01-51JA