D23-0011
ユーザーマニュアル

Version 1.0
2025/03/21

株式会社ディーディーエル

目次

第1章 本マニュアルについて

本マニュアルについて株式会社ディーディーエルが提供するWi-SUN FANモジュールについての情報を記載するものである。本製品に搭載するＩＣ及びベースになるプロトコルスタックはルネサス製の物を使用している為、一部ツール及びドキュメントはルネサス製の物を参照している。

第2章 要約

Wi-SUN 業界アライアンスは、ワイヤレス スマート ユーティリティ ネットワーキングに重点を置き、ワイヤレス IEEE 802.15.4-2015 PHY および MAC レイヤーを使用したプロトコル スタックを定義する技術プロファイル仕様書を準備しました。結果として得られる仕様は、ネットワーク参加手順、セキュリティ メカニズム、ルーティング プロトコルなど、さまざまな側面をカバーしています。この要件を満たす技術実装の 1 つが、Wi-SUN FAN（フィールド エリア ネットワーク）仕様です。

Wi-SUN FAN 仕様は、次の機能をカバーしています。

- IEEE 802.15.4 SUN 物理層
- 周波数ホッピング、ネットワーク検出/参加、プロトコルディスパッチ
- 6LoWPAN、アドレス管理、RPLを使用したルーティング、ユニキャストおよびマルチキャスト転送を含むIPv6プロトコルスイート
- 認証、承認、暗号化を含む標準ベースの多層セキュリティ仕様。

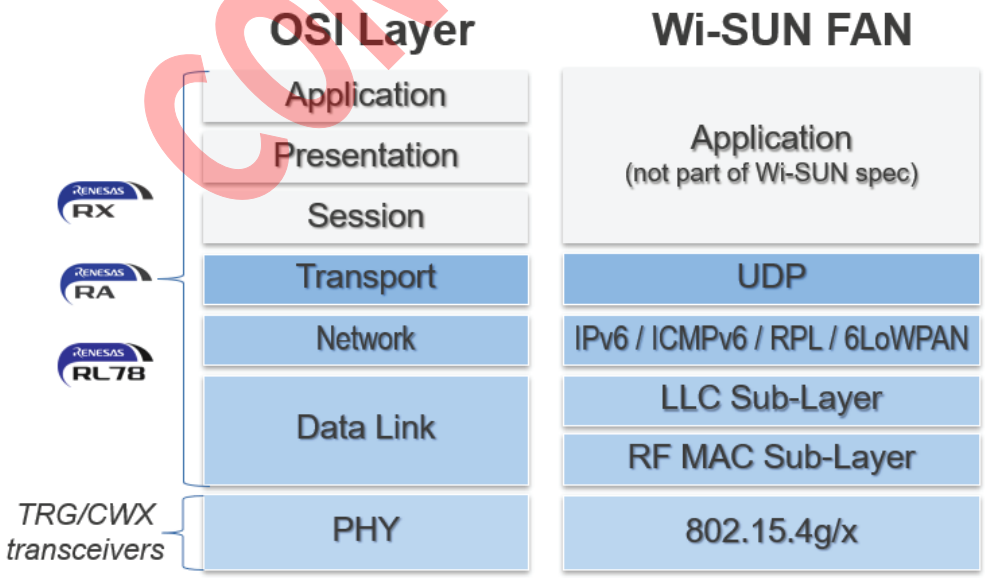


図2-1: Wi-SUN FANプロトコルスタックのレイヤ

第3章 略語

ADC	アナログ-デジタルコンバータ
AES	高度暗号化標準
AFE	アナログフロントエンド
AGC	自動ゲインコントロール
AMM	自動メーター管理
API	アプリケーションプログラミングインターフェース
AWGN	加法性ホワイトガウスノイズ
BPF	バンドパスフィルター
BR	ボーダールータ
CRC	巡回冗長検査
DAC	デジタル-アナログコンバータ
DAG	有向非巡回グラフ
DFE	有向フレーム交換
DHCPv6	IPv6 の動的ホスト構成プロトコル
DLL	データリンク層
DODAG	目的地指向DAG
EDFE	拡張有向フレーム交換
FAN	フィールドエリアネットワーク
FER	フレームエラー率
FFN	フル機能ノード
GUI	グラフィカル・ユーザー・インターフェース
HW	ハードウェア
IETF	インターネットエンジニアリングタスクフォース
IP	インターネットプロトコル
LBR	低電力で損失の少ないボーダールーター（BR と同じ）
LFN	限定機能ノード
LLN	低消費電力とロスのあるネットワーク
LPF	ローパスフィルタ
LSI	大規模統合
MAC	Media Access Control
MCU	マイクロコントローラユニット
MP	Mass Production
MBM	狭帯域ノイズ
NVM	不揮発性メモリ
PA	パワーアンプ
PHY	物理層
PIB	パーソナルエリアネットワーク情報ベース
PLC	電力線通信

PMK	ペアワイズマスターキー
QA	品質保証
RADIUS	リモート認証ダイヤルインユーザーサービス
RN	ルーターノード
RPL	低電力および損失の多いネットワーク向けの IPv6 ルーティング プロトコル
SNR	信号対雑音比
SW	ソフトウェア
TX	送信機
RTOS	リアルタイムオペレーティングシステム
RX	受信機
Wi-SUN	ワイヤレススマートユーティリティネットワーク
WS	実用的なソリューション

第4章 はじめに

このドキュメントでは、ディーディーエル提供のFANモジュール及びルネサス社対応の評価ボード上で実行される Wi-SUNFAN プロトコル スタックの詳細な説明を提供します。

DDL社製

通信モジュール（D23-0011）、ボーダールーター及びルーターノード用モジュール。

通信モジュール（D15-0013）、リーフノード用モジュール

開発ボード、USBドングル

※開発ボード及びUSBドングルは、通信モジュール（D23-011、D15-0013）とは共通プラットフォームとなっている。

ルネサス提供の

TK-RLG1H+SB2評価ボード

MB-RA604S-02評価ボード

RTK0EE0013D評価ボード

第5章 参考資料

以下は、このユーザー マニュアルと併せて使用する参考資料です。

タイトル	説明	位置
Wi-SUN FAN Technical Profile Specification v1.1v08	Technical implementation and behavior of the Wi-SUN Field Area Network 1.1v08	[1]
20150727-PHY-Wi-SUN_PHY_Specification-1V09	Technical description of the PHY operating modes for Wi-SUN 1V09	[2]
20200609-PHY-Profile-Wi-SUN_PHY_Specification-Amendment-1VA10	Wi-SUN PHY Technical Specification - Amendment 1VA10	[3]
R30UW0055EJ0106_SubGHz_RfDriverApi_SettingTable	Sub-GHz RF Driver, Channel, Transmission Power Setting Table, Rev.1.06 July 2022	[4]
RX651 PLC/RF Evaluation Board Programming	Flash programming instructions for the Tessera MB-RA604S-02 and Y-G-HYBRID-PLC-RF boards	[5]
RL78/G1H RF Evaluation Board Programming	Flash programming instructions for the Tessera TK-RLG1H+SB and TK-RLG1H+SB2 evaluation boards	[6]
How to Update Firmware	RTK0EE0013D10001BJ Firmware Update Instructions (Evaluation kit for North America)	[7]
How to Update Firmware	RTK0EE0013D10002BJ Firmware Update Instructions (Evaluation kit for Europe)	[8]
How to Update Firmware	RTK0EE0013D10003BJ Firmware Update Instructions (Evaluation kit for Japan)	[9]
Wi-SUN FAN UI Tool	User Manual for the Renesas Wi-SUN UI (Developer UI + Demonstrator UI)	[10]
csmp-agent-lib	CSMP agent (client) implementation and documentation	[11]
Renesas Wi-SUN FAN Border Router Gateway	Release package that contains all the software and documentation required to setup and configure the border router gateway Note: CWX-based evaluation kits that run with a FAN stack version of 1.8.0 or higher can only be used with a BR gateway package of version 1.2.0 or higher!	[12]
CSMP RFC Draft	Current RFC draft for the CSMP protocol	[13]
Wi-SUN FAN -> CSMP Mappings	Spreadsheet to illustrate which Wi-SUN FAN stack is used to create CSMP objects	[14]
Wi-SUN FAN -> DLMS Mappings	Spreadsheet to illustrate which Wi-SUN FAN stack is used to create DLMS objects	[15]

表5-1 参考文献

[1] https://ws.wi-sun.org/wg/FAN_WG/document/16144
[2] https://ws.wi-sun.org/wg/WS/document/15809
[3] https://ws.wi-sun.org/wg/PHY_WG/document/16010
[4] リリースパッケージ内のファイル「R30UW0055EJ0106_SubGHz_RfDriverApi_SettingTable.pdf」
[5] リリースパッケージ内のファイル「r11uz0266ed0100-rx651-plc-rf-programming.pdf」
[6] リリースパッケージ内のファイル「r11uz0294gd0100-rl78-glh-rf-programming.pdf」
[7] https://www.renesas.com/document/sws/howtoupdatefirmwarena
[8] https://www.renesas.com/us/ja/document/sws/how-update-firmware-rtk0ee0013d10002bj
[9] https://www.renesas.com/document/sws/howtoupdatefirmwarejp
[10] リリースパッケージ内のファイル「r11qs0022ed0163-Wi-SUN-FAN-UI-Tool.pdf」
[11] https://github.com/CiscoDevNet/csmp-agent-lib
[12] https://www.renesas.com/document/sws/wi-sun-fan-br-gateway
[13] https://datatracker.ietf.org/doc/draft-duffy-csmp
[14] リリースパッケージ内のファイル「csmp_wisun_mapping.pdf」
[15] リリースパッケージ内のファイル「dlms_wisun_mapping.pdf」

第6章 ソフトウェアプロジェクトのセットアップ

次のセクションでは、RL78/G1H マイクロコントローラおよび RX65x マイクロコントローラ用の Wi-SUN FAN プロトコル スタック ソフトウェアを構築するためのソフトウェア プロジェクトのセットアップについて説明します。

※下記プロトコルスタックの入手にはルネサスとのソフトウェア利用許諾契約の締結が必要になします。

6.1 e²studio 開発（RX65x、RL78）

次のセクションでは、ライブラリまたはソース コード バージョンとして提供される Wi-SUN FAN プロトコル スタック ソフトウェアを構築するためのソフトウェア プロジェクトのセットアップについて説明します。プロトコル スタックのライブラリまたはソース コード バージョンは、RX65x ベースのソリューションでのみ使用できることに注意してください。

6.1.1 e²studio ツールチェーンのインストール

Wi-SUN FAN プロトコル スタックは、主に Renesas e²studio IDE 用に開発された Wi-SUN IPv6 プロトコル用に開発された IPv6 スタックに基づいています。対応するインストーラは、Renesas の Web サイト（ https://www.renesas.com ）から入手できます。e²studio のインストール中に、対象のマイクロコントローラ ファミリに適したコンパイラ パッケージや追加のプラットフォーム固有のツールなど、目的のプラットフォームに必要なすべてのコンポーネントがインストールされていることを確認してください。表 6-1 に、各ターゲット プラットフォームに必要なすべてのソフトウェア コンポーネントと、そのサポート/テスト済みバージョンを示します。

商品名	*説明*	*バージョン*	*必須*
e²スタジオ	統合開発 ルネサスMCU向け環境 (Eclipse ベース)	2023-07 (23.7.0)	すべてのプラットフォーム
CC-RX	RX MCU用コンパイラパッケージ	V3.06.00	RX65x
CC-RL	RL MCU 用コンパイラ パッケージ	V1.12.01	RL78/G1H
GCC ARM 組み込み	ARM MCU 用コンパイラ パッケージ	12.2.1.arm-12mpacbti-34	RA
FSP	柔軟なソフトウェアパッケージ	4.6.0	RA

表6-1 ツールバージョン情報

e²studio IDE と必要なコンパイラが適切にインストールされたら、Renesas コンパイラ (CC-RX および CC-RL) のライセンスもインストールされていることを確認します（ *Renesas License Manager*ツールを使用して）。

6.1.2 e²studioでのプロジェクト設定

Wi-SUN FAN プロトコル スタックは、ソース コードまたはライブラリ バージョンのいずれかとして提供されることに注意してください。相関関係は次のとおりです。

¥Wi_SUN_FAN_RX_FW Wi-SUN FAN プロトコル スタック ソース コード バージョン

¥Wi_SUN_FAN_RX_FW_LIB Wi-SUN FAN プロトコルスタックライブラリバージョン

両方のフォルダには、異なるターゲットプラットフォーム（RX65x、RAなど）用の異なるe²studioプロジェクトのサブディレクトリが含まれています。プロジェクトをロードするには、e²studio IDEの「インポート」機能（メニュー「ファイル」->”インポート…” を選択します。
インポート ウィザードの選択から “一般” を選択し、次に “既存のプロジェクトをワークスペースに” を選択します。 “次へ >” ボタンを押します。 “プロジェクトのインポート” ダイアログで “ルート ディレクトリを選択” を選択し、 “参照…” ボタンを押してプロジェクトの場所に移動します。

図 6-1 に示すような画面が表示され、「プロジェクト」ウィンドウ領域に*Wi_SUN_FAN_RX_FWプロジェクト*または*Wi_SUN_FAN_RX_FW_LIB*プロジェクトのいずれかが表示されます。

プロジェクトのインポートを終了するには、「完了」ボタンを押してください。

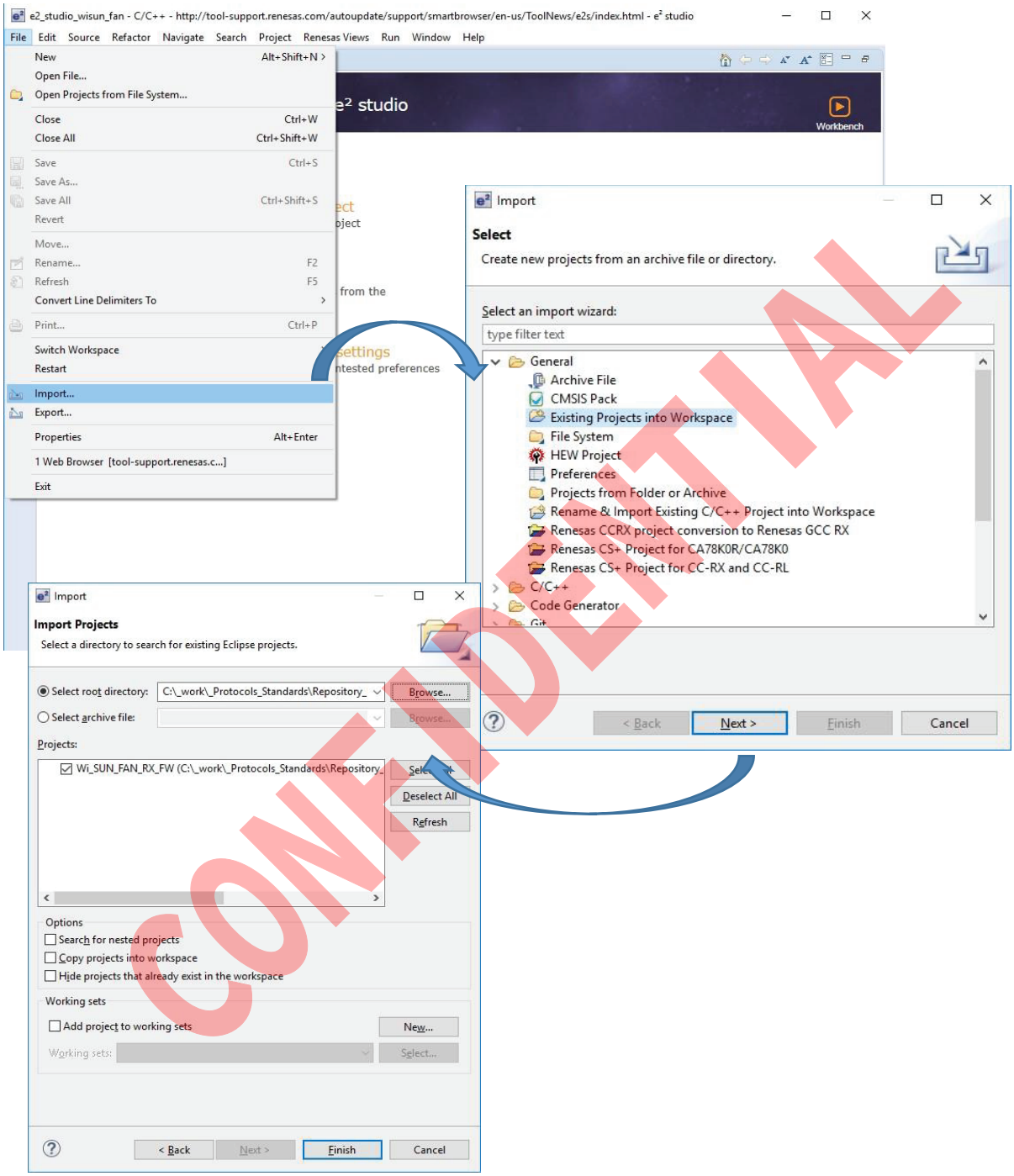


図6-1 e²studioプロジェクトのインポート

プロジェクトが e²studio IDE に正常にインポートされると、図 6-4 に示すような画面が表示されます。

同じ手順を使用して、RL78/G1H (Wi_SUN_FAN_RL_FW)および RA プラットフォーム(Wi_SUN_FAN_RA_FW)のプロジェクトをロード/インポートできます。

6.1.3 e²studio によるデバイスのセットアップとデバッグ

6.1.3.1 Wi-SUN FAN　無線モジュール評価機

無線モジュール評価機もしくはUSBドングルでプログラムをデバッグするには、次の手順に従います。

- 評価機とデバッガーを図 6-2もしくはドングルとデバッカーを図6-3 に示すように接続します。

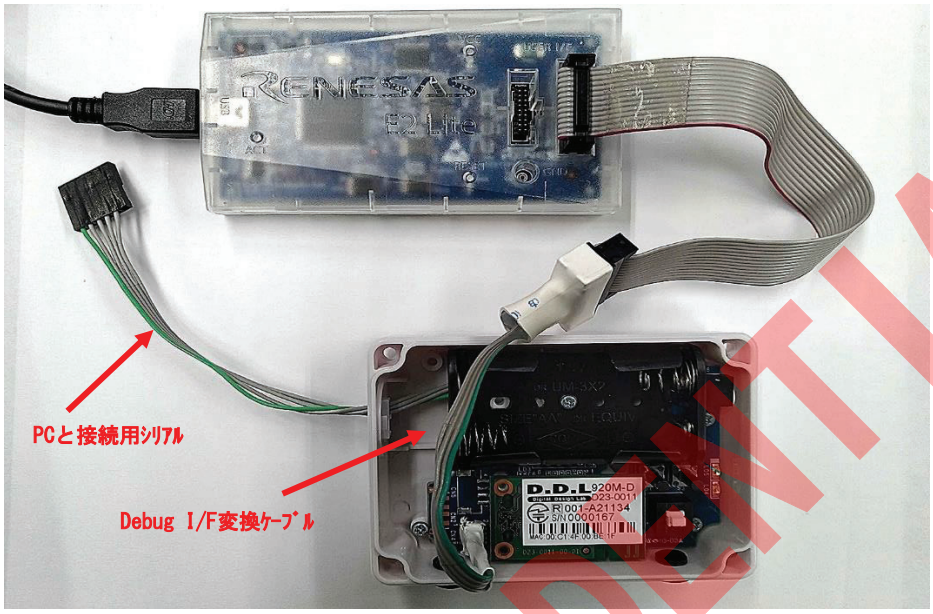


図6-2 無線モジュール評価機デバッグ構成

図6-3 無線モジュールUSBドングルデバッグ構成

・E1及びE2等のデバッガーをPCと接続する。
・評価機のPC接続用ｼﾘｱﾙに、USBシリアルケーブル（FTDI社製 TTL-232R-3V3）を使用しPCと接続します。
・ドングルはPCに直接接続します。
・このドキュメントの第 6 章の説明に従って、「Wi_SUN_FAN_RX_FW」e²studio プロジェクトを開きます。

※BR及び、RN,LFNを使用する時には、プロジェクトのビルド構成は下記に合わせてください。

BR：FFN-BR(FAN1.1 Core + HT + LE,Border Router, PHY-CWX)
RN：FFN-RN(FAN1.1 Core + HT + LE,Router Node, PHY-CWX)
LFN:RL-LFN(FAN1.1 LE,Limited Function Node,RL78/G1H)

6.2 IAR開発セットアップ（RL78/G1H）

注意: 2024 年 7 月現在、Renesas Wi-SUN FAN プロトコル スタックを IAR ツール チェーンと共に使用することは非推奨です。以前のリリース パッケージの「RL-FFN-Core-RN」IAR プロジェクトは、このリリース パッケージでも引き続き使用できますが、今後のリリースでは予告なしに削除される可能性があります。すべてのお客様には、プロジェクトを e²studio と CC-RL ツール チェーンに移行することを強くお勧めします。e²studio を使用した RL78/G1H のプロジェクト設定については、セクション 6.1.2 で説明します。

第7章 Wi-SUN FAN評価アプリケーション

Wi-SUN FANの接続評価のためにルネサスが提供するWi-SUN-FAN UIツールがあります。
このツールを使用する事により、簡単にモジュールの接続状態を確認する事ができ、モジュールを使用した開発の参考になります。

使い方等の詳細は、“Renesus Wi-SUN-FANUI Tool” Quick Start Guideを参照ください。

第8章 Wi-SUN FANサンプルアプリケーション

8.1 RX65x、RX65N、RA、RL78/G1Hのサンプルアプリケーション

Wi-SUN FAN サンプル アプリケーションを使用すると、RX65x MCU ファミリ（MB-RA604S-02 や RTK0EE0013D など）または RA ファミリ（RTK0EE0014D など）に基づくサポートされている評価キットで実行することで、Wi-SUN FAN プロトコル スタックに慣れることができます。Wi-SUN FAN プロトコル スタックは、ソース コードまたはライブラリ バージョンとして提供されます。違いについては、次の章で説明します。ＤＤＬのモジュールもこのアプリケーションをベースの変更を加えています。

8.1.1 ビルド構成

Wi-SUN FAN サンプル アプリケーションは、FAN1.1 プロトコル スタックのさまざまな拡張ステージと、さまざまな RX65x ターゲット プラットフォームおよび評価ボードをサポートするさまざまなビルド構成を提供します。サポートされているビルド構成は、次の表に示されています。

ビルド構成	***説明***	***詳細***
FFN-BR	FAN1.1 コア + HT + LE、ボーダールータ、PHY-CWX	FAN1.1コア実装（以下を含む） ***高***スループット（ ***MDR***サポート）と ***低***エネルギー（LFNペアリング支援）機能、 ボーダールータの設定、 PHYドライバ: CWX 対象: RTK0EE0013D 評価ボード DDLモジュール D23-0011
FFN-RN	FAN1.1 コア + HT + LE、ルータ ノード、PHY-CWX	FAN1.1コア実装（以下を含む） ***高***スループット（ ***MDR***サポート）と ***低***エネルギー（LFNペアリング支援）機能、 ルータノード構成、 PHYドライバ: CWX 対象: RTK0EE0013D 評価ボード DDLモジュール D23-0011
FFNコアBR	FAN1.1コア、 ボーダールータ、PHY-TRG	FAN1.1コア実装、 ボーダールータの設定、 PHYドライバ: TRG 対象: MB-RA604S-02 評価ボード
FFNコアRN	FAN1.1コア、 ルータノード、PHY-TRG	FAN1.1コア実装、 ルータノードルータ構成、 PHYドライバ: TRG 対象: MB-RA604S-02 評価ボード
FFN-コア-LE-BR	FAN1.1 コア + LE、 ボーダールータ、PHY-TRG	FAN1.1コア実装（以下を含む） ***低***エネルギー（LFNペアリング支援）機能、 ボーダールータの設定、 PHYドライバ: TRG

		対象: MB-RA604S-02 評価ボード
FFN-コア-LE-RN	FAN1.1 コア + LE、 ルータノード、PHY-TRG	FAN1.1コア実装（以下を含む） *低*エネルギー（LFNペアリング支援）機能、 ルータノードルータ構成、 PHYドライバ: TRG 対象: MB-RA604S-02 評価ボード
FFN-コア-LE-BRG-ハイブリッド	FAN1.1 コア + LE、 ボーダールータ、PHY-TRG	FAN1.1コア実装（以下を含む） *低*エネルギー（LFNペアリング支援）機能、 ボーダールータの設定、 PHYドライバ: TRG 対象: YG-HYBRID-PLC-RF評価ボード
FFN-コア-LE-RNG-ハイブリッド	FAN1.1 コア + LE、 ルータノード、PHY-TRG	FAN1.1コア実装（以下を含む） *低*エネルギー（LFNペアリング支援）機能、 ルータノードルータ構成、 PHYドライバ: TRG 対象: YG-HYBRID-PLC-RF評価ボード
LFN	FAN1.1 LE、 限定機能ノード、 物理-TRG	FAN1.1*低エネルギー*実装、 限定機能ノード、 PHYドライバ: TRG 対象: MB-RA604S-02 評価ボード

表8-1 ビルド構成 RX65x

ビルド構成	*説明*	*詳細*
RL-FFN-Core-RN	FAN1.1 Core, Router Node, RL78/G1H	FAN1.1コア実装、 ルータノード構成、 対象: TK-RLG1H+SB2評価ボード
RL-LFN	FAN1.1 LE, Limited Function Node, RL78/G1H	FAN1.1低エネルギー実装、 限定機能ノード 対象: TK-RLG1H+SB2評価ボード (only CC-RL) DDLモジュール D15-0013

表8-2 ビルド構成 RK78/G1H

8.1.2 ライブラリバージョン

アプリケーション層はソース コードとして提供され、ライブラリとして提供されるネットワーク層とそれ以下の層へのアクセスを提供します。PHY 層は例外で、これもソース コードとして提供されます。アプリケーションには 2 つの異なるライブラリ バージョンが含まれており、1 つはボーダールータ機能を含み、もう 1 つはルータ ノード構成のみをカバーします。

Border Router ライブラリは Router Node 機能もサポートしていることに注意してください。

e²studio IDE ベースの Wi-SUN FAN アプリケーションの概要を下図に示します。

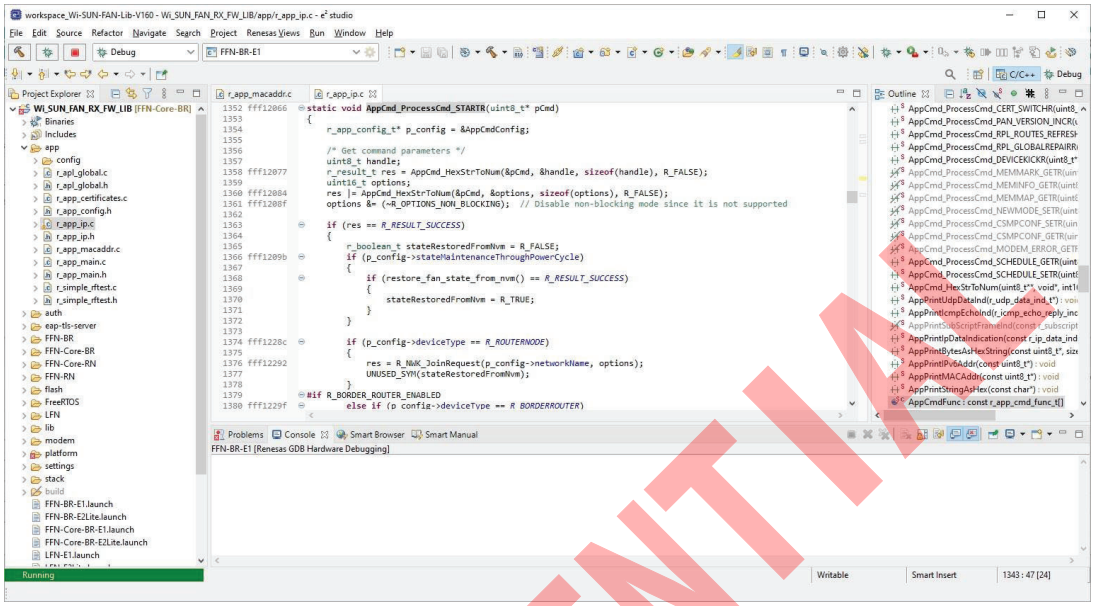


図8-1 e²studio Wi-SUN FANサンプルアプリケーションの概要

Wi-SUN FAN アプリケーションのパーティション分割を次の表に示します。

Wi_SUN_FAN_RX_FW(_LIB)	説明
├app	アプリケーション層ソース
├auth	認証ソース（mbedTLS、WPA）
├flash	FLASHソース
├ eap-tls-server	認証サーバーソース（認証子と AAA サービス）
├FreeRTOS	無料のRTOSソース
├Kib	ライブラリフォルダ
├loggen（注1）	生成されたRLogヘッダーファイルとRLogデコーダーが含まれています
├logging（注1）	RLog フレームワーク
├modem	モデムソース（HDLCエンコーダー/デコーダー）
├platform	プラットフォーム フォルダー
│ └ mb_rx604s_02_rx651	MB-RX604S-02 および RTK0EE0013D ハードウェアおよびデバイス構成
│ ├ ra	RTK0EE0014D11001BJ ハードウェアとデバイス構成（RA6M5）

│　└ TK_RLG1H_SB2	TK-RLG1H+SB2 ハードウェアとデバイス構成 (RL78/G1H)
├project	e²studio プロジェクト ファイル
└stack	ネットワーク、MAC、PHY レイヤー
├freertos_wrapper	FreeRTOS ラッパー
├inc	ファイルフォルダを含める
├lib	ライブラリフォルダ
│　├ RX-R-DHCPv6-Server-Lib.lib	DHCP サーバー ライブラリ
│　├ RX-FFN-BR-Lib.lib │　├ RX-FFN-RN-Lib.lib	FAN1.1 コア + HT + LE、 ボーダールータおよびルータノードライブラリ 、 PHY-CWX (RTK0EE0013Dボード)
│　├ RX-FFN-Core-BR-Lib.lib │　├ RX-FFN-Core-RN-Lib.lib	FAN1.1コア、 ボーダールータおよびルータノードライブラリ 、 PHY-TRG (MB-RA604S-02ボード)
│　├ RX-FFN-Core-LE-BR-Lib.lib │　├ RX-FFN-Core-LE-RN-Lib.lib	FAN1.1 コア + LE、 ボーダールータおよびルータノードライブラリ 、 PHY-TRG (MB-RA604S-02ボード)
│　└ RX-LFN-Lib.lib │	FAN1.1 LE 限定機能ノード、 LFNライブラリ、 PHY-TRG (MB-RA604S-02ボード)
└phy	PHY 層
├ raa604s00	PHY-TRG ドライバ (MB-RA604S-02 ボード)
├ cwx_rx	PHY-CWXドライバ (RTK0EE0013Dおよび RTK0EE0018D11001BJボード)
├ cwx_m_ra	PHY-CWX ドライバ (RTK0EE0014D11001BJ ボード)
└ raa604s00_rl78	RL78/G1Hのルネサス PHYドライバ

(注1) ソースコード版のみ。

表8-3 Wi-SUN FANアプリケーションパーティショニングRX65x

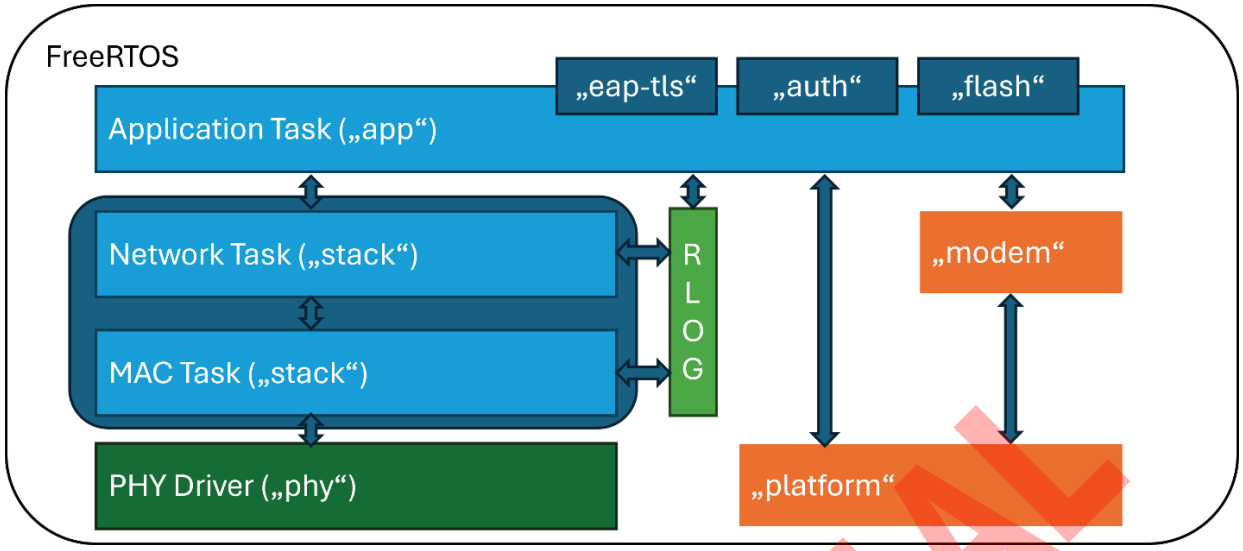


図8-2 Wi-SUN FANプロジェクトパーティショニング

異なるビルド構成を選択して有効にするには、プロジェクトの「プロパティ」メニューに移動して、「ALT + Enter」または「プロジェクト」メニューの「プロパティ」タブを選択します。対応するビルド構成をアクティブに設定してください。

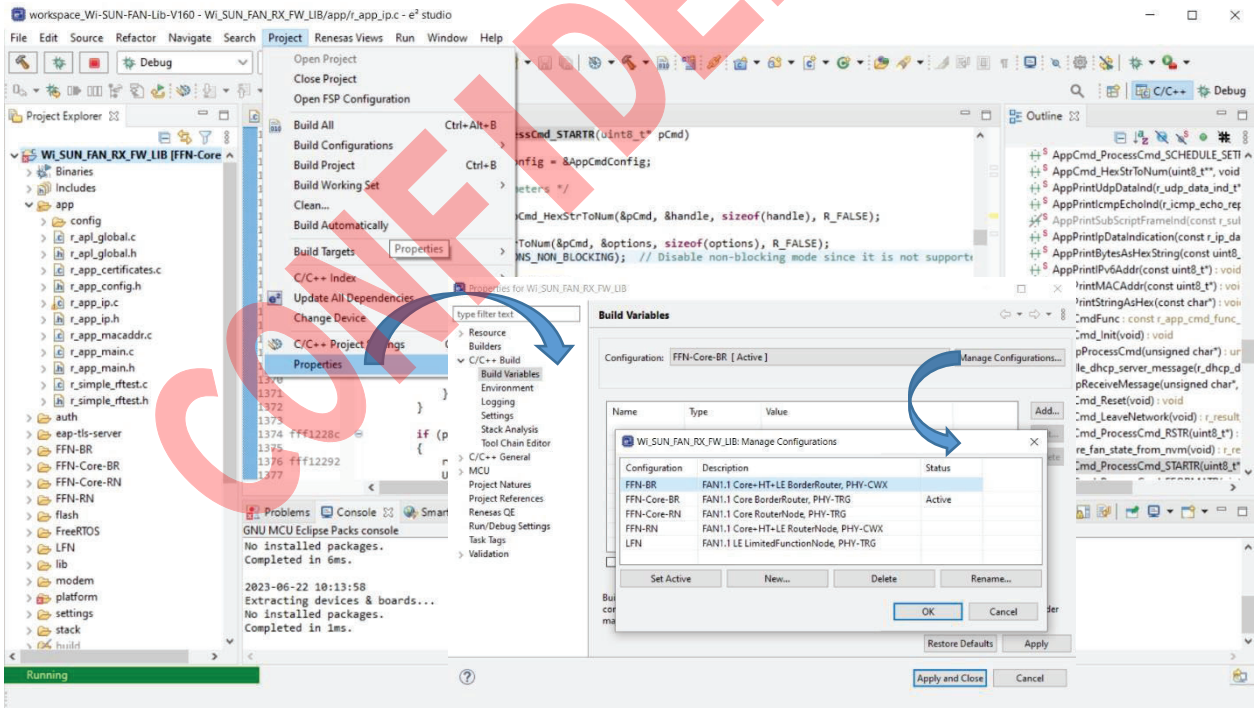


図8-3 Wi-SUN FANアプリケーションのビルド構成

「BorderRouter」(BR) ビルド構成で生成された実行可能ファイルは、ルーター ノード (RN) デバイスと同様に機能することに注意してください。

E1 または E2Lite オンチップ デバッグ エミュレーターを使用してデバッグ セッションを開始する前に、アクティブなビルド構成に応じて適切なアプリケーションが選択されていることを確認してください。

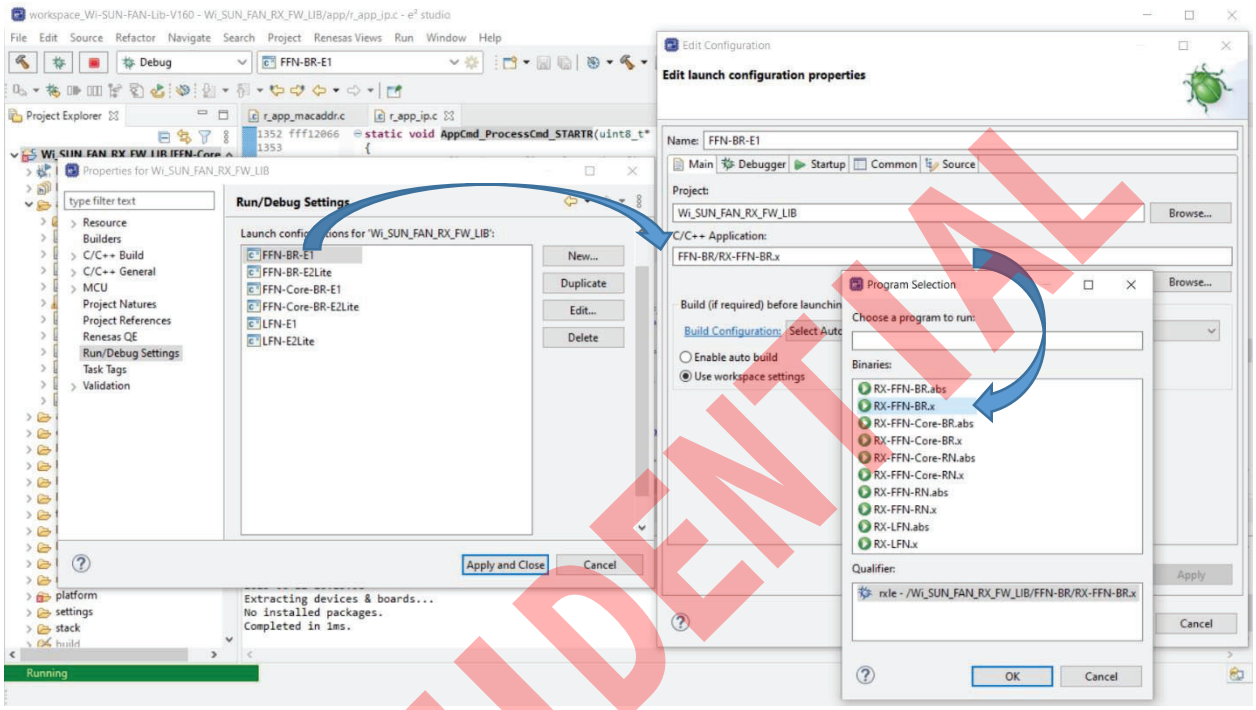


図8-4 Wi-SUN FANアプリケーションのデバッグ構成

8.1.2.1 RAMメモリマッピング

デフォルトでは、Wi-SUN FAN アプリケーションは MB-RA604S-02 評価ボードの外部 1 MByte SRAM を使用して、最大 1000 ルータ ノードの規模の大規模ネットワークをサポートします。これを実現するために、外部 SRAM は RX651 デバイスの外部アドレス空間 CS3、アドレス領域 0x05000000 - 0x05FFFFFF にマッピングされます。セクション「expRAM」と「expRAM_1」は、たとえば HEAP メモリ、バッファ、EAP-TLS サプリカント テーブル、ネイバー テーブルを外部 SRAM にマッピングするために使用されます。

内部メモリのみを使用する場合、つまり、384 kByte 拡張 RAM をサポートする RX65x グループ デバイスをベースにした RTK0EE0013D 評価ボードを使用する場合は、セクション「expRAM」のアドレスを 0x05000000 から 0x00800000（拡張 RAM の開始アドレス）に変更してください。ソース ルーティング テーブルとネイバー テーブルのサイズを内部 RAM に収まるように調整する必要がある場合があることに注意してください。

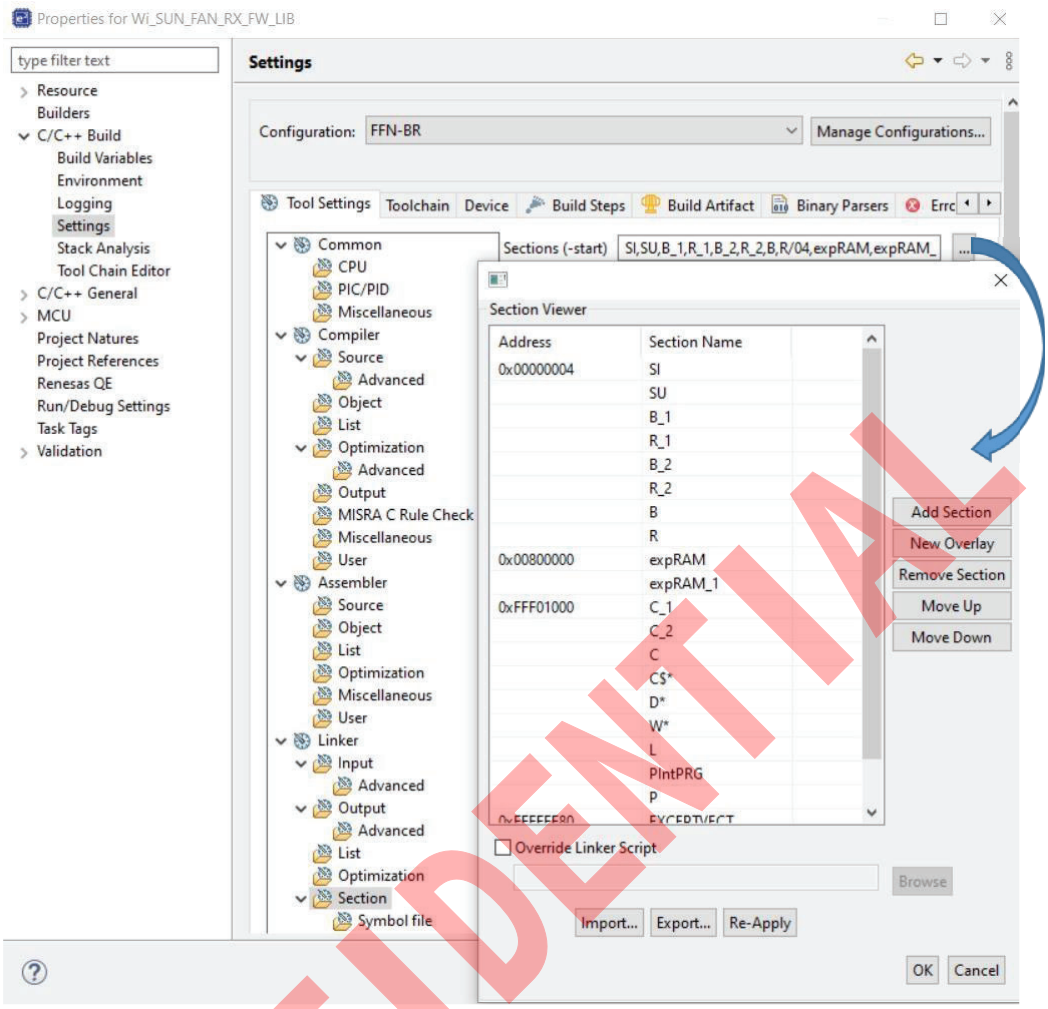


図8-5 RAMメモリマッピング

8.1.2.2 DHCPv6 サーバー

Wi-SUN FAN サンプル アプリケーションには、Wi-SUN FAN プロトコル スタックに組み込まれた DHCPv6 サーバー機能が含まれています。Linux などの標準サーバー ソリューションを利用したいユーザー向けに構成を柔軟にするため、DHCPv6 サーバーはアプリケーション層の一部になっています。対応するメッセージ送信は Wi-SUN FAN プロトコル スタックによって提供され、アプリケーション層とネットワーク層間のメッセージングが可能になります。Wi-SUN FAN プロトコル スタックのライブラリ バージョンでは、DHCPv6 サーバーは専用のライブラリにカプセル化されています。

統合 DHCPv6 サーバーでは、次の API 関数とメッセージがサポートされ、提供されます。

DHCPv6 サーバー:

API 関数:

R_DHCPV6_ServerInit　DHCPv6サーバーを初期化する

R_DHCPV6_ServerStop DHCPv6サーバーを停止する
R_DHCPV6_ServerProcessMsg DHCPv6サーバのプロセスメッセージ

メッセージ:
R_NWK_API_MSG_DHCP_IND DHCPv6サーバへの受信メッセージ

シリアル API 関数の詳細については、このドキュメントの第 10 章を参照してください。ユーザーは、この DHCPv6 サーバー ライブラリを簡単に削除し、独自の推奨実装に置き換えることができます。ただし、それに応じてシリアル API を調整または拡張する必要がある場合があります。

8.1.2.2.1 外部DHCPv6サーバの使用

外部 DHCPv6 サーバの使用は、ボーダールータで実行されている統合 DHCPv6 サーバを無効にすることで有効にできます。これを実現するには、コンパイル時オプション*R_DHCPV6_SERVER_ENABLED*を 0 に設定するか、未定義のままにする必要があります。これにより、ボーダールータ上のすべての DHCPv6 サーバ コードが無効になり、代わりに必要なリレー エージェント機能が有効になり、Wi-SUN ネットワークから外部 DHCPv6 サーバに DHCPv6 パケットを転送したり、その逆を行ったりできます。ボーダールータは、すべての着信 DHCPv6 クライアント メッセージ（クライアントがボーダールータに直接接続されている場合の Solicit メッセージ、またはクライアントがボーダールータから 1 ホップ以上離れている場合の Relay-Forward メッセージ）を DHCPv6 リレー メッセージでカプセル化し、外部 DHCPv6 サーバに転送します。DHCPv6 サーバからの応答の場合、ボーダールータは外部リレーを取り除き、その内容を元の送信者 （クライアント自体、または Solicit メッセージにリレー ヘッダーを追加したその親ノード）に転送します。

外部 DHCPv6 サーバーを使用するには、ボーダールーター ゲートウェイ（セクション 7.1.6 を参照）が起動して実行されている必要があり、ボーダールーター アプリケーションには外部 DHCPv6 サーバーの GUA/ULA が提供されていなければなりません。後者は、Demonstrator API で*DHCPV6SERVER_SETR*コマンド（14.1.12 を参照）を使用するか、*r_app_ip.cモジュール*内の*AppCmdConfig構造体*の*dhcpServerAddr フィールドを変更することによって構成できます。*デフォルトのアドレスは*fd00:cafe:cafe::1です。*これは、統合ボーダールーター ゲートウェイ ソリューションの一部である DHCPv6 サーバー（ *dnsmasq* ）のデフォルトのアドレスであるためです。DHCPv6 サーバー パラメータのこの 1 回限りの構成以外、提供されているボーダールーター サンプル アプリケーションは、それ以上の構成なしで動作するはずです。実装は、Linux で実行されている*dnsmasq* DHCPv6 サーバーでテストされています。

さらに、ボーダールータの FAN ネットワーク インターフェイスのデフォルトの IPv6 アドレスは 2017::1 に変更されます。これは、ボーダールータの統合 DHCPv6 サーバーのデフォルトの実装のように、ハードウェア アドレスと完全に一致する IPv6 アドレスをクライアントに割り当てる一般的な DHCPv6 サーバー実装がないためです。通常は、代わりにハードウェア アドレスからハッシュが作成され、IPv6 アドレスがアドレス空間に疑似ランダムに分散されます。ボーダールータが FAN インターフェイスに別の IPv6 アドレスを使用する場合は、 *IPADDR_SETRコマンド（14.1.10 を参照）を使用して Demonstrator API 経由で設定するか、 r_app_ip.cモジュール*内の*AppCmdConfig構造体*の*fanIpAddrフィールド*を変更する必要があります。

8.1.2.3 認証サーバー

Wi-SUN FAN 認証プロセスは、ボーダールータおよびルータ ノードで WiSUN 認証メッセージ（EAPOL にカプセル化）を処理する別のモジュールとして実装されます。デフォルトでは、Wi-SUN FAN サンプル アプリケーションには、Wi-SUN FAN プロトコル スタックで使用される独自の認証サーバー機能が含まれています。RADIUS などの標準サーバー ソリューションを利用したいユーザー向けに構成を柔軟にするため、認証サーバーはアプリケーション層の一部になっています。対応するメッセージ送信は、アプリケーション層とネットワーク層間のメッセージングを可能にするために、Wi-SUN FAN プロトコル スタ

ックによって提供されます。認証サーバーは、`auth` フォルダ内にあるサンプル アプリケーションの一部としてソース コードとして提供されます。

統合認証サーバーでは、次の API 関数とメッセージがサポートされ、提供されます。

認証サーバー:

API 関数:

R_AUTH_BR_Reset	EAP-TLSモジュールをリセットして初期化する
R_AUTH_BR_Start	EAP-TLSモジュールを起動します
R_AUTH_BR_ProcessDirectMessage	受信したEAPOLダイレクトメッセージを処理する
R_AUTH_BR_ProcessRelayMessage	受信EAPOLリレーメッセージを処理する
R_AUTH_BR_PeriodicUpdate	GTKの定期的な更新を実行する

メッセージ:

R_NWK_API_MSG_AUTH_BR_START_IND	EAP-TLSモジュールを開始する指示
R_NWK_API_MSG_EAPOL_DATA_IND	サプリカントからの EAP-TLS サーバーへの受信 EAPOL ダイレクト メッセージ
R_NWK_API_MSG_UDP_DATA_IND	サプリカントからの EAP-TLS サーバへの受信 EAPOL リレー メッセージ
R_NWK_API_MSG_AUTH_BR_PER_UPDATE	GTKの定期的な更新を実行します

API 関数の詳細については、このドキュメントの第 10 章を参照してください。ユーザーはこれらのソースを簡単に削除し、独自の推奨実装に置き換えることができます。RADIUS などの外部認証サーバーを使用する場合は、ユーザーがシリアル API 対話の一部を調整/拡張する必要があることに注意してください。プロセスに関する追加のガイダンスは、第 7.1.2.3.2 章に記載されています。

8.1.2.3.1 追加のIdevID検証

Wi-SUN FAN TPS は、*Wi-SUN IdevID* の構築方法とフォーマット方法に関するさまざまな要件を定義します。これらの要件のほとんどは、たとえば 99991232235959Z （一般化時刻） の明示的な有効期限など、認証モジュールによって内部的に検証されます。認証サーバーはソース コードとして提供されるため、ユーザーはアプリケーションの必要に応じて追加のチェックを追加/変更できます。対応するソース ファイル*x509.c*は *mbedTLS* ライブラリの一部であり、 *auth/mbedtls/library*にあります。

しかし、Wi-SUN FAN TPSのセクション6.5.1.1では、

証明書には、id-on-hardwareModuleName [RFC4108] タイプの代替 OtherName を含む SubjectAlternativeName 拡張（SANE）が含まれている必要があり、SANE はクリティカルとしてマークされている必要があります。

*id-on-hardwareModuleName*の内容は製造元固有であり、SANE には他の名前も含まれる可能性があるため、サンプル アプリケーションは、証明書の検証中に認証モジュールによって呼び出される、SANE のカスタム検証用のコールバックを提供します。これにより、SANE 検証を完全にカスタマイズでき、その結果は戻り値を介して認証モジュールに返されます。コールバック関数は、プロジェクトの*appディレクトリ内のr_app_certificates.c*ソース ファイルにあります。

構文:

int R_APP_Certs_ValidateSubjectAltNameExtension(const uint8_t subject_alt_names,*
uint16_t names_length、int
ownCert);

引数:

(in) const uint8_t* subject_alt_names	ASN1 でエンコードされたサブジェクト別名拡張 (SANE) を含むバッファ。
(in) uint16_t names_length	入力バッファのサイズ (バイト単位)。
(in) int ownCert	渡された証明書が自分のものである場合は True。証明書が別のデバイスから送信された場合は False。

説明:

この関数は、認証モジュールによってデバイス証明書が検証されるたびに呼び出されます。
EAP-TLS 交換。デバイス証明書の*SubjectAlternativeName 拡張 (SANE)を検証するための顧客固有のコードが含まれています。この関数の戻り値は、呼び出し元の認証モジュールに検証結果を通知します。*

制限事項:

ボーダールータでは、この関数は他のデバイスからの受信証明書に対してのみ呼び出されます。ルータ ノードでは、この関数は自身の証明書に対してのみ呼び出されます。
この関数はデバイス証明書に対してのみ呼び出されます。つまり、CA 証明書に対しては呼び出されません。

戻り値:

0	SANE 検証が成功しました。認証モジュールは、残りの (内部) チェックが成功する限り、対応する証明書を有効と見なします。
!= 0	SANE 検証に失敗しました。認証モジュールは、いずれの場合でも対応する証明書を拒否します。

8.1.2.3.2 EAP-TLS の外部認証サーバー (RADIUS) の使用

Wi-SUN FAN 認証プロセスの第 1 フェーズでは、認証者 (ボーダールータ) が各 FAN ノード (*サプリカントとも呼ばれる*) と EAP-TLS ハンドシェイクを実行して、その ID を検証し、共通のペアワイズ マスター キー (PMK) を導出する必要があります。PMK は、以降の認証フェーズで必要になります。デフォルトでは、EAP-TLS サーバはボーダールータ サンプル アプリケーションに統合されているため、ボーダールータ自体が各サプリカントと EAP-TLS ハンドシェイクを実行します。EAP-TLS サーバは、 *eap-tls-server フォルダ内に別のモジュールとして実装され、ボーダールータ認証モジュールr_auth_br.c (セクション 7.1.2.3 を参照)*と連携して動作します。または、Wi-SUN FAN TPS を使用すると、EAP-TLS サーバ機能を RADIUS サーバなどの外部認証サーバに委任できます。これは、ボーダールータのリソース消費を削減したり (EAP-TLS サーバはサプリカントごとに大量の RAM を必要とします)、認証のパフォーマンスを向上させたりするのに役立つ場合があります。

外部認証サーバーの使用は、ボーダールーターで実行されている統合 EAP-TLS サーバーを無効にすることで有効にできます。これを実現するには、コンパイル時オプション*R_EAP_TLS_SERVER_ENABLED*を 0 に設定するか、未定義のままにする必要があります。これにより、ボーダールーターのすべての EAP-TLS サーバー コードが無効になり、代わりに RADIUS 転送モジュール（ *r_auth_br_radius.c* ）が有効になります。*このモジュールは、Wi-SUN FAN TPS で指定されているように EAP-TLS 交換を実行するために必要な RADIUS プロトコルの部分を実装し、 FreeRADIUSサーバー*でテストされています。RADIUS モジュールは、サプリカントから最初の「キー要求」メッセージを受信した後、それぞれのクライアントにコンテキスト構造を割り当て、最初の EAP メッセージ（「要求、アイデンティティ」）をサプリカントに返します。その後、ボーダールーターは、このサプリカントからのすべての着信 EAP メッセージを RADIUS アクセス要求メッセージにカプセル化し、RADIUS サーバーに転送します。同様に、ボーダールータは RADIUS サーバ メッセージから EAP メッセージを抽出し、それぞれのサプリカントに返送します。サプリカントの観点からは、EAP-TLS 交換がボーダールータ自体で実行されるか、外部認証サーバで実行されるかに違いはありません。ボーダールータは、EAP-TLS 交換の最終メッセージを受信すると、メッセージから PMK を抽出し、ボーダールータ認証モジュールに渡します。これにより、PMK は Wi-SUN 認証プロセスの後続フェーズで使用できるようになります。

外部認証サーバーを使用するには、ボーダールーター ゲートウェイ（セクション 7.1.6 を参照）が起動して実行されている必要があり、ボーダールーター アプリケーションに外部認証サーバーの GUA/ULA を提供する必要があります。後者は、AUTH *SERVER_SETR*コマンド（14.1.11 を参照）を使用して Demonstrator API 経由で構成するか、 *r_app_ip.cモジュール*内の*AppCmdConfig構造体のauthServerAddrフィールドを変更することで構成できます*。デフォルトのアドレスは*fd00:cafe:cafe::1です*。これは、統合ボーダールーター ゲートウェイ ソリューションの一部である RADIUS サーバーのデフォルトのアドレスであるためです。さらに、RADIUS モジュールは*R_EAP_TLS_Radius_Init*関数経由で適切に初期化する必要があり、構成パラメーターは使用中の RADIUS サーバーのものと一致する必要があります。たとえば、RADIUS サーバーの UDP ポートと、ボーダールーターと RADIUS サーバー間の RADIUS メッセージの暗号化と認証に使用される共有シークレットは、同じである必要があります。この 1 回限りの構成を除き、ボーダールーターのサンプル アプリケーションと認証モジュールは、EAP-TLS 交換を実行するために RADIUS モジュールと必要なすべてのやり取りをすでに提供しているため、RADIUS モジュールはそれ以上の調整なしで動作するはずです。

RADIUS 転送モジュールと Wi-SUN ボーダールータ認証モジュールの相互作用の詳細については、それぞれの実装ファイル（ *auth/r_auth_br_radius.c*および*auth/r_auth_br.c* ）を参照してください。

8.1.3 RX65x、RX65N、RX671 デバイスの BSP およびペリフェラル ドライバ

Wi-SUN プロトコル スタックで使用されるボード サポート パッケージ（BSP）と周辺機器ドライバは、それぞれのファームウェア統合テクノロジ（FIT）コンポーネントに基づいています。次の表は、プロトコル スタックがテストされた、使用されているコンポーネントと各コンポーネントのバージョンをまとめたものです。

FIT Component	説明	Version
r_bsp	Board Support Package	7.42
Config_Ext Bus	Buses (Code Generator component) (Only for TRG Boards)	1.11.0
r_byteq	Byte-based circular buffer library	2.10
r_cmtx_rx	CMTW Timer driver	2.80

r_dmaca_rx	DMAC driver	3.20
r_flash_rx	Flash API	5.11
r_datfrx_rx	Flash Memory Data Manager	2.10
r_gpio_rx	GPIO driver	5.00
Config_TPU3	Normal Mode Timer (Code Generator component)	1.12.0
r_rtc_rx	Real-Time Clock driver	2.90
r_sci_rx	SCI driver	5.00
r_sci_iic_rx	Simple IIC driver	2.71

表8-4 Wi-SUNプロトコルスタックで使用されるFITコンポーネント

上記のコンポーネントをプロジェクトに追加するには、 Smart Configuratorツールを使用します。

Smart Configurator は、Renesas マイクロコントローラで利用可能な周辺機器の構成を開発者が容易に行えるようにするユーティリティです。このツールは、GPIO、通信バス、タイマー、メモリ管理サブシステムなどのさまざまな周辺機器が提供する機能のセットアップと統合を簡素化し、高速化します。

Smart Configurator は、"/project/RX6xx/e2studio" ディレクトリに配置されている "scfg" ファイル拡張子を持つ構成ファイルに依存し ます。構成ファイルに目的のコンポーネントを追加すると、Smart Configurator ツールは、使用される各コンポーネントのソース コードを自動的に生成します。コード生成は、ユーザーが手動でトリガーすることも、プロジェクトのビルド時に自動的にトリガーすることもできます。ただし、 Wi-SUN プロトコル スタックは、生成されたソース コードのいくつかのカスタマイズに依存しているため、各プロジェクト ビルド前の自動コード生成は無効になっています。したがって、ユーザーがプロジェクトの FIT コンポーネントを追加または変更する必要がある場合は、プロジェクトをビルドする前に、Smart Configurator ツールによるコード生成を手動でトリガーする必要があります（図 8-6 を参照）。

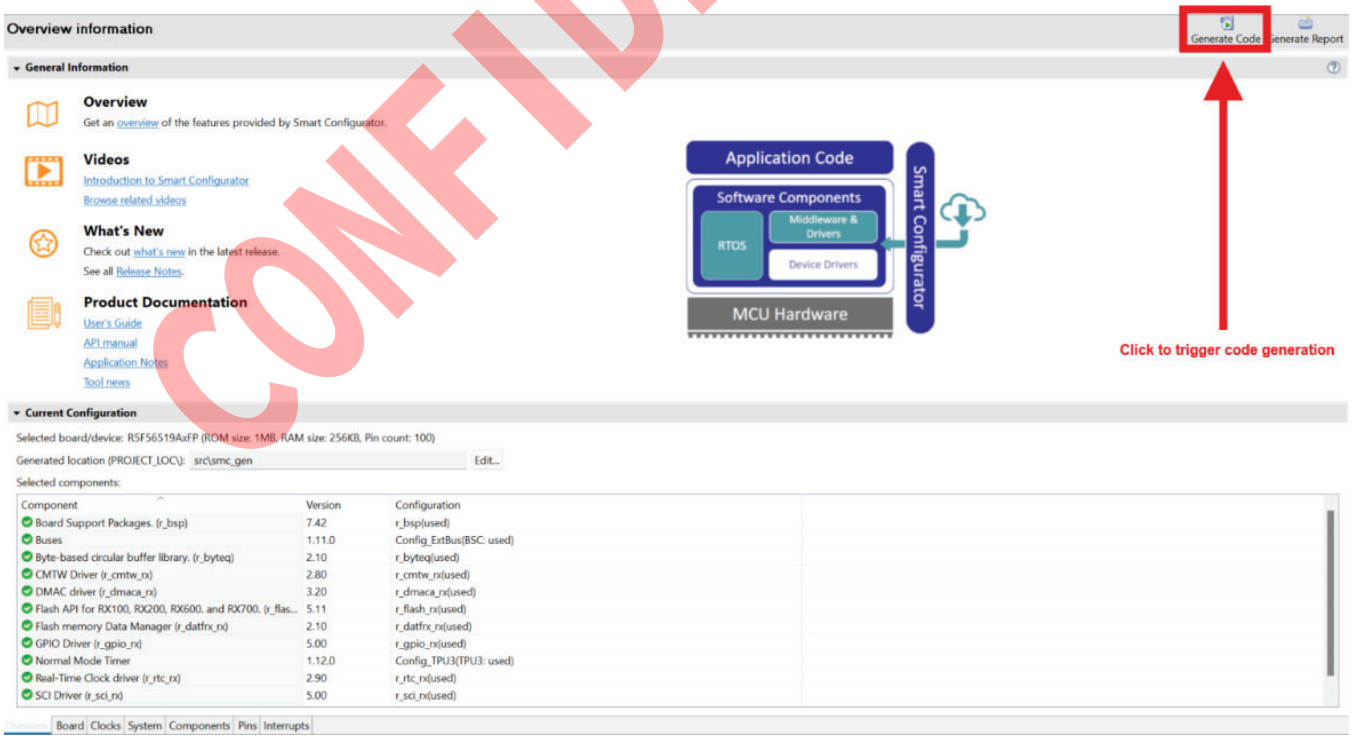


図8-6 スマートコンフィギュレータを使用したコード生成のトリガー

Smart Configurator で手動でコード生成をトリガーしたり、FIT コンポーネントを変更したりする場合は、Wi-SUN プロトコル スタックで使用される生成

されたソース コードのカスタマイズを省略しないように注意する必要があります。FIT コンポーネントの変更を容易にするために、「R_USE_RENESAS_STACK」マクロは、Wi-SUN プロトコル スタックによって自動生成されたコードに加えられたカスタマイズを保護します。

したがって、ユーザーは、差分ツールやバージョン管理システムなどの適切な方法を使用して、変更を元のソースコードと比較し、特定のターゲットプラットフォームとユースケースに応じてカスタマイズされた部分の省略を元に戻す必要があります。

8.1.4 ハードウェアリソース

RX65x デバイスの次のハードウェア リソースは、Wi-SUN FAN プロトコル スタックとサンプル アプリケーションによって使用されます。

ソフトウェアモジュール	*RX65x 周辺*	*RX671*	*RA6M5*	*RL78/G1H*	*説明*
サブGHz用RFドライバー	SCI0 0(TRG) SPI2(CWX)	SCI11	SPI1	CSI20	RFドライバ制御
	DMA (0,1)	DMAC(0,1)	DMAC(0,1)	DTC(0,1)	RFドライバ制御
	TPU2	TOU2	GPT0	INTTM00	RFドライバ制御
	TPU4(TRG) CMTW0(CWX)	CMTW0	GPT1	INTTM02	総送信時間（デューティ比）の取り扱い
Wi-SUN FAN プロトコル スタック	TPU3	TPU3	SYSTMR	INTIT	システム ティック タイマー（周波数ホッピングと FreeRTOS システム タイマーの処理）
	-	-	-	INTRTC	周波数ホッピング タイミングの調整
	TPU2	TPU2	GPT0	INTTM02	RF ドライバーと Wi-SUN FAN スタック間のタスク管理の処理。 (TPU2 は RF ドライバー制御と共有されます。)
シリアルインター	SC6	SCI3	SCI5	UART1(SAU0)	上位層へのシリアル通信

フェース	SCI12	SCI12			外部E²PROMへのI²C通信 (外部 E²PROM によるフレーム カウンターの保存の場合にのみ使用されます)。

表8-5 ハードウェアリソースRX65x

さらに、Wi-SUN FAN プロトコル スタックは、評価ボードでサポートされている LED を使用します。LED は次の情報を示すために使用されます。

MB-RA604S-02 & TK-RLG1H+SB2

- LED0 : ブロードキャストDWELL間隔アクティブ時間
- LED1 : ユニキャストDWELL間隔アクティブ時間
- LED2: TXフレーム表示

LED1は周波数ホッピングが有効な場合にのみ駆動されます。LEDを別の目的で使用する必要が生じた場合は、 RdrvLEDx_ONを変更することで対応するポートへのアクセスを削除できます。

それぞれRdrvLEDx_OFF関数であり、ソース コード ファイル「hardware.c」として入手できます。

Wi-SUN FAN プロトコル スタックは、シリアル インターフェイス SCI6 に対して 500kbps と 1500kbps の 2 つのボー レートをサポートしています。500kbps はすべての TRG ベースのソリューションのデフォルト設定であり、1500kbps はすべての CWX ベースのソリューションで使用されます。ボー レートをカスタマイズするには、関数「RdrvUART_Initialize」(ファイル hardware.c) の入力パラメータをそれに応じて変更してください。

8.1.5 メモリリソース

さまざまなビルド構成の標準的なメモリ リソースと必要なメモリ リソースを以下の表に示します。

Wi-SUN FAN スタック ライブラリ自体のメモリ リソースは固定されていますが、さまざまなテーブルの RAM リソース要件はプロジェクト設定に大きく依存することに注意してください。

RX651/TRG system configuration

		FFN-Core-LE-RN (TESSERA board)	FFN-Core-LE-RN (Customer design)		FFN-Core-LE-BR (TESSERA board)	FFN-Core-LE-BR (Customer Design)
ROM		335.3kB	335.3kB		397.7kB	397.7kB
RAM	Source routing Table	–	–		43B x 512 = 21.5kB	43B x 512 = 21.5kB
	MAC Neighbor Cache RPL Neighbor Cache ARO Neighbor Cache	75B x 69= 5.1kB 45B x 16 = 0.7kB 36B x 48 = 1.7kB	75B x 69 = 5.1kB 45B x 16 = 0.7kB 36B x 48 = 1.7kB		75B x 221 = 16.2kB 45B x 16 = 0.7kB 36B x 200 = 7.0kB	75B x 221 = 16.2kB 45B x 16 = 0.7kB 36B x 200 = 7.0kB
	Authentication Server	–	–		131B x 512 = 65.5kB *RSN context information	131B x 512 = 65.5kB *RSN context information
	Parallel EAP-TLS Handshakes (default value 4)	–	–		14840B x 4 = 58kB	14840B x 4 = 58kB
	Logging Buffer	16kB	16kB		16kB	16kB
RAM Total		113.2kB	113.2kB		329.3kB	329.3kB
EEPROM	Receiver frame counter	–	0kB (RAM storage only) - 24 x 256 = 6.1kB (RAM)		–	0kB (RAM storage only) - 24 x 1000 = 24kB (RAM)
	Transmitter frame counter	–	0.1kB		–	0.1kB
	Certificates (LDevID, optional)	–	0kB - 5kB		–	–
EEPROM Total		–	0.1kB – 5.1kB		–	0.1kB
H/W	RF / MCU	*RAA604S00 *R5F56519ADFP Code Flash : 1MB RAM : 256kB Data Flash : None *External SRAM: 1MB	*RAA604S00 *R5F56519ADFP Code Flash : 1MB RAM : 256kB Data Flash : None	*RAA604S00 *R5F5651EDDFP Code Flash : 2MB RAM : 640kB Data Flash : 32kB	*RAA604S00 *R5F56519ADFP Code Flash : 1MB RAM : 256kB Data Flash : None *External SRAM: 1MB	*RAA604S00 *R5F5651EDDFP Code Flash : 2MB RAM : 640kB Data Flash : 32KB
	External X'tal (min. 20ppm)	16MHz	16MHz	16MHz	16MHz	16MHz
	External EEPROM	–	*Type name : M24C64-DFMC6TG (8KB)	–	–	–
	External SRAM	–	–	–	MLV0816BGSB-4S2 (1MB)	–

RX651/CWX & RA/CWX system configuration

		FFN-RN (RTK0EE0013D)	FFN- BR (RTK0EE0013D)	FFN-RN (RTK0EE0014D)	FFN- BR (RTK0EE0014D)
ROM		351.3kB	413.8kB	368.5kB	430.2kB
RAM	Source routing Table	–	43B x 512 = 21.5kB	–	43B x 512 = 21.5kB
	MAC Neighbor Cache RPL Neighbor Cache ARO Neighbor Cache	103B x 255 = 6.9kB 45B x 16 = 0.7kB 36B x 48 = 1.7kB	103B x 221 = 22.2kB 45B x 16 = 0.7kB 36B x 200 = 7.0kB	103B x 255 = 6.9kB 45B x 16 = 0.7kB 36B x 48 = 1.7kB	103B x 221 = 22.2kB 45B x 16 = 0.7kB 36B x 200 = 7.0kB
	Authentication Server	–	131B x 512 = 65.5kB *RSN context information	–	131B x 512 = 65.5kB *RSN context information
	Parallel EAP-TLS Handshakes (default value 8)	–	14840B x 4 = 58kB	–	14840B x 4 = 58kB
	Logging Buffer	16kB	16kB	16kB	16kB
RAM Total		134.8kB	348.8kB	135.7kB	352.4kB
EEPROM	Receiver frame counter	0kB (RAM storage only) - 24 x 256 = 6.1kB (RAM)	0kB (RAM storage only) - 24 x 1000 = 24kB (RAM)	0kB (RAM storage only) - 24 x 256 = 6.1kB (RAM)	0kB (RAM storage only) - 24 x 1000 = 24kB (RAM)
	Transmitter frame counter	0.1kB	0.1kB	0.1kB	0.1kB
	Certificates (LDevID, optional)	0kB - 5kB	–	0kB - 5kB	–
EEPROM Total		0.1kB – 5.1kB	0.1kB	0.1kB – 5.1kB	0.1kB
H/W	RF / MCU	*R9A06G062GNP *R5F565NEDDFP Code Flash : 2MB RAM : 640kB Data Flash : 32kB	*R9A06G062GNP *R5F565NEDDFP Code Flash : 2MB RAM : 640kB Data Flash : 32kB	*R9A06G062GNP *R7FA6M5AH3CFP Code Flash : 2MB RAM : 512kB Data Flash : 8kB	*R9A06G062GNP *R7FA6M5AH3CFP Code Flash : 2MB RAM : 512kB Data Flash : 8kB
	External X'tal (min. 20ppm)	16MHz	16MHz	16MHz	16MHz
	External EEPROM	–	–	–	–

RL78/G1H system configuration

		FFN Router Node Core (TESSERA board)	LFN (TESSERA board)
ROM Total		397.2kB	384.4kB
RAM	Source routing Table	—	—
	MAC Neighbor Cache RPL Neighbor Cache ARO Neighbor Cache	52B x 23 = 1.1kB 46B x 6 = 276B 32B x 12 = 384B	56B x 8 = 448B 46B x 2 = 92B 32B x 1 = 32B
	Logging Buffer	1kB	1kB
RAM Total		45.4kB	43.5kB
NVM	Receiver frame counter	0kB (RAM storage only)	0kB (RAM storage only)
	Transmitter frame counter	0.1kB	0.1kB
	Certificates (LDevID, optional)	0kB - 5kB	0kB - 5kB
NVM Total		0.1kB – 5.1kB	0.1kB – 5.1kB
H/W	RF / MCU	*RL78/G1H Code Flash : 512kB RAM : 48kB Data Flash : 8kB	*RL78/G1H Code Flash : 512kB RAM : 48kB Data Flash : 8kB
	External X'tal (min. 20ppm)	32.768kHz	32.768kHz
	External EEPROM	—	—

注: RL78/G1HのRAM制限により
デバイスの以下の機能は制限されています:
Maximum support of up to 18 Neighbors
Initiation of EDFE not supported Initiation of MPL transfers not supported
Limited amount of simultaneous MPL forwarding transfers
※DDLモジュールD15-0013もLFNと同様に上記制限となります。。

8.1.6 ゲートウェイ機能（ボーダールータ）

Wi-SUN FAN サンプル アプリケーションには、WiSUN FAN プロトコル スタックに組み込まれたゲートウェイ機能の例が含まれています。FAN ネットワーク外の宛先をターゲットとするすべての着信 IPv6 パケットは、ボーダールータによって転送されます。これは、アプリケーション層に IP データ指示を発行することによって行われます。

MB-RA604S-02（RX65x）評価ボードには LAN インターフェイスが含まれていないため、FAN ネットワーク外への転送は、例として第 14.1 章で説明されているデモンストレータ シリアル API を使用して行われます。

ゲートウェイ機能では、次のメッセージ タイプと機能がサポートされ、提供されます。

R_NWK_IP_DataRequest		IPデータリクエスト この関数は、APP 層から NWK 層への IPv6 データ要求を開始します。メッセージとして、生の IPv6 データ パケットが期待されます。このメッセージは、FAN ネットワークの外部から着信する IPv6 パケットを FAN ネットワークに転送するために使用されます。 このメッセージ タイプは、FAN ネットワーク経由で APP レイヤーから任意の生の IPv6 メッセージの送信を開始するためにも使用できます。 詳細については、10.1.12章も参照してください。
R_NWK_API_MSG_IP_DATA_IND		IPデータ表示 このメッセージ タイプは、FAN ネットワーク外部の宛先をターゲットとする着信 IPv6 パケットに対して、NWK 層から APP 層に送信されます。FAN ネットワーク外部への転送は、APP 層内で、シリアル API を介して生の IPv6 パケットを FAN ネットワーク外部の宛先に送信することによって行われます。 詳細については、第0章も参照してください。

詳細については、ソースコードとして提供されるアプリケーション層を参照してください。

外部ネットワークとの通信を可能にするために、ルネサスは、Raspberry Pi 3/4 を使用して Wi-SUN FAN ネットワークと外部 IPv6 ネットワーク間でパケットを転送するボーダールータ ゲートウェイと、ボーダールータ アプリケーションを実行する互換性のある評価ボードの統合ソリューションを提供しています。後者は、セクション 7.1.6 で説明されている IPv6 転送機能を使用します。統合ボーダールータ ゲートウェイの一般的なアプローチを図 8-7 に示します。詳細については、ボーダールータ ゲートウェイ配布のユーザー マニュアルを参照してください。

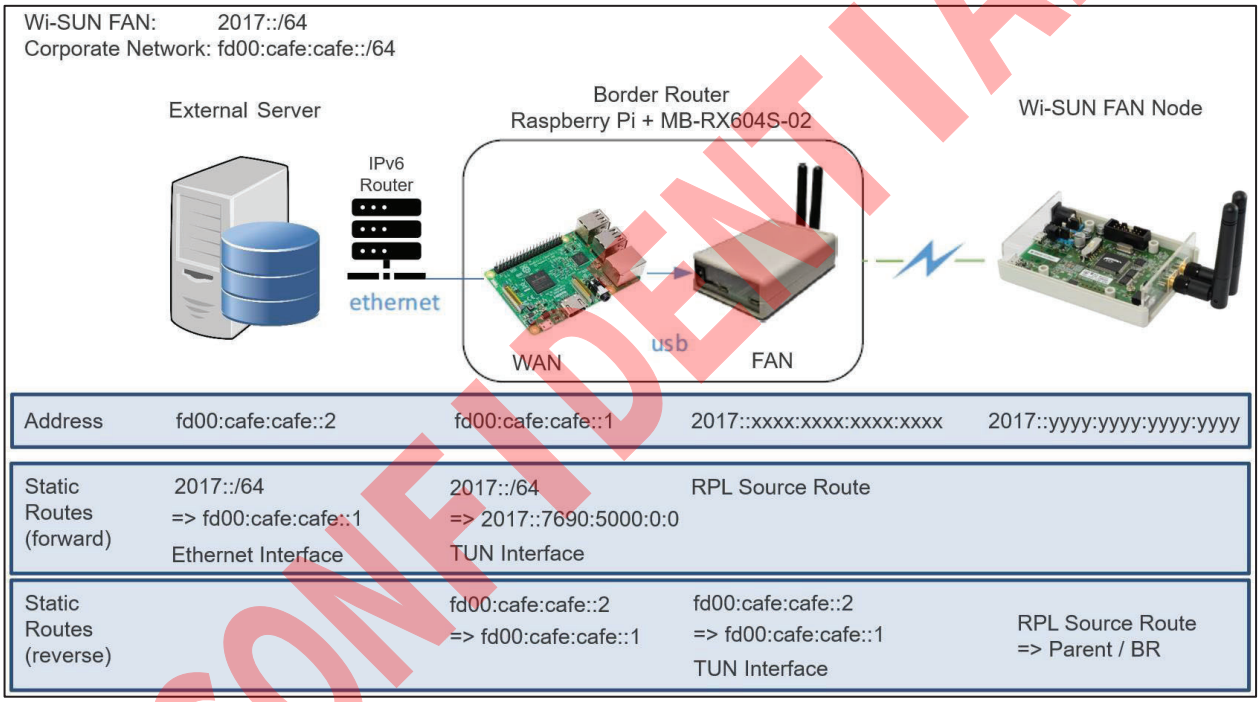


図8-7 Wi-SUN FANネットワークと外部サーバー間のIPv6接続

8.2 コンパイル時のオプション

Wi-SUN FAN プロトコル スタックでは、次のコンパイル時オプションがサポートされています。これらを使用して、スタックの構成をカスタマイズできます。たとえば、ボーダールータやルータ ノードの機能を有効にしたり、ネイバー キャッシュ テーブルのサイズを設定したり、事前定義されたセキュリティ キーを使用したりすることができます。

- R_BORDER_ROUTER_ENABLED=1この設定は、境界ルータ機能を有効にします。このオプションを省略すると、スタックはルータ ノード構成でのみ動作します。境界ルータ機能が有効な場合、スタックはルータ ノードとして動作できます。

- R_APP_AUTO_START=1このシンボルが定義されている場合、デバイスはセクション 8.2 で説明されているように Wi-SUN FANスタックを自動的に構成して起動します。デフォルトでは、この設定は無効になっています。

- R_APP_CFG_FREQ_HOP_ENABLED=0 1に設定すると、この設定により周波数ホッピングが有効になります。デフォルトのデバイス構成（セクション8.1を参照）。0に設定すると、デフォルトのデバイス構成は固定チャネル操作になります。

- R_APP_CFG_MDR_ENABLED=0 1に設定すると、この設定によりデフォルトのデバイスでMDRが有効になります構成（セクション 8.1 を参照）を設定し、適切な新しいモードのセットを構成します。0 に設定すると、デフォルトのデバイス構成で MDR が無効になります。

- R_CSMP_ENABLED=0 1 に設定すると、CoAP Simple Management Protocol (CSMP) のサポートが有効になり、このノードは 参加状態 5 に達すると、設定された NMS への登録を自動的に試行します。CSMP の設定と使用方法の詳細については、セクション 7.15 を参照してください。このオプションを 0 に設定すると、CSMP コードは無効になります。

- R_EAP_TLS_SERVER_ENABLED=1この設定により、統合 EAP-TLS サーバー機能が有効になります。このオプションは、境界ルーターの設定と組み合わせて使用します。このオプションを省略または無効にすると、統合 EAP-TLS サーバー機能が無効になり、RADIUS 転送モジュールに置き換えられ、代わりに外部 RADIUS サーバーを使用できるようになります（詳細については、セクション 7.1.2.3.2 を参照してください）。

- R_EAP_TLS_SRV_MAX_CONCURRENT_SUPPLICANTS=4この設定は、この BR と同時に EAP-TLS ハンドシェイクを実行できるサプリカントの最大数を定義します。統合 EAP-TLS サーバーを使用する場合、追加サプリカントごとに EAP-TLS ハンドシェイクを実行するために境界ルーター上に約 14.5 kB の RAM が必要です。外部 RADIUS サーバーを使用する場合、追加サプリカントごとに約 3.5 kB の RAM が必要です。後者の場合、 R_EAP_TLS_SRV_MAX_CONCURRENT_SUPPLICANTSのデフォルト値は 32 に設定されます。

- R_DHCPV6_SERVER_ENABLED=1この設定により、統合 DHCPv6 サーバー機能が有効になります。このオプションは、境界ルータ設定と組み合わせて使用します。このオプションを省略または無効にすると、統合 DHCPv6 サーバー機能が無効になり、DHCPv6 リレー エージェント機能に置き換えられ、代わりに外部 DHCPv6 が使用されるようになります（詳細については、セクション 7.1.2.2.1 を参照してください）。

- R_AUTH_MULTIPLE_CERTS=1この設定により、2 つの独立した認証証明書の使用が可能になります。このオプ ションは、EAP-TLS サーバー機能と組み合わせて

使用されます。証明書はr_app_certificates.c で定義されます。メイン証明書と代替証明書の切り替えは、変数r_auth_certs_use_alternate によって行われます。この変数には、 CERT_SWITCHRシリアル API コマンドからもアクセスできます。

•R_MAX_PAN_SIZE=512この設定は、境界ルータがサポートできる Wi-SUN FAN ネットワークの最大サイズを定義します。このオプションは、境界ルータの設定と組み合わせて使用します。境界ルータがサポートできるサプリカントとソースルートの最大数を定義します。

•R_MAX_PARENT_CANDIDATES=16|6この設定は、候補となる親を格納するRPL親ネイバーキャッシュの最大サイズを定義します（両方の親を格納するにはサイズが少なくとも2である必要があります）。
優先親と代替親）。可能性のある親から DIO メッセージを受信すると、キャッシュ エントリが登録されます。このオプションは、どの RN でも適用できます（BR の場合、BR は RPL ルート ノードであり、親を選択しないため、この設定は不要です）。デフォルト値は 16 です。RL78/G1H ルータ ノードの場合、リソースの制約により、値は 6 に削減されます。

•R_MAX_CHILD_NODES=200|48|16この設定は子ノードの最大サイズを定義します
登録された IP アドレスを含むネイバー キャッシュ。キャッシュ エントリは、別のノードからアドレス登録オプション（ARO）を含むネイバー要請メッセージを受信することによって登録されます。このオプションの変更は、BR および RN 構成でサポートされています。デフォルト値は、境界ルータの場合は 200、ルータ ノードの場合は 48、RL78/G1H で実行されているルータ ノードの場合は 12 です（リソースの制約のため）。

• R_LOG_BUFFER_SIZE=16384この設定は、RLog データを格納するために使用されるリング バッファのサイズを定義します。16 kB にすると、リング バッファが十分な大きさになり、ポーリング間隔中にログ ステートメントが上書きされることがなくなるため、デバッグ機能が確保されます。RAM を最適化するために、RLogステートメントの欠落/破損以外の機能的な影響なしに、この値を減らすことができます。RLog フレームワークについては、セクション 7.13 で説明します。

•R_MPL_SEED_ENABLED=1この設定により、低電力および低消費電力のマルチキャストプロトコルが有効になります。ロスの多いネットワーク（MPL）。このオプションは、ルータ ノード構成と組み合わせて使用します。このオプションにより、ルータ ノードでの MPL メッセージの処理と転送が可能になります。このオプションはライブラリ バージョンでは常に有効になっており、変更でき ないことに注意してください。

•R_DEV_FIXED_GTKS=1このオプションは、定義済みのGTKとLGTKを
境界ルーター。このオプションを設定すると、次のグループ キーがデフォルトで設定されます。

GTK0: 0102030405060708090a0b0c0d0e0f10

GTK1: 0202030405060708090a0b0c0d0e0f10

GTK2: 0302030405060708090a0b0c0d0e0f10

GTK3: 0402030405060708090a0b0c0d0e0f10

LGTK0: 0502030405060708090a0b0c0d0e0f10

LGTK1: 0602030405060708090a0b0c0d0e0f10

LGTK2: 0702030405060708090a0b0c0d0e0f10

このオプションを省略すると、BR は代わりにランダム キーを作成します。カスタム GTK および LGTK は、シリアル API コマンド
「GTKS_SETR」および「LGTKS_SETR」によって設定することもできます。

・R_FLASH_ENABLED=0このオプションは、電源サイクル間の状態維持をサポートするために、特定のデータ（送信フレーム カウンターな ど）を保存するために使用する不揮発性メモリのタイプを指定します。このオプションは、RX65x および RX65N マイクロコントローラでのみサポートされます。
R_FLASH_ENABLED オプションには次の値を指定できます。

- R_FLASH_ENABLED = 0（デフォルト） 不揮発性ストレージなし、RAM経由のエミュレーション
- R_FLASH_ENABLED = 1 内部データフラッシュを使用した不揮発性ストレージ
- R_FLASH_ENABLED = 2 外部E2PROMを使用した不揮発性ストレージ

詳細：

・デフォルト設定 R_FLASH_ENABLED = 0 を使用すると、アプリケーションは RX65x または RX65N デバイスの内部 RAM を状態維持関連データの保存に使用します。この構成は、FLASH ストレージをエミュレートする基本的な例として考えることができます。

・使用する RX65x または RX65N ターゲット デバイスが内部データ FLASH をサポートしている場合にのみ、オプション R_FLASH_ENABLED = 1 を有効にしてください。この場合、状態維持関連データの保存は、RX フラッシュ モジュールとフラッシュ メモリ データ管理モジュール（DATFRX）を使用して内部データ FLASH で行われます。

・オプション R_FLASH_ENABLED = 2 は、外部 E²PROM を使用して状態維持関連データの保存を有効にします。アプリケーションは、SCI12 に接続され、シンプルな I²C モードで動作する E²PROM タイプ M24C64 を使用します。

注：

・R5F56519ADFP デバイス（データ FLASH なし）を使用する MB-RA604S-02 評価ボードの場合は、このオプションをゼロに設定してください。
この場合、FLASH は内部 RAM を介してエミュレートされます。

・RL78/G1Hデバイスの場合、このオプションは無関係です。
不揮発性データストレージは常に内部データフラッシュを使用して実行されます。

- RP_CONF_RX_BUF_NUM=6 CWX PHY ドライバーを使用するノードの場合、このオプションは PHY ドライバーによって割り当てられる受信バッファーの数を制御します。数値が低いほどメモリ フットプリントは小さくなりますが、短時間に複数のフレームを受信した場合、データ損失の可能性が高くなります。デフォルト値は 6 です。

- R_PLATFORM_G_HYBRID=1このオプションはYG-HYBRID-PLC-RFボードの使用を有効にします ターゲットハードウェアとして。

- FEM_SE2435Lこのオプションは、Skyworks SE2435Lフロントエンドモジュールの使用を有効にします。このオプションを有効にすると、PHYの初期化はSE2435Lフロントエンドモジュールの特性に従って行われます。デフォルトではアンテナ1が使用され、動作モードは

(ANT1 ポート) / ケース 2。詳細については、アプリケーション ノート「RL78/G1H、RAA604S00 RF フロントエンド コンポーネントの制御方法」(ドキュメント番号 R01AN4968) も参照してください。この構成は、YG-HYBRID-PLC-RF ボードで Wi-SUN FAN スタックを実行する場合に必要です。

- FEM_SE2435L_ANT2_PORT_CASE1 このオプションは、Skyworks SE2435L フロントエンド モジュールと組み合わせて、アンテナ 2、動作モード (ANT2 ポート) /ケース 1 の使用を有効にします。このオプションは、定義 "FEM_SE2435L" に加えて指定してください。この構成は、VIZMO_MODULE_28dBm または YG- HYBRID-PLC-RF ターゲット ハードウェアで Wi-SUN FAN スタックを実行する場合に必要です。

次のコンパイル時オプションは、Wi-SUN FAN スタックのソース コード バージョンに対してのみ変更できますが、ライブラリ バージョンに対してはユーザーが変更することはできません。

- R_SHARED_NWK_MEM=0|1 このオプションは、すべてのモジュールとタスクで共有される単一のヒープの使用を有効にします。無効にすると、各モジ ュール (認証、MPL、タスク間メッセージングなど) に専用のヒープが使用されます。境界ルータの場合、この設定は無効です (つまり、境界ルータは、より高いメモリ要件を犠牲にしてヒープ メモリの可用性をより確実にするため、異なるモジュールに専用のヒープを使用します)。その他のすべてのビルド構成 (RN、LFN) の場合、メモリ効率を向上させるためにこの設定が有効になっています。Wi-SUN スタックのライブラリ バージョンの場合、プロジェクト設定は Wi-SUN スタックを含む静的ライブラリと一致している必要があるため、この設定を変更してはなりません。

次のコンパイル時オプションは固定されており、ユーザーが変更することはできません。

- RP_WISUN_FAN_STACK=1 このオプションは、プロトコル スタックの Wi-SUN FAN 機能を有効にします。このオプションは変更しないでください。

- RTOS_USE=1 このオプションは、Wi-SUN FANスタックによるRTOSの使用を有効にします。変更しないでください。
  このオプション。

- FreeRTOS=1 このオプションは、FreeRTOSをリアルタイムオペレーティングシステムとして有効にします。変更しないでください。

- RP_CPU_CLK=80/96 このオプションは、RL78/G1H、RX65x、
  または RX65N マイクロコントローラ。このオプションは変更しないでください。

- RP_aMaxPHYPacketSize=1600 このオプションは、PHY ドライバーがサポートする最大 PSDU 長を指定します。このオプションは変更しないでください。

- RP_CONF_MAX_TX_ACK_DATA=54 CWX PHYドライバを使用するノードの場合、このオプションは確認応答フレームの最大サイズ。この値は変更しないでください。

- R_MINIMAL_ROUTER_NODE=1 このオプションは、Wi-SUN FAN スタックの RAM メモリ要件を削減し、スタックを RL78/G1H デバイスのメモリ フットプリントに適合させます。RL78/G1H デバイスで Wi-SUN FAN スタックを使用する場合は、このオプションを変更しないでください。

- MCU_R78G1H このオプションは、ターゲット CPU が RL78/G1H ベースであることを指定します。RL78/G1H デバイスで Wi-SUN FAN スタックを使用する場合は、このオプションを変更しないでください。

8.3 FreeRTOS タスク

Renesas Wi-SUN FAN サンプル アプリケーションは、オペレーティング システムとして FreeRTOS を使用し、デフォルトで 3 つの異なるタスクを定義します。

1. MAC タスク（最高優先度）
2. NWKタスク
3. APLタスク（最低優先度）

これらの組み込みタスクと、スタック メモリ、優先度、タスク ID などの関連リソースは、 *app/config/r_os_wrapper_config.h内のカスタム プリプロセッサ マクロとディレクティブによって定義されます* 。これらの構造により、FreeRTOS を直接変更したり操作したりすることなく、FreeRTOS タスクを定義できます。Renesas Wi-SUN FAN サンプル アプリケーション内のタスクを定義する*R_OS_TASK_LIST_ENTRY*マクロには、次の引数があります。

1. *id*はタスクIDです（上記の要件を参照）
2. *名前*はタスクを識別/説明する文字列です
3. *func*はこのタスクのメイン関数の名前です
4. *スタック*は、このタスクに割り当てられたスタックのサイズ（バイト単位）です。
5. *優先度*はタスクの優先度です（上記の要件を参照）
6. *suspended* は、タスクを最初に停止状態にするかどうかを示すブール値です。

8.3.1 カスタムFreeRTOSタスク

既存のタスクIDを変更しないでください。r_os_wrapper_config.h内のIDはいずれも変更しないでください。

サンプル アプリケーションに FreeRTOS タスクを追加する必要がある場合は、前のセクションで説明したのと同じ概念を使用することをお勧めします。この方法では、新しいタスクを追加するために必要なコード変更は最小限に抑えられ（図 7-7 を参照）、FreeRTOS の変更や直接のやり取りは必要ありません。新しいタスクを定義するときは、次の要件に細心の注意を払ってください。

- タスク ID は、ギャップのない昇順である必要があります（つまり、システム内の最高のタスク ID が 2 の場合、新しいタスクはタスク ID 3 を使用する必要があります）。
- 各タスクには一意のタスクIDが割り当てられている必要があります
- 新しいタスクの優先度は 0 にする必要があります（通常の WiSUN FAN プロトコル スタックの動作を妨げないようにするため）
- タスク関数には任意の名前を使用できますが、そのシグネチャは変更できません（戻りパラメータを持たず、引数として単一の void ポインタを受け入れる必要があります）。
- *R_OS_NUM_TASKS*の値は*R_OS_TASK_LIST_ENTRY*要素の数と等しくなければなりません。

*r_os_wrapper_config.h*のタスク定義に加えて、タスク エントリ関数を提供する必要があります。FreeRTOS では、このタスク エントリ関数が決して戻ったり終了したりしないようにする必要があることに注意してください。したがって、このような関数は通常、無限ループを使用して実装されます。図 8-7 で定義された新しいタスクに対応するタスク エントリ関数は次のようになります。

```
void customer_tsk(void* 未使用)
{
    while(1)
    {
```

```
        // カスタム処理を実行します…。
        R_OS_DelayTaskMs(10000);
    }
}
```

```
215 /***********************************************************************
216 * Tasks
217 ***********************************************************************/
218 #define ID_mac_tsk              0
219 #define ID_nwk_tsk              1
220 #define ID_apl_tsk              2
221+#define ID_customer_tsk         3
222
223 #define MAC_TSK_DEFAULT_PRIORITY 3
224 #define NWK_TSK_DEFAULT_PRIORITY 1
225 #define APL_TSK_DEFAULT_PRIORITY 0
226
227 #define HIGHEST_TSK_PRIORITY     4
228
229 void mac_tsk(void*);
230 void nwk_tsk(void*);
231 void apl_tsk(void*);
232+void customer_tsk(void*);
233
234 #if R_MINIMAL_ROUTER_NODE || R_MODEM_SERVER
235 // 308 bytes seem to be used for some internal state
236 #define ID_mac_tsk_stack_size (1200 + 308)
237 #define ID_nwk_tsk_stack_size (1800 + 308)
238 #define ID_apl_tsk_stack_size (2400 + 308)
239 #else
240 #define ID_mac_tsk_stack_size (4 * 1200)
241 #define ID_nwk_tsk_stack_size (4 * 2400)
242 #define ID_apl_tsk_stack_size (4 * 2400)
243 #endif
244
    -#define R_OS_NUM_TASKS       3
245+#define R_OS_NUM_TASKS       4
246 R_OS_TASK_LIST_BEGIN()
247 R_OS_TASK_LIST_ENTRY(ID_mac_tsk, "MAC_TSK", mac_tsk, ID_mac_tsk_stack_size, MAC_TSK_DEFAULT_PRIORITY, 1)
248 R_OS_TASK_LIST_ENTRY(ID_nwk_tsk, "NWK_TSK", nwk_tsk, ID_nwk_tsk_stack_size, NWK_TSK_DEFAULT_PRIORITY, 1)
249 R_OS_TASK_LIST_ENTRY(ID_apl_tsk, "APL_TSK", apl_tsk, ID_apl_tsk_stack_size, APL_TSK_DEFAULT_PRIORITY, 0)
250+R_OS_TASK_LIST_ENTRY(ID_customer_tsk, "customer_tsk", customer_tsk, 4500, 0, 0)
251 R_OS_TASK_LIST_END()
252
253 #endif /* R_OS_WRAPPER_CONFIG_H */
```

図8-7 カスタムFreeRTOSタスクの定義

8.4 サポートされている周波数帯域とチャネルパラメータ

次の周波数帯域とチャネル パラメータは、さまざまな PHY ドライバ バージョン（TRG および CWX）の Wi-SUN FAN プロトコル スタックによってサポートされています。

8.4.1 サポートされている周波数帯域、PHY ドライバ TRG (FSK)

周波数バンド (MHz)	地域	規制ドメイン値	オペレーティングクラス	PHY オペレーティングモード (シンボルレート)	チャンネル間隔 (kHz)	合計チャンネル数	中心チャンネル周波数0 (MHz)
863-870	EU	0x03	1	#1a (50kbps)	100	69	863.1
			2	#2a (100kbps)	200	35	
870-876			3	#1a (50kbps)	100	55	870.1
			4	#2a (100kbps)	200	27	870.2
865-867	IN	0x05	1	#1a (50kbps)	100	19	865.1
			2	#2a (100kbps)	200	10	
866-869	SG	0x0D	1	#1a (50kbps)	100	29	866.1
			2	#2a (100kbps)	200	15	
902-928	NA	0x01	1	#1b (50kbps)	200	129	902.4
			2	#3 (150kbps)	400	64	902.4
			2	#4a (200kbps)	400	64	902.4
902907.5 915-928	BZ	0x07	1	#1b (50kbps)	200	90	902.2
			2	#3 (150kbps)	400	43	902.4
915-928	AZ/NZ	0x08	1	#1b (50kbps)	200	64	915.2
			2	#3 (150kbps)	400	32	915.4
920-928	JP	0x02	1	#1b (50kbps)	200	38	920.6
			2	#2b (100kbps)	400	18	920.9
			2	#3 (150kbps)	400	18	920.9

表8-9 サポートされている周波数帯域、PHYドライバTRG

8.4.2 サポートされている周波数帯域、PHY ドライバー CWX (FSK および OFDM)

周波数帯域 (MHz)	規制ドメイン	PHYモードID	チャンプランID	有効合計数	チャンネルマスク	オペレーティングクラス リファレンス
863-870	EU1 (0x03)	1	32	62	00:00:00:00:00:00:80:E1 : E6	1
		3 84-86	33	29	00:00:00:D8:FB	2
870-876	EU2 (0x03)	1	34	55	00:00:00:00:00:00:80	3
		3 84-86	35	27	00:00:00:F8	4
863-876	EU3 (0x03)	1	36	118	00:00:00:00:00:00:80:E1 : 06:00:00:00:00:00:00:E0	-
		3 84-86	37	56	00:00:00:D8:03:00:00:C0	-
902-928	NA (0x01)	2, 3 84-86	1	129	00:00:00:00:00:00:00:00: 00:00:00:00:00:00:00:00: FE	1
		5, 6 68-70	2	64	00:00:00:00:00:00:00:00	2
		51-54	4	32	00:00:00:00	-
		34-38	5	21	00:00:E0	-
902-907.5 915-928	BZ (0x07)	2, 3 84-86	1	90	00:00:00:FC:FF:FF:FF:FF :01:00:00:00:00:00:00:00: FE	1
		5 68-70	2	43	00:F0:FF:FF:01:00:00:00	2
		51-54	4	22	C0:FF:00:00	-
		34-38	5	13	F8:07:E0	-
920-928	JP (0x02)	2、84-86	21	29	FF:01:00:00:C0	1
		5 68-70	22	14	0F:00:FC	2
		51-54	24	7	03:FE	-
865-868	IN (0x05)	1	39	29	00:00:00:E0	1
		3	40	15	00:80	2

表8-10 サポートされている周波数帯域、PHYドライバCWX

8.5 限定機能ノード（LFN）

限定機能ノード（LFN）は、低電力/バッテリー動作向けに最適化されています。LFN は、FAN ネットワークの検出と参加、または IPv6 パケットの送受信のための最小限の機能を提供します。LFN は FAN 1.1 FFN の子としてのみ動作し、他のノードを親にすることはできません。LFN の機能と特徴の詳細については、Wi-SUN FAN TPS を参照してください。

現在、ルネサスの Wi-SUN FAN プロトコル スタックは、RL78/G1H デバイスの省電力メカニズムのみをサポートしています。Wi-SUN FAN プロトコル スタックは、RL78/ G1H デバイスの STOP モード スタンバイ機能を利用して、システムの動作電流を削減します。デバイスはほとんどの時間 STOP モードのままで、ユニキャストおよびブロードキャストのサンプリングされたリスニング ウィンドウ中に起動して、FAN1.1 FFN 親と通信します。RL78/G1H ベースの LFN が起動すると、アクティブな省電力メカニズムによって、シリアル API コマンドへの応答性が制限されます。いずれにしても、デバイスのデータ表示（たとえば、UDP データ表示）は、シリアル API コマンドを利用してデバイスによってさらに処理および報告されます。

デモ用に、RX65x および RX65N デバイスでは LFN 構成もサポートされています。このソリューションでは、省電力機能を除き、FAN ネットワークの検出と参加、または IPv6 パケットの送受信に関するすべての LFN 関連機能がサポートされています。これにより、動作中にシリアル API インターフェイスを介して LFN デバイスを完全に制御し、たとえばデバイスから RLog ログ情報を取得するなどして、デバイス動作を完全に透過的にすることができます。

8.6 変調およびデータレートネゴシエーション（MDR）

MDR は、Wi-SUN FAN TPS バージョン 1.1 でオプション機能として導入されました。TPS では、PAN 内のすべてのノードが同じ基本 PHY 動作モードを使用することが依然として求められますが、MDR では、送信側と受信側の両方でそれぞれのモードがサポートされている場合、ノードが別の PHY 動作モードに一時的に切り替えることができます。MDR のコンテキストでは、これらの追加の PHY 動作モードは*新しいモードと呼ばれます*。ノードは、POM-IE を介して、サポートされている新しいモードのセットを相互に検出します。MDR の詳細については、Wi-SUN FAN TPS を参照してください。

現在、ルネサスWi-SUN FANプロトコルスタックは、モードスイッチPPDUに基づくモードスイッチのみをサポートしており、MACコマンドフレームはサポートしていません。さらに、MDRはCWXプラットフォームでのみサポートされており、FANプロトコルスタックのバリアントと組み合わせて「高スループット」（HT）をサポートしています。
「コア」バリアント。そうでない場合、MDR サポートは FAN スタック内で無効になり、対応するパラメータは使用できなくなるか、単に無視されます。

ノードで MDR を有効にするには、アプリケーションは*R_NWK_mdrNewModes*属性の NWK SET 要求を発行して、ノードが基本 PHY 動作モードに加えて MDR に使用できる最大 4 つの PHY 動作モードのセットを構成することができます。新しいモードのセットが構成されると、ノードはこれらの新しいモードを自身の POM-IE に含め、近隣ノードによる新しいモードの検出を有効にします。さらに、このノードの PHY 層は、これらの PHY 動作モードへのモード切り替えを実行できるようになります。初期化中にノードに新しいモードは構成されないため、アプリケーションによって少なくとも 1 つの新しいモードが明示的に構成されない限り、ノードはモード切り替えを実行しません。R_NWK_mdrNewModes が繰り返し設定されると、以前の値はすべて Wi-SUN FAN プロトコル スタックから削除され、最近指定された新しいモードのみがノードで使用可能になります。

デフォルトでは、送信フレームの PHY 動作モードの選択は MAC 層によって行われます。MAC 層は、各ネイバーから受信した POMIE に基づいて、各ネイバーでサポートされている新しいモードのリストを自動的に検出して維持します。このように、ノードは、自身と受信ネイバーでサポートされている新しいモードのみを考慮します。これは、サポートされている新しいモードの両方のセットの重複です。さらに、ノードは、各ネイバーとの通信でサポートされている個々の PHY モードごとに ETX 値を維持します。これらの ETX 値は、データ レートの改善（ベース モードのデータ レートと比較）と信頼性の組み合わせです。送信ごとに、MAC 層は、しきい値 140 を超えない限り、次の送信で ETX 値が最も低い PHY 動作モードを自動的に選択します。すべての新しいモードがこのしきい値を超えると、ノードはベース PHY 動作モードにフォールバックします。ノードが EACK を 2 回受信しない場合は、フレームの後続の送信でもベースにフォールバックします。すべての PHY 動作モードの ETX 値が定期的に更新されるようにするために、MAC 層には、ETX 値がより優れた他のモードが存在する場合でも、一定時間 ETX 値が更新されていない PHY 動作モードが次の送信に優先されるようにするプローブ メカニズムが含まれています。

アプリケーションは、特定のMAC層の自動PHY動作モード選択をオーバーライドすることができる。UDPまたはICMPv6データ要求を、それぞれのデータ要求のMDRパラメータを変更することで制御します（セクション10.1.11および10.1.9を参照）。ベースPHY動作モードの使用を強制するか（*R_MAC_MDR_SELECTOR_DISABLED*）またはカスタムphyModeIdの使用（*R_MAC_MDR_SELECTOR_FIXED*を使用し、カスタム phyModeId を指定します。データ要求で指定されるカスタム phyModeId は、前述のように*R_NWK_mdrNewModes*属性を使用してデバイス上で事前に設定されている必要があることに注意してください。そうでない場合、データ要求は失敗します。ただし、ノードは、カスタム phyModeId が受信ネイバーでもサポートされているかどうかをチェックしなくなります。

8.7 Ｊｏｉｎメトリック

Wi-SUN FAN 1.1 では、Joinノード（Join状態 1）が最適な PAN および EAPOL ターゲットを決定するために使用できるJoinメトリックが導入されました。Joinメトリックはボーダールータによって作成および送信され、専用の情報要素 （JM-IE） を介して他のノードによって PAN 全体に配布されます。Joinメトリックと PAN 選択の詳細については、Wi-SUN FAN TPS の現在のバージョンを参照してください。現在、PAN 負荷係数（PLF）は、Wi-SUN FAN TPS によって定義されている唯一のJoinメトリックです。これは、PAN の相対負荷をパーセンテージで指定します。

Wi-SUN FAN スタックには、すべての送信 PA および LPA フレームに JM-IE が含まれています。WiSUN FAN TPS の推奨に従い、最後に受信した JM-IE バージョン更新以降に PA 送信を抑制した場合、FFN はブロードキャスト ULAD フレームに JM-IE も含め、PAN 全体にわたる JM-IE 更新の配信を加速します。

ボーダールータでは、JM-PLF の計算と送信は、Wi-SUN FAN スタックでデフォルトで有効になっています。Wi-SUN FAN スタックを起動する前に、NWK 属性*R_NWK_jmPlfEnabled*を false に設定することで無効にできます。PAN 内のノードの最大数は、 Wi-SUN FAN スタックを起動する前に*R_NWK_panCapacity で設定する必要がありますが*、現在のノード数は、Wi-SUN FAN スタックの起動後に、アプリケーションによって*R_NWK_panSize*属性で継続的に更新されることが予想されます。これらの値は、Wi-SUN FAN スタックによって現在の PLF を計算するために使用されます。提供されているサンプル アプリケーションは、初期化中にノードの最大数を*R_BR_PAN_SIZEの値に設定します*。これは、認証モジュールのサプリカント テーブルのサイズなどを定義するためです。新しいサプリカントがサプリカント テーブルに追加されると、認証モジュールは Wi-SUN FAN スタック内の現在のノード数を更新し、PLF が常に最新の状態になるようにします。

Joinノード（Join状態 1）は、常に着信 PAN アドバタイズメント（PA）フレームをチェックして、JM-IE と JM-PLF メトリックを探します。JM-PLF が存在する場合、Joinノードは PLF を使用して、最も適切な PAN および EAPOL ターゲットを決定します。これは、最も低い PLF を持つターゲットになります。JM-PLF が存在しない場合は、 Wi-SUN FAN 1.1 以前と同様に、着信 PA からの*PanRoutingCost 値*と*Pan Size*値を使用して EAPOL 候補が選択されます。Joinノードが混合 PA（一部に JM-PLF が含まれ、一部に含まれない）を受信した場合、JM-IE を含む PA が常に優先されます。つまり、JM-IE のない PA は、JMPLF 値が 100% 未満の PAN 負荷係数を示さない場合にのみ考慮されます。

8.8 ネイバーキャッシュ

Wi-SUN FAN プロトコル スタックのネイバー キャッシュは、RPL 親キャッシュと子ノード キャッシュの 2 つの論理キャッシュで構成されます。

近隣キャッシュ	説明	状態
RPL 親キャッシュ	RPL 親ネイバー キャッシュには、すべての可能性のある RPL 親が含まれます。キャッシュ エントリは、可能性のある親から DIO メッセージを受信することによって登録されます。対応する RPL 親キャッシュ エントリのステータスと使用状況は、ステータス情報によって示されます。	0 = ″RPL 親ではない″ 1 =「RPL親候補」 2 = 「優先RPL親」 3 =「代替RPL親」

子ノードキャッシュ	子ノード キャッシュには、すべての子ノードが含まれます。キャッシュ エントリは、ノードからアドレス登録オプションを含む近隣要請メッセージを受信することによって登録されます。子ノード キャッシュ エントリは、ステータス情報「子ノード」によって示されます。子ノードのリンク メトリックは使用も維持もされないため、常にゼロであることに注意してください。	4 = 「子ノード」

表8-11 近隣キャッシュ

ネイバー キャッシュの分割により、同じネイバーに対して RPL キャッシュに 1 つのエントリ、子ノード キャッシュに 1 つのエントリが存在することは珍しくありません。

8.9 マルチキャスト

Wi-SUN FAN プロトコル スタックは、Wi-SUN FAN TPS の要件に従って IPv6 マルチキャストをサポートします。*低電力および低損失ネットワーク向けマルチキャスト プロトコル（MPL）*は、レルム スコープ（またはより高スコープ）のマルチキャスト送信を実現するために使用されます。

8.9.1 マルチキャストグループ

Wi-SUN FAN プロトコル スタックが起動されると、Wi-SUN FAN TPS によって義務付けられた次のマルチキャスト グループに自動的に参加します。

リンク範囲:

1. FAN ノードは、リンクローカルスコープの全ノードグループ（ff02::1）[RFC4291] に参加する必要があります。
2. FAN ノードは、リンクローカルスコープの全ルーターグループ（ff02::2）[RFC4291] に参加する必要があります。
3. FAN ノードは、リンクローカルスコープの全 RPL ノード グループ（ff02::1a）[RFC6550] に参加する必要があります。

レルムスコープ:

4. FAN ノードは、レルム スコープの全ノード グループ（ff03::1）に参加する必要があります。
5. FAN ノードは、レルムスコープの全ルーター グループ（ff03::2）に参加する必要があります。
6. FAN ノードは、レルムスコープ all-mpl-forwarders グループ（ff03::fc）に参加する必要があります。

各デバイスは、上記の必須グループを含め、最大 8 つの異なるマルチキャスト グループに同時に参加できます。特定のマルチキャスト グループに参加または離脱するための NWK API 機能と、対応する NWK 表示については、それぞれセクション 10.1 と 0 で説明します。

マルチキャスト動作の詳細については、Wi-SUN FAN TPS を参照してください。

8.9.2 MPL の構成（レルムスコープ以上）

MPL の動作には、トリクル タイマーに基づいて受信したマルチキャスト メッセージのバッファリングと繰り返し転送が含まれるため、その構成はパフォーマンス/信頼性とリソース消費（特に RAM 使用量）の間でトレードオフになります。さまざまなユースケースに適切な構成を可能にするために、サンプル アプリケーションには、 *app/config*ディレクトリ内に MPL 構成をカスタマイズするための 2 つの構成ファイルが含まれています。

- ソースファイル*r_mpl_config.c*には、実行時に特定のMPLパラメータを設定するための変数が含まれています。
  例えば、維持されるシードの最大数やバッファリングされるメッセージの最大数など
- ヘッダーファイル*r_mpl_config.h*には、コンパイル時に特定のMPLパラメータ（MPLモジュールに割り当てられるヒープメモリの量など）を制御するためのプリプロセッサディレクティブが含まれています。

詳細については、それぞれのシンボルのソース コードとドキュメントのコメントを参照してください。

8.10 電源サイクルによる状態維持（不揮発性メモリ）

電源サイクル後の状態維持を可能にするために、次のデータが不揮発性メモリに保存され、電源サイクル後に復元される可能性があります。

すべてのデバイスタイプ:

- NWK バージョン（バージョン情報、保存されたデータの形式がファームウェアで想定されるデータ形式と一致しているかどうかを示す）
- 送信フレームカウンタ（セクション7.11.3を参照）
- GTK/LGTK
- GTK/LGTKハッシュ
- 送信 MPL シーケンス番号（デバイスが MPL 送信をサポートしている場合）

ボーダールータのみ:

- GTK/LGTK の有効期間
- アクティブ GTK/LGTK インデックス
- 放送スケジュール識別子
- DODAG バージョン
- DTSN
- PANバージョン
- LFNバージョン

ルーターノードのみ:

- 認証子 EUI-64

デフォルトでは、有効なデータが不揮発性メモリに保存されている場合でも、電源サイクル後の FAN 状態の復元は無効になっています。サンプル アプリケーション内で直接有効にすることも（変数*AppCmdCo*

nfig.stateMaintenanceThroughPowerCycle)、DEVCONF_SETR シリアル コマンド（セクション 14.1.2 を参照）することもできます。この設定を有効にすると、サンプル アプリケーションは再起動後に不揮発性メモリに保存されているすべてのデータを自動的に復元し、Wi-SUN FAN スタックが電源サイクル前と同じ状態で起動されるようにします。

重要: 不揮発性メモリのデータは、データの書き込みに使用されたのと同じファームウェアとデバイスタイプ構成を使用してのみ復元できます。したがって、新しいファームウェア イメージがデバイスにダウンロードされた場合、またはデバイス タイプが変更された場合（例: BR -> RN)、不揮発性メモリを完全に消去（フォーマット）する必要があります。前者の状況は、保持する実際のデータとともに NWK バージョンを不揮発性メモリに追加で書き込むことによって検出されます。データの復元時に不揮発性メモリ内に保存されている NWK バージョンが現在の NWK バージョンと一致しない場合は、フラッシュ フォーマットがトリガーされ、不揮発性メモリがクリーンな状態にリセットされます。

8.10.1 ルータノードの状態の永続的な保存

電源サイクル後の起動時に以前使用した GTK/LGTK を復元できるルータ ノードは、初期認証（参加状態 2 中）をスキップして、ブートストラップ プロセスを高速化します。この動作は、ネットワーク内で多数の同時認証試行を回避するため、複数のデバイスの電源が同時に失われたり回復したりする場合の遅延を回避するのに役立ちます。ルータ ノードは GTK のみを復元でき、PTK または PMK は復元できないことに注意してください。したがって、GTK のみが変更された場合でも、ボーダールータとの次の認証交換は常に完全な EAP-TLS 交換になります。

電源中断期間中にネットワーク内で使用されるアクティブな GTK が変更された場合、ルータ ノードは参加状態 3 で PAN 構成フレームを復号化できないため、PCS_MAX_TRANSMISSIONS 後に参加状態 1 に戻り、復元された GTK を削除して、FAN TPS で説明されている完全な参加手順を実行します（参加状態 2 中の完全な認証交換を含む)。

電源中断期間中にボーダールータによって新しい GTK が設定された場合、または古い GTK が削除された場合、デバイスは PAN 構成フレーム内の GTKHASH-IE を介して GTK の変更を検出し、ボーダールータに GTK 更新を直ちに要求します。その結果、上記の理由により完全な EAP-TLS 交換が行われます。

8.11.2 ボーダールータの状態の永続的な保存

ボーダールータは、以前使用された FAN 状態、および GTK と LGTK をその有効期間とともに復元します。これにより、ボーダールータが再び動作可能になると、以前参加していたデバイスが動作を継続できます。不揮発性メモリから復元されたキーの有効期間は、以前のものと完全に一致せず、常にわずかに長くなることに注意してください。その理由の 1 つは、現在のハードウェア プラットフォームでは、電源サイクル間で実行し続けるバッファ クロックが提供されていないため、デバイスの電源が切断されている間隔を考慮できないことです。キーの有効期間は、デバイスの現在のクロックの秒値とともに、絶対タイムスタンプ（クロックの秒数、つまりデバイスの電源がオンになってから経過した秒数）として不揮発性メモリに保存されます。後者は、データが書き込まれるたびに更新されます。これは、復元後にキーの有効期間にずれが生じるもう 1 つの理由です。デバイスの実際のクロックと不揮発性メモリに保持されている値の間には常に小さなギャップがあるためです。

FAN スタックが再起動されると、ボーダールータはブロードキャスト スケジュール識別子（BSI）も変更し、PAN バージョンを増分して、ネイバーとの再同期を高速化します。 不揮発性メモリの消耗を減らして寿命を延ばすために、ボーダールータはネイバー キャッシュまたはソース ルーティング テーブルを不揮発性メモリに保存しません。 これらのキャッシュの再設定を高速化するために、ボーダールータは再び動作可能になると（参加状態 5)、DODAG バージョンと DTSN を増分します。 この操作は、「グローバル修復」または「DODAG 修復」とも呼ばれます（RFC 6550 を参照)。 DODAG バージョンを増分すると、ネットワーク内のすべてのルータ ノードが自身の DODAG バージョンを増分し、親の選択を再実行し、ARO で新しいネイバー要請メッセージを送信するように強制されます。これにより、ボーダールータ上の ARO ネイバー キャッシュを高速に再設定できます。 DTSN を増分すると、ネットワーク内のすべて

のルータ ノードから新しい DAO メッセージが強制され、ソース ルーティング テーブルを高速に再設定できます。ボーダールータがネットワーク内で設定されたバージョンよりも高い DODAG バージョンを検出した場合（たとえば、DODAG バージョンが適切に復元または増分されていないため）、ボーダールータはネットワーク内に存在する最高の DODAG バージョンを自動的に使用します。

8.11.3 フレームカウンタの永続的な保存

Wi-SUN FAN スタックは、 *R_NWK_macFrameCounterOffsetで定義される特定のフレーム受信間隔で、TX フレーム カウンターを不揮発性メモリに保存します*。この間隔は、NWK API を介して設定できます（第 11 章を参照）。デバイスの TX フレーム カウンターの不揮発性ストレージは、Wi-SUN FAN TPS が電源サイクル後に機能を維持するために必要なため、 stateMaintenanceThroughPowerCycle 設定が有効になっていない場合でも、デバイスの起動後に認証モジュールが起動すると、フレーム カウンターは自動的に復元されます。*RX65x*デバイスの場合、フレーム カウンターの保存は、内部 RAM（FLASH エミュレーション）、内部データ FLASH、または外部 E²PROM を使用して行うことができます。選択は、コンパイル時オプション R_FLASH_ENABLED によって行われます。詳細については、第 7.3 章を参照してください。RL78/G1H デバイスの場合、フレーム カウンターの保存は常に内部データ FLASH を使用して行われます。

RX フレーム カウンターのストレージはさまざまな方法で実現できます。それぞれの方法で、ストレージ要件とリプレイ攻撃に対する堅牢性の間の妥協点が提供されます。組み込みデバイス内のストレージの可用性は無限ではないため、実際には情報のストレージはリスニング範囲内のすべてのノードのサブセットに制限されます。デバイス テーブルが完全に設定されると、デバイス テーブルに存在しないノードからの受信フレームは、デバイス テーブル ストレージのオーバーフローによるリプレイ攻撃を回避するためにすぐにドロップされます。デバイス テーブル内のエントリは、アクティブに使用されていない場合、一定期間が経過すると期限切れになります。エントリの期限が切れると、新しい受信フレーム用に新しいエントリを作成できます。

RX フレーム カウンターの保存には次のオプションが考えられます。

1. 最大限のセキュリティ
    a. すべてのRXフレームカウンタをRAMおよび/または不揮発性メモリに保存し、受信フレームごとに更新します。
    b. 大容量の不揮発性メモリ（24バイト×Nノード）と非常に頻繁な書き込み操作が必要
    c. 運用中の完全なリプレイ攻撃防止
    d. 電源投入後のリプレイ攻撃を完全に防止
2. 高中程度のセキュリティ
    a. すべてのRXフレームカウンタをRAMに保存し、受信フレームごとに更新する
    b. 保存されたRXフレームカウンタをRAMから不揮発性メモリに定期的にコピーする
    c. 大容量の不揮発性メモリ（24バイト×Nノード）と頻繁な定期的な書き込み操作が必要
    d. 運用中の完全なリプレイ攻撃防止
    e. 前回の周期間隔中に送信されたフレームを使用してリプレイを行うことができるため、電源サイクル後のリプレイ攻撃が軽減されます。
3. 中程度のセキュリティ
    a. すべてのRXフレームカウンタをRAMに保存し、受信フレームごとに更新する
    b. RXフレームカウンタを不揮発性メモリに保存しない
    c. 不揮発性メモリの追加要件なし
    d. 運用中の完全なリプレイ攻撃防止
    e. 電源投入後のリプレイ攻撃なし

フラッシュ メモリの消去/プログラム サイクルの数が限られていることを考慮すると、フラッシュ メモリを使用する場合、オプション 1 は実現可能ではないと思われます。オプション 2 は、非常に限定された方法でのみ使用できます。100,000 サイクルで寿命が 15 年の場合、周期間隔は約 80 分になります(EEPROM を使用すると、より柔軟にすることができます)。RX フレーム カウンターの不揮発性ストレージは推奨されていますが、FAN 仕様では必須ではありません。したがって、選択するストレージ方法は、主にシステム要件によって異なります。現在の Wi-SUN FAN ソリューションは、デフォルトでオプション 3 を実装しており、RX フレーム カウンター チェックは、NWK 属性*R_NWK_macFrameCounterCheckEnabled*を設定することでアクティブ化できます。

8.11 証明書

Wi-SUN FAN スタックには、Wi-SUN のテストおよび開発目的で使用できるさまざまな X509 証明書が付属しています。

証明書の詳細 （ファイル名が「testcert.x509」であると仮定） は、次のコマンドでターミナルで表示できます。

```シェル
openssl x509 -inform der -in "testcert.x509" -text
「」
```

「 ¥certs¥certificates 」ディレクトリには、Wi-SUN FAN スタックに使用できる 2 つの証明書チェーンが含まれています。

1. 公式Wi-SUN FAN証明書作成ツールを使用して作成された「Wi-SUN Test」証明書。
2. GlobalSign が発行する IDevID（IEEE Std 802.1AR-2018）

Wi-SUN テスト証明書と GlobalSign 証明書は、開発とテストのみを目的としています。さらに、両方の証明書チェーンは、公式の Wi-SUN FAN 認証テストベッドで使用されます。

事前にインストールされた証明書は開発およびテスト目的のみに使用されることに注意してください。最終製品では使用しないでください。

Wi-SUN FAN スタックにプリインストールされているエンコード ツールを使用してバイナリ証明書ファイルをヘッダー ファイルに変換するワークフローを次の図に示します。証明書生成ツール、生成フロー、およびコマンド ライン オプションの詳細については、readme ファイル「 ¥certs¥Readme.md 」、「 ¥certs¥encode¥Readme.md 」、および「 certs¥motgen¥Readme.md 」を参照してください。

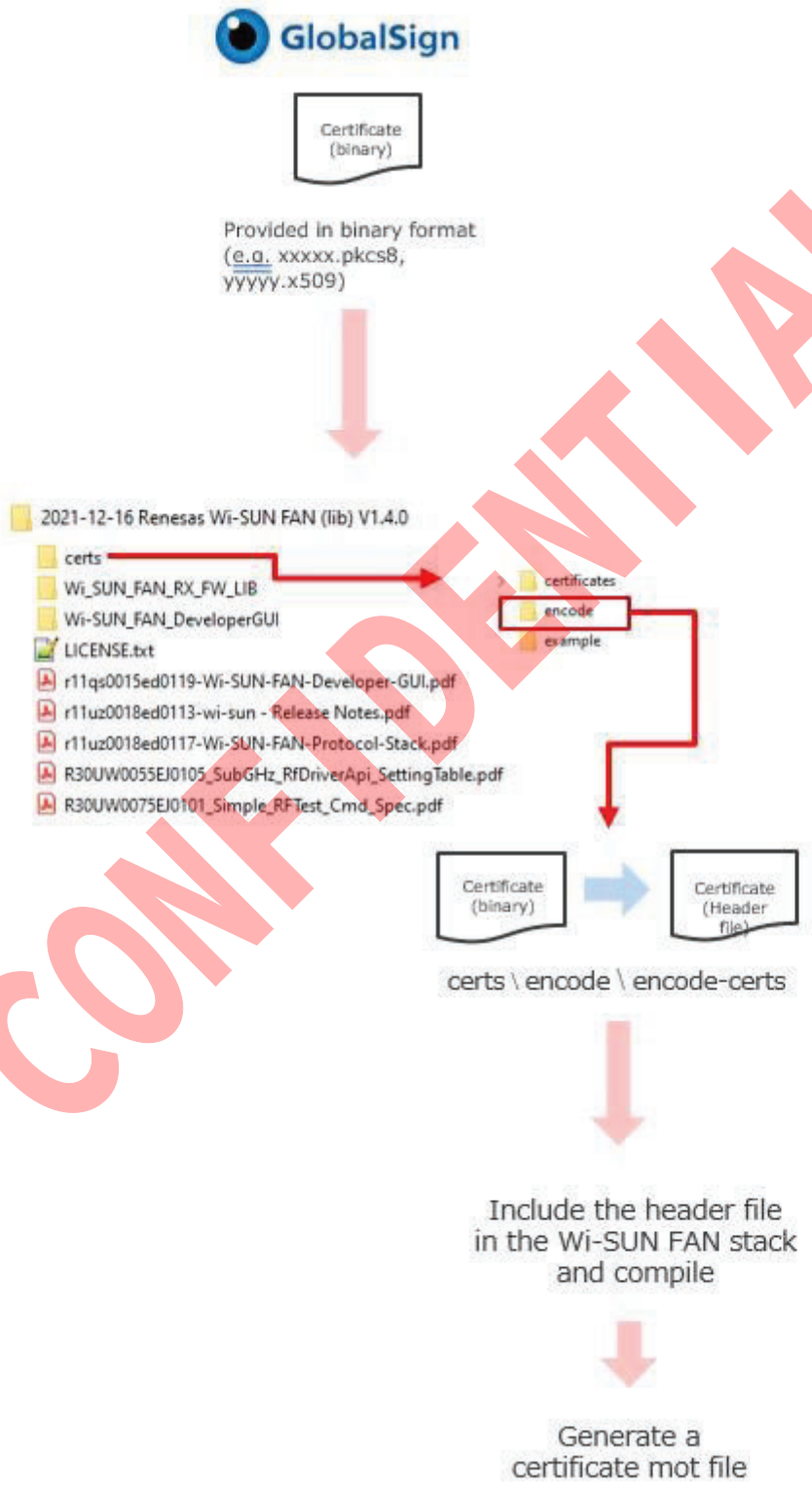


図7-8 証明書生成フロー

8.11.1 ストレージ

IEEE 802.1AR 仕様では、次のように規定されています。「DevID 機能を備えたデバイスには、製造元が提供するグローバルに一意な初期デバイス識別子 (IDevID) が組み込まれており、変更できないように保存されます。デバイスは、ネットワーク管理者によるローカルで有効なデバイス識別子 (LDevID) の作成をサポートできます。各 LDevID は、暗号バインディングに影響を与えるために使用される秘密キーを知らなければ、偽造したり、異なる IDevID を持つデバイスに転送したりすることが不可能な方法でデバイスにバインドされます。」

上記を考慮すると、IDevID と LDevID は次のように保存されます。

- IDevID は製造中に内部コードまたはデータ フラッシュに保存され、その後ロックや TSIP の使用などによって変更できないようにする必要があります。IDevID は取り消すことはできません。
- LDevID は、内部データ フラッシュまたは外部 EEPROM に保存されます。LDevID は、製造後に取り消したり設定したりできます。内部データ フラッシュをロックしたり、外部 EEPROM に暗号化して保存するなど、保護が必要です。

LDevID は、Wi-SUN FAN スタックではデフォルトではサポートされていないことに注意してください。LDevID の使用は、エンド カスタマーの責任で行ってください。

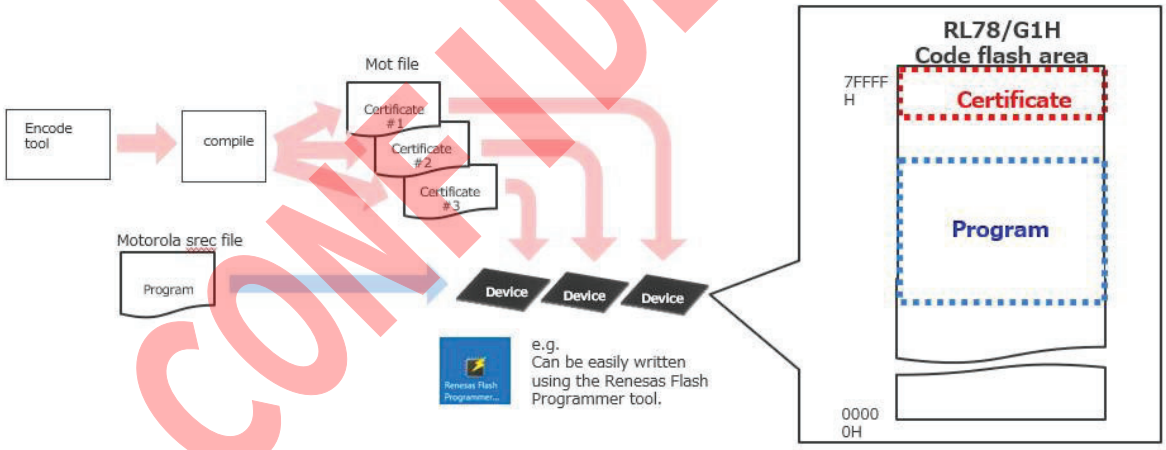


図8-9 証明書の保存 (RL78/G1H)

上図は、RL78/G1H デバイスの証明書の保存場所を示しています。RX65x デバイスの場合、証明書はコード フラッシュ領域*0xFFF00008 ~ 0xFFF02000に配置されます*。

8.11.2 作成

Wi-SUN FANスタックで使用される証明書は、公式Wi-SUNバージョン18を使用して作成されています。FAN証明書作成ツール。このツールは 、ソース コード ライブラリ パッケージの「 ¥certs¥certificates¥WisunTest¥GenWisunTestCertsV18 」ディレクトリにあります。

証明書の階層は次のとおりです。

```
Root
 ├── DevFromRoot-X
 └──MCA
       └──MICA
             └──DevFormMica-X
```

* root: 証明書作成ツールに埋め込まれたWi-SUN FANルート証明書
* MCA: ルートによって署名された製造元のCA
* MICA: MCA によって署名された製造業者中間 CA
* DevFromRoot-X: ルートによって署名されたデバイス証明書
* DevFromMica-X: MCA によって署名されたデバイス証明書

`DevFromX-1` 証明書はボーダールータ用であり、「TLS Web サーバー認証」拡張キー使用属性が含まれています。

その他の「DevFromX-Y」証明書はルータ ノード用であり、「TLS Web クライアント認証」拡張キー使用属性が含まれています。

`x509` ファイルは証明書です。`pkcs8` ファイルは秘密鍵です。

証明書の詳細は、次の方法で表示できます。

```
openssl x509 -inform der -in 証明書 -テキスト
```

対応するエンコードについては、「 ¥certs¥encode 」ディレクトリと対応する readme ファイルを参照して、WiSUN FAN ファームウェアで使用するために証明書をエンコードするために必要な手順を確認してください。さまざまな例が「 ¥certs¥example 」ディレクトリにあります。

Wi-SUN-FAN スタックの現在の実装では、証明書の処理と暗号化計算を行うために内部で mbedTLS を使用しています。原則として、mbedTLS は PEM ファイルの読み取りと書き込みをサポートしていますが、コード サイズの削減のため、コンパイル時に無効になっています。

スタック内で証明書を使用するには、証明書のバイナリ情報を含むヘッダー ファイルを作成します。ヘッダー ファイルは DER 証明書から作成する必要があります。

DERファイルをPEMファイルに変換するには、PEMファイルをCRTファイルに変換するには、たとえば、Linux では「 openssl 」。

例えば：

```
openssl x509 -inform DER -in certfile.DER -outform PEM -out certfile.PEM

openssl x509 -inform PEM -in certfile.PEM -outform CRT -out certfile.CRT
```

証明書の内容を表示する場合は、次のコマンドを使用してください。

```
openssl x509 -inform PEM -in certfile.PEM -text
```

-inform パラメータを変更して適切なファイルを指定すると、CRT ファイルまたは DER ファイルでも同じことができます。

8.12 DHCPv6 メッセージ内の追加情報の送信

DHCPv6 クライアント モジュールとサーバー モジュールはどちらも、クライアント（ルーター ノード）からサーバー（ボーダールーター）へ、またその逆方向に、任意の情報（DHCPv6 データに加えて）を送信できる API を提供します。これにより、特に大規模ネットワークでは、各 DHCPv6 クライアント（ルーター ノード）と DHCPv6 サーバー（ボーダールーター）間の要請/応答交換中に追加の情報が送信できるため、ネットワーク帯域幅を節約できます。追加のデータは、*要請メッセージと応答*メッセージに含めることができる DHCPv6 ベンダー固有の情報オプションにカプセル化されます。このオプションの詳細については、RFC 3315 のセクション 22.17 を参照してください。
この機能はデフォルトでは無効になっていますが、クライアント デバイスの NWK API（第 11 章を参照）内の*R_NWK_dhcpEnableVendorOption属性を使用して有効にすることができます。有効にすると、クライアントは、ベンダー固有の情報オプションをサーバーに要求するためのオプション要求オプションとともに、ベンダー固有の情報オプションをSolicit*メッセージに含めます。DHCPv6 サーバーは、クライアントがオプション要求オプションを介して要求した場合にのみ、ベンダー固有の情報オプションを送信します。
ベンダー固有のデータの送受信を可能にするために、DHCPv6 クライアント モジュールとサーバー モジュールは、ユーザー アプリケーションで実装する必要があるコールバック メソッド プロトタイプを提供します。サンプル アプリケーションには、これらのオプションの送受信を示す基本的な実装が含まれています。含まれるオプション データの最大サポート サイズは、DHCPv6 メッセージごとに 200 バイトです。

8.12.1　ベンダー固有のオプションデータを送信するための共通コールバック関数

DHCPv6 クライアントとサーバーは両方とも、Solicit/Reply メッセージに VendorSpecific Information Option を含めるときに、次のコールバック関数を使用します。

構文：

uint16_t R_DHCPV6_VendorOptionSendCallback(uint32_t enterprize_number、　uint8_t* option_data、*
uint16_t max_option_data_size);

引数:

(出力) *uint32_t* enterprise_number*	IANAに登録されているベンダーの登録エンタープライズ番号
(出力) *uint8_t* option_data*、	受信側の DHCPv6 クライアントおよびサーバー上のベンダー固有のコードによって解釈される、option_data_size オクテットの不透明なオブジェクト。オプション データは、RFC 3315 に従ってフォーマットする必要があります。つまり、DHCP オプション フィールドと同一の形式のコード/長さ/値フィールドのシーケンスとしてエンコードされる必要があります。
(in) *uint16_t max_option_data_size*	option_data バッファに書き込むことができる最大バイト数。

説明：

アプリケーション コールバックは、DHCPv6 ベンダー固有情報オプションに書き込まれるデータを提供します。DHCPv6 クライアントとサーバーは両方とも、ベンダー固有情報オプションを要請/応答メッセージに含めるときに、次のコールバック関数を使用します。この関数が返されると、DHCP メッセージの送信が継続されます。

制限事項:

DHCPv6 クライアント デバイスでは、NWK API を介して DHCPv6 ベンダー固有情報オプションのサポートが有効になっている場合にのみ、この関数が呼び出されます。
にベンダー固有情報オプションの*オプション要求オプション*が含まれている場合にのみ、この関数が呼び出されます。

戻り値:
option_dataバッファに書き込まれたバイト数

8.12.2　DHCPv6サーバでベンダー固有のオプションデータを受信する

DHCPv6 サーバー モジュールはアプリケーション タスク内で実行されるため、クライアントからのメッセージの一部としてベンダー固有情報オプションを受信すると、次のユーザー コールバックを直接実行します。

構文：
void R_DHCPV6_VendorOptionRecvCallback(const r_eui64_t src、uint32_t enterprize_numbaer、const uint8_t* option_data、uint16_t option_len);*
引数:

(in)const *r_eui64_t* src*	DHCPv6クライアント固有識別子（DUID-LL）から取得したEUI-64（リンク層アドレス）
(in) *uint32_t enterprise_number*	IANAに登録されているベンダーの登録エンタープライズ番号
(in)const *uint8_t* option_data*	DHCPv6 ベンダー固有情報オプションからのオプション データ（RFC 3315 に準拠した形式）
(in) *uint16_t option_len*	提供されたoption_dataバッファのサイズ（バイト単位）

説明：

着信 DHCPv6 ベンダー固有情報オプションからのデータを処理するアプリケーション コールバック。

制限事項:

オプション データをこのコールバックの外部または実行後で使用する場合は、必ずコピーする必要があります。提供されたオプション データ バッファーへのその後のアクセスは不正です。

戻り値:

なし

8.12.3　DHCPv6 クライアントでベンダー固有のオプションデータを受信する

DHCPv6 クライアント モジュールは NWK タスクの一部であるため、DHCPv6 サーバーのようにユーザー アプリケーションで定義されたコールバック関数を直接実行しません。代わりに、着信するベンダー固有の情報オプションごとに、タイプ R_NWK_API_MSG_DHCP_VENDOR_OPT_DATA（セクション 0 を参照）の NWK 通知メッセージをアプリケーション タスクに送信します。
*AppIpReceiveMessage関数*内の NWK タスクからのすべての受信指示を処理します。理解を容易にするために、NWK 指示からのデータは、オプションの実際の処理が実装される*R_DHCPV6_VendorOptionRecvCallback*関数に渡されるだけです。したがって、DHCPv6 クライアント モジュールからの受信オプション データは、最終的に、セクション 7.13.2 で説明した DHCPv6 サーバー モジュールからのオプション データを処理するために使用されるのと同じ関数によって処理されます。

8.13 ロギングフレームワーク（RLog）

Wi-SUN FAN プロトコル スタックは、通信ライブラリおよび組み込みシステム用の RLog ロギング フレームワークを利用します。これにより、開発者の利便性とメモリの効率的な使用、コンパイル時間と実行時の柔軟性が組み合わされます。

このフレームワークにより、開発者は一般的な「printf」イディオムを使用できます。RLog コード生成ツールは、ログ記録ステートメントと汎用的で効率的なログ記録ライブラリの間にブリッジを生成し、実行時に最小限の情報、つまりログ記録呼び出しのパラメータの値とオプションでタイムスタンプのみが記録されるようにします。

ログ フレームワークによって生成されたすべてのログ ステートメントは、専用のリング バッファーに内部的に保存され、最も古いログ ステートメントが上書きされます。通常、ログ バッファーのサイズは、プロジェクト設定の R_LOG_BUFFER_SIZE コンパイル時オプションで指定します。このオプションを省略すると、ログ バッファーのデフォルト サイズは 256 バイトになります。

ログ ステートメントは、Rlog API（セクション 7.13.1 で説明）または LOGENTRIES_GETR シリアル API コマンド（セクション 14.1.37 を参照）を使用してデバイスを定期的にポーリングする Wi-SUN UI などの外部ツールを介してアプリケーションによって内部リング バッファから取得できます。ログ情報がログ バッファから取得されるよりも速く書き込まれる場合、ログ情報は内部リング バッファで上書きされる可能性があります。ログ バッファに必要なサイズとポーリング間隔は、ログ フレームワークの詳細度設定によって異なります。詳細度には 4 つのレベルがあり、メモリ消費量（ログ バッファ用）と実行時の情報レベルの間でトレードオフが行われます。

- R_LOG_SEVERITY_OFF (0): ログ記録が無効です。このノードではログステートメントは保存されません。
- R_LOG_SEVERITY_ERR (1): ログステートメントの量が少ない。重大なエラーのみがログに記録されます。
  この詳細度は、ログ バッファーを短い間隔でポーリングできない、長時間実行されるテストに適しています。
- R_LOG_SEVERITY_WARN (2): 中程度のログステートメント。ノードは、ドロップ/拒否/無効なフレームや、
  次のような一時的なエラーなど、予期しないが重大ではないイベントをログに記録します。
  ヒープ メモリが不足しているなど。この詳細レベルは、ログ バッファが小さいデバイスやポーリング間隔が長いデバイスに適しています。
- R_LOG_SEVERITY_DBG (3): 大量のログ ステートメント。ノードは、通常の処理手順を含むすべての
  種類のイベントをログに記録します。この詳細度は開発とデバッグに適していますが、情報の損失
  を避けるために、大きなログ バッファーと短いポーリング間
  隔が必要です。

Wi-SUN FAN UI で使用される補完ツール（RLog デコーダーまたはlogcatと呼ばれる）は、この情報を解析し、コード生成ツールによって記録されたログ記録ステートメントのメタ情報とマージして、従来のテキスト出力またはログ記録データの構造化ドキュメントを提供します。

RLog エンコーダーとデコーダーの互換性をチェックして保証するために、RLog フレームワークは 16 バイトのバージョン ID を使用します。これは基本的に完全なマッピング テーブルのハッシュです。エンコーダーの RLog バージョン ID は、VERBOSE_SETR シリアル API コマンドの確認で返されます。詳細については、このドキュメントの第 14 章を参照してください。

注記（ライブラリ版）：

- RLog ログ ステートメントと対応するデコード情報は、ライブラリ ビルド プロセス中に作成されるため、固定されており、変更できません。Wi- SUN UI でも使用される RLog デコーダー(logcat) は、ライブラリ リリー

ス パッケージの「rlog」ディレクトリにあります。

このログ デコーダーは、この Wi-SUN FAN スタック ライブラリ パッケージ用に特別に構築されているため、他のバージョンの Wi-SUN FAN スタック ライブラリとは互換性がありません。

注記（ソースコードバージョン）:

- RLogコード生成ツールは、サンプルアプリケーションをビルドする前の段階で実行されます。RLogログステートメントと対応するデコードステートメントを生成します。
Wi-SUN FAN UI が接続ノードのバイナリ データを人間が読めるテキストに変換するために必要な情報です。RLog デコーダーは通常読み取り専用であるため、Wi-SUN UI のインストール ディレクトリに自動的にコピーできないことに注意してください。Wi- SUN UI によって使用される RLog デコーダーを自動的に更新するには、対応するコマンド ライン引数を使用して、ログ デコーダーへのパスをソース コード パッケージの「¥Wi-SUN_FAN_RX_FW¥loggen¥dist」ディレクトリに設定する必要があります。詳 細については、Wi-SUN UI のユーザー マニュアルを参照してください。ソース コード パッケージの RLog デコーダーはビルドごとに自動的に更新されるため、Wi-SUN UI の RLog デコーダーも同様に更新されます。新しい RLog デコーダーに切り替えるには、Wi-SUN UI を再起動する必要があることに注意してください

•Windows コマンド ラインから「Wi_SUN_FAN_RX_FW」ディレクトリ内の「RLog-Gen.bat」スクリプトを実行して､RLog ツールチェーンをテストします。
  - oスクリプトはエラーなしで終了するはずです。
  - o「¥loggen」ディレクトリには、プロジェクト用に生成された RLog ヘッダー ファイルが含まれている必要があります。
  - oさらに、「¥loggen」ディレクトリには、生成された RLog デコーダー「logcat」が含まれている必要があります。

  注:
    - 「RLog-Gen.bat」スクリプトはe²studioの実行中に自動的に実行されます。それぞれ IAR Embedded Workbench のビルド前フェーズです。

- 「Wi_SUN_FAN_RX_FW」ディレクトリ内でバッチ スクリプト「RLog-Clean.bat」を実行し、「¥loggen」ディレクトリ内のすべてのファイルを再帰的に削除しますが、ディレクトリ自体は保持します。これにより、e2studio がプロジェクトのインクルード パスから「¥loggen」ディレクトリを削除する問題を回避できます。したがって、このスクリプトを使用して（e2studio プロジェクト エクスプローラーでダブルクリックして実行できます）、プロジェクトからすべての RLog リソースを削除する必要があります。

既知の制限事項：

ログ ステートメントは複数行にまたがってはなりません。たとえば、次のログ ステートメントは無効です。

```
R_LOG_DBG("Remove address registration %{ipv6addr}",
                nbr->ipAddress.bytes);
```

プロジェクト内のすべてのソースファイル（.c）は、一意のファイル名を持つ必要
があります（RLogコード生成に必要）

8.13.1　ロギングフレームワーク API

ロギングフレームワークは、Wi-SUN FANサンプルアプリケーション用のパブリックインターフェースを提供し、ロギングフレームワークのメモリ使用
量を構成できます。これには、

- デバイス上で実行時にログ情報を保存するバッファとそのサイズを指定します
- ログメッセージの重大度しきい値を設定する
- ログを無効または有効にする
- 内部ログバッファからデータを取得する

ロギング フレームワーク API の個々の機能については、次のセクションで説明します。

8.13.1.1 R_LOG_Init

R_LOG_Init	ログフレームワークを初期化してアクティブ化する

構文：

```
void R_LOG_Init(uint8_t* buffer、size_t size、uint8_t severityThreshold);
```

引数:

(in) uint8_t* buffer	ログ データを格納するために使用するバッファ。
(in) size_t size	**バッファのサイズ（バイト単位）。**
(in) uint8_t severityThreshold	ログ ステートメントを記録するための重大度のしきい値。この詳細レベル（同じ詳細度またはそれ以上）を超えるログ ステートメントは、ログ バッファーに記録されます。詳細度が低いログ ステートメントは無視されます。

説明：

この関数は、ログ データを格納するためのリング バッファーとして使用される任意のサイズのバッファーをログ フレームワークに提供するために使用されます。

制限事項:

指定されたログ バッファの所有権はログ記録フレームワークに転送されるため、R_LOG_Init を呼び出した後は、バッファに直接アクセスしたり、バッファを使用したりすることはできません。

戻り値:

なし。

8.13.1.2 R_LOG_SetSeverityThreshold

R_LOG_SetSeverityThreshold	現在の重大度しきい値を設定する

構文：

```
void R_LOG_SetSeverityThreshold(uint8_t severityThreshold);
```

引数:

(in) uint8_t severityThreshold	ログ ステートメントを記録するための重大度のしきい値。この詳細レベル（同じ詳細度またはそれ以上）を超えるログ ステートメントは、ログバッファーに記録されます。詳細度が低いログ ステートメントは無視 されます。

説明：

この関数は、将来のログ メッセージのしきい値を変更しますが、すでに保存されているメッセージは変更しません。つまり、既存のログ ステートメントはログ バッファーから削除されません。

制限事項:

この関数は、ログ バッファー内の既存のログ ステートメントの詳細レベルが新しいseverityThreshold より低い場合でも、それらの既存のログ ステートメントを削除しません。

戻り値:

なし。

8.13.1.3　R_LOG_GetSeverityThreshold

R_LOG_GetSeverityThreshold	

構文：

uint8_t R_LOG_GetSeverityThreshold(void);

引数:

なし。

説明：

この関数は、ログ バッファーに記録されるログ ステートメントの現在設定されている重大度しきい値を返します。この詳細レベル（同じ詳細度またはそれ以上）を超えるログ ステートメントは、ログ バッファーに記録されます。詳細度が低いログ ステートメントは無視されます。

制限事項:

なし。

戻り値:

uint8_t	ログ ステートメントを記録するための重大度のしきい値。この詳細レベル（同じ詳細度またはそれ 以上）を超えるログ ステートメントは、ログ バッファーに記録されます。詳細度が低いログ ステートメントは無視されます。

8.13.1.4　R_LOG_CurrentSize

R_LOG_CurrentSize	ログバッファ内の現在のデータ量を取得する

構文：

```
size_t R_LOG_CurrentSize(void);
```

引数:

なし。

説明：

この関数は、ログ バッファーに書き込まれたログ データの現在のサイズを返します。

制限事項:

戻り値がログ バッファーの容量と等しい場合、内容が上書きされているかどうかはわかりません。

戻り値:

size_t	ログ バッファーに書き込まれたログ データの現在の量（バイト単位）。

8.13.1.5 R_LOG_GetLog

R_LOG_GetLog	

構文：

size_t R_LOG_GetLog (uint8_t * dest, size_t size)

引数:

(out) uint8_t* dest	ログ バッファーからバイトをコピーする宛先。
(in) size_t size	コピーする最大バイト数

説明：

この関数は、ログ バッファーから最大 size バイトをdestにコピーし、コピーしたバイト数をログ バッファーから削除します。

制限事項:

この関数は、ログ ステートメント間の区切り文字を考慮せずに、現在のログ バッファーから可能な限り多くのバイトをコピーします。そのため、コピーされたログ レコードが完全であるという保証はありません。

戻り値:

size_t	destにコピーされたバイト数(<= size)、またはバッファの末尾の 0。

8.14 ネットワークからデバイスを削除する

ネットワーク内のデバイス管理を容易にするために、以前参加したデバイスをネットワークから削除することができます。対応するデバイスのソフトウェア リセットとは対照的に、このアプローチでは、デバイスがネットワークから離れようとしていることを他のネットワーク参加者に通知し、適切に対応して（一時的な）ネットワーク障害や不整合を回避できるようにします。

8.14.1 ネットワークを離れる（ルータノード）

ルータ ノードでは、セクション 10.1.17 で説明されているように、アプリケーションからネットワークを離れるには NWK API 関数を実行するだけで十分です。この機能により、ルータ ノードはすぐにネットワークを離れ、リセット状態（参加状態 0）に遷移します。

8.14.2 ネットワークからデバイスをキックする（ボーダールータ）

ルータ ノードは、ボーダールータによってネットワークから削除されることもあります。このため、提供されているサンプル アプリケーションは特別な「Kick」メッセージを定義します。これは、ボーダールータからネットワークから削除する必要があるルータ ノードに送信される、既知のポートとペイロードを使用する UDP データグラムです。たとえば、サンプル アプリケーションはこのメッセージに UDP ポート 3611 と 0xABCDEF のペイロードを使用します。UDP ポートとペイロードは、ネットワーク内のすべてのデバイスで同じであり、UDP ポートが Wi-SUN FAN 操作に必要なポート（DHCPv6 や EAPOL リレーなど）と競合しない限り、どちらもカスタマイズできます。Kick メッセージを受信すると、ルータ ノード上のサンプル アプリケーションは対応する UDP データ表示を処理し、ペイロードが期待どおりであることを確認し、ネットワークを離れる NWK API 関数を実行します（セクション 10.1.17 を参照）。そのため、動作はルータ ノードが自発的にネットワークを離れる場合と同じです。ボーダールータでは、離脱デバイスのソース ルートの削除は、ETX エポックの次の有効期限が切れたときに実行されます。これには最大 60 秒かかる場合があります。

8.15 CSMP によるネットワーク管理

CoAP シンプル管理プロトコル (CSMP)は、以前はシスコ独自のリモート ネットワーク管理プロトコルであり、IoT ネットワーク デバイスを含む帯域幅が制限された大規模なネットワークを対象としています。CSMP は、ネットワーク管理システム (NMS)とも呼ばれる中央サーバーを使用します。このサーバーには、ネットワーク内のすべてのデバイスを登録する必要があります。NMS は、監視とトラブルシューティングのために、ネットワーク デバイスに関するさまざまなメトリックと診断を収集して維持するだけでなく、ファームウェア更新などの管理機能も提供します。

2024年頃、シスコはCSMPを一般公開することを決定し[13]、リファレンス実装を提供した。
Raspberry Pi上で動作するcsmp-agent-lib [11]と呼ばれるCSMPクライアント用の
オープンプロトコルとシスコではOpenCSMPとも呼ばれる実装。ルネサスは、
csmp-agent-libのスナップショット(コミット ID 593d1acc0f4cb16a163b6fc5216a16ce5aa10280) により、Renesas Wi-SUN FAN サンプ ル アプリケーションは、すべてのターゲット プラットフォーム (リソース制約のため RL78/G1H を除く) で CSMP をサポートできるようになります。次のサブセクションでは、Renesas Wi-SUN FAN サンプル アプリケーションでの CSMP の構成と使用方法について詳しく説明します。

統合されたcsmp-agent-libコードとルネサスのカスタマイズはまだ開発中であり、将来のリリースで実装される特定の機能が欠けていることに注意してください。利便性に加えて
および使いやすさの問題により、現在の Wi-SUN FAN アプリケーションでは CSMP に次の機能制限が適用されます。

- 一部のメッセージ (TLV)はまだ実装されていません
  - o 各ノードのFND概要では、一部の情報フィールドが空または欠落している場合があります。
- ファームウェアの更新と管理は不可能
- FNDの「メトリックの更新」アクションが機能しない

NMSは外部サーバ (Wi-SUNネットワークの一部ではない)であるため、図に示すようにWi-SUNノードとNMS間の通信を可能にするために境界ルータゲートウェイ (セクション7.1.6および[12]を参照)が必要である。
図7-7. 以下のセクションでは、境界ルータゲートウェイが[12]で説明されているように設定されていることを前提としています。
境界ルータはこのゲートウェイに接続され、Wi-SUN ネットワーク全体の外部サーバーとの通信が可能になります。

Wi-SUN FANデバイスのNMSとしては、シスコのIoTフィールドネットワークディレクター (FND)が使用できます (テスト済みバージョン： 4.6.1-61) 。FNDはシスコ製品であるため、このドキュメントではインストールとセットアップについては説明しません。シスコは「CSMP開発者チュートリアル」を提供しています (GitHubリポジトリのdocs フォルダ内[11]) 。
これには、IoT FND サーバーのセットアップに関する手順と、評価および開発目的で事前構成された IoT FND を備えた仮想マシンを取得するための連絡先メール アドレスが含まれており、これが推奨される開始点です。残りのセクションでは、IoT FND が「CSMP 開発者チュートリアル」のシスコの手順に従ってインストールおよびセットアップされており、Renesas Wi-SUN ノードが CSV ファイルのインポートによって EUI-64 とともに FND インスタンスに追加されていることを前提としています。これはシスコのドキュメントにも記載されています。図 7-11 は、次の CSV ファイルによって 3 つの Renesas ノードがインポートされた後の FND の初期デバイスページを示しています (ノードはまだ起動されていません)。

eid、デバイスタイプ、機能
74905000000000000、cgmesh、メートル
74905000000000001、cgmesh、メートル
74905000000000002、cgmesh、メートル

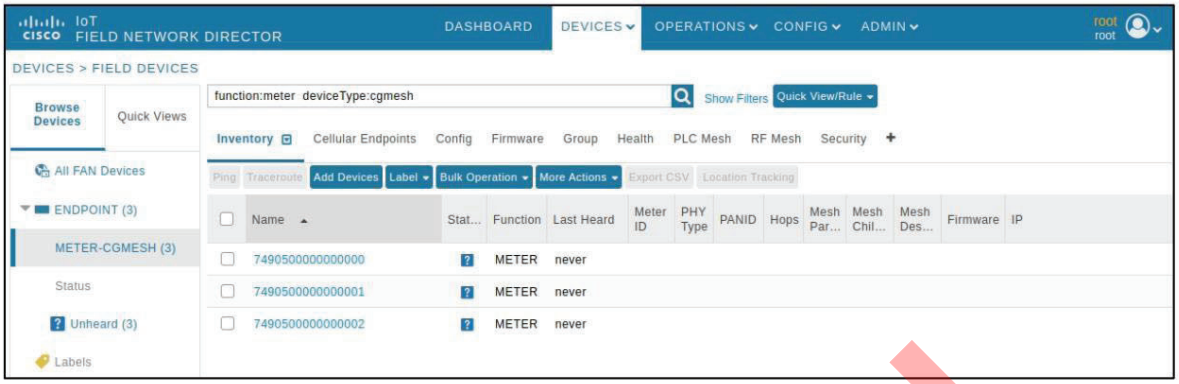


図8-11 FND内の未登録Wi-SUNノード

8.15.1　サンプルアプリケーションでの CSMP の構成

Wi-SUN FANサンプルアプリケーションでは、アプリケーションのメモリ使用量を減らすために、CSMPはデフォルトで無効になっていま す。アプリケーションをビルドする前に、コンパイル時オプションR_CSMP_ENABLEDを1に設定することで有効にすることができます（セクション7.2を参照）。このコンパイル時オプションでCSMPを有効にすると、各ノードは

完全に動作可能になると（つまり、参加状態 5 に達すると）、NMS への登録を自動的に試行します。

このため、各ノードを起動する前に、CSMP サーバーの IPv6 アドレスと登録の最小間隔および最大間隔（注: 登録はトリクル タイマーに基づいています）を設定する必要があります。これは、ノードを起動する前に（Wi-SUN UI のマニュアルに記載されているように、［シリアル コマンドの送信］ボタンを使用して）Wi-SUN UI 経由で

CSMPCONF_SETRコマンド（セクション 14.1.47 を参照）を送信することで行われます。次のコマンド例では、ノードでCSMP サーバー アドレスをfd00:cafe:cafe::cccc、最小登録間隔を10 秒、最大登録間隔を120 秒に設定します。

CSMPCONF_SETR 01 FD00CAFECAFE0000000000000000000000CCCC 000A 0078

Wi-SUN UI を介してノード上で CSMP 構成を正常に設定した結果を図 8-12 に示します。ノードが「シリアル API コマンドが見つかりません!」と応答した場合、このノードで実行されているファームウェアで CSMP が有効になっていません。

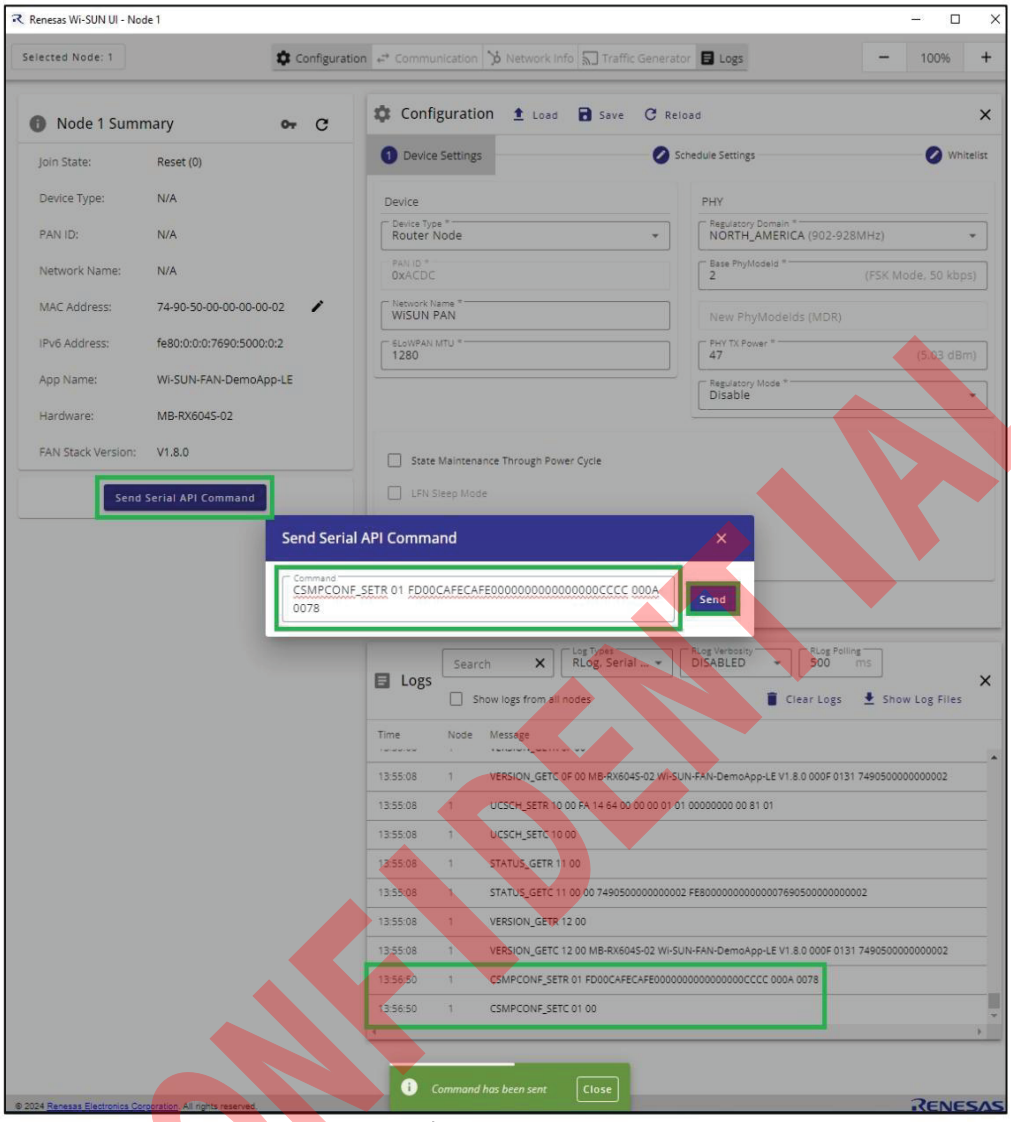


図8-12 Wi-SUN UIを介したノード上のCSMP構成の

8.15.2　IoT FNDにおけるWi-SUNノードの管理

CSMP 対応の Wi-SUN デバイスが起動し（セクション 7.15.1 で説明）、動作可能（参加状態 5）になると、FND のデバイス リストは図 7-13 に示す ように変わります。つまり、動作中のデバイスのステータスが「未認識」（青）から「稼働中」（緑）に変わり、ノードの PAN ID、ファームウェア バージョン、IPv6 アドレスなどの情報も表示されます。

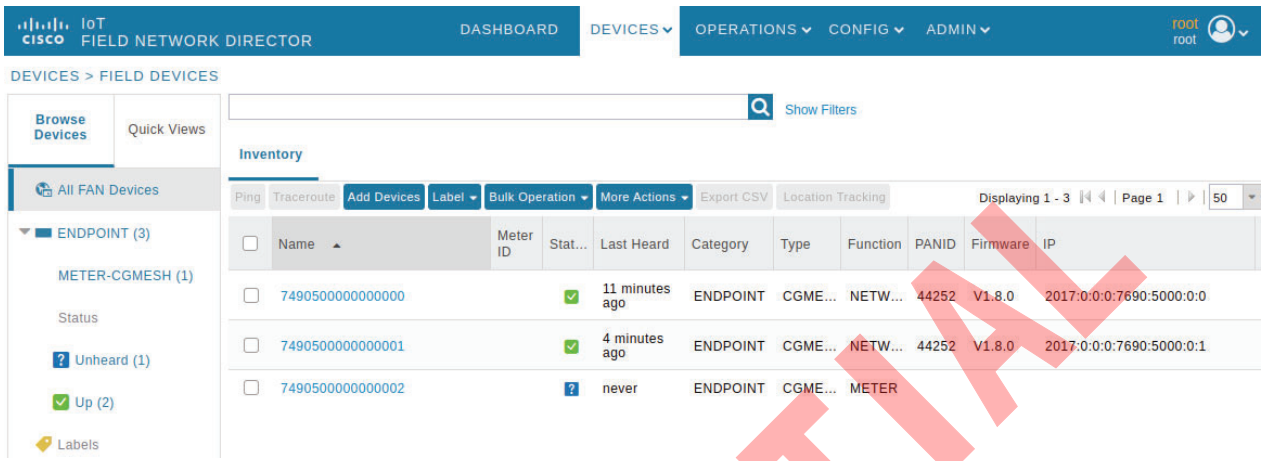


図8-13 FNDに登録されたWi-SUNノード

Wi-SUN ノードの FND への登録が期待どおりに機能しない場合、最も可能性の高い原因は IPv6 通信の問題であるため、最初のトラブルシューテ ィング手順は、FND PC から Wi-SUN ネットワーク内の稼働中のルータ ノードに ICMPv6 エコー要求を送信し、その逆も行うことです。この ping が失敗する（エコー応答が返されない）場合は、 Wireshark やtshark（Wireshark のコマンド ライン バージョン）などのパケット キャプチャ ツールを使用してパケット フローをトレースし、パケットが失われた場所を特定する必要があります。パケットは、ファイアウォール アプライアンスまたは無効な IPv6 ルートが原因でドロップされる可能性があります。以前のテストベッド セットアップでは、IoT FND を実行している仮想マシンとの間の IPv6 通信は、IPv6 ルートが壊れているか欠落しているために不安定になることがよくありました。解決策は、BR ゲートウェイ（Raspberry Pi）経由で FND（VM）の Wi-SUN ネットワークに静的ルートを追加し、ホスト システム経由で BR ゲートウェイ（Raspberry Pi）の FND（VM）に静的ルートを追加することでした。

VM を実行します。

登録されているデバイスの名前をクリックすると、FND は対応するノードのデバイス情報ページを表示します。このページには、デバイスによって報告された大量の情報が含まれています。CSMP の実装はまだ開発中であるため、これらの情報フィールドの一部は空または欠落している可能性がありますが、将来のリリースでサポートされる予定です。表示される情報は、CSMP プロトコルで定義された TLV の一部として Wi-SUN によって送信 され、CSMP の RFC ドラフト [13] またはcsmp-agent-libコードで定義されています。このリリース パッケージには、CSMP TLV を作成するために Wi-SUN FAN サンプル アプリケーションとプロトコル スタックからのどの情報が使用されるかを示すスプレッドシート [14] も含まれています。

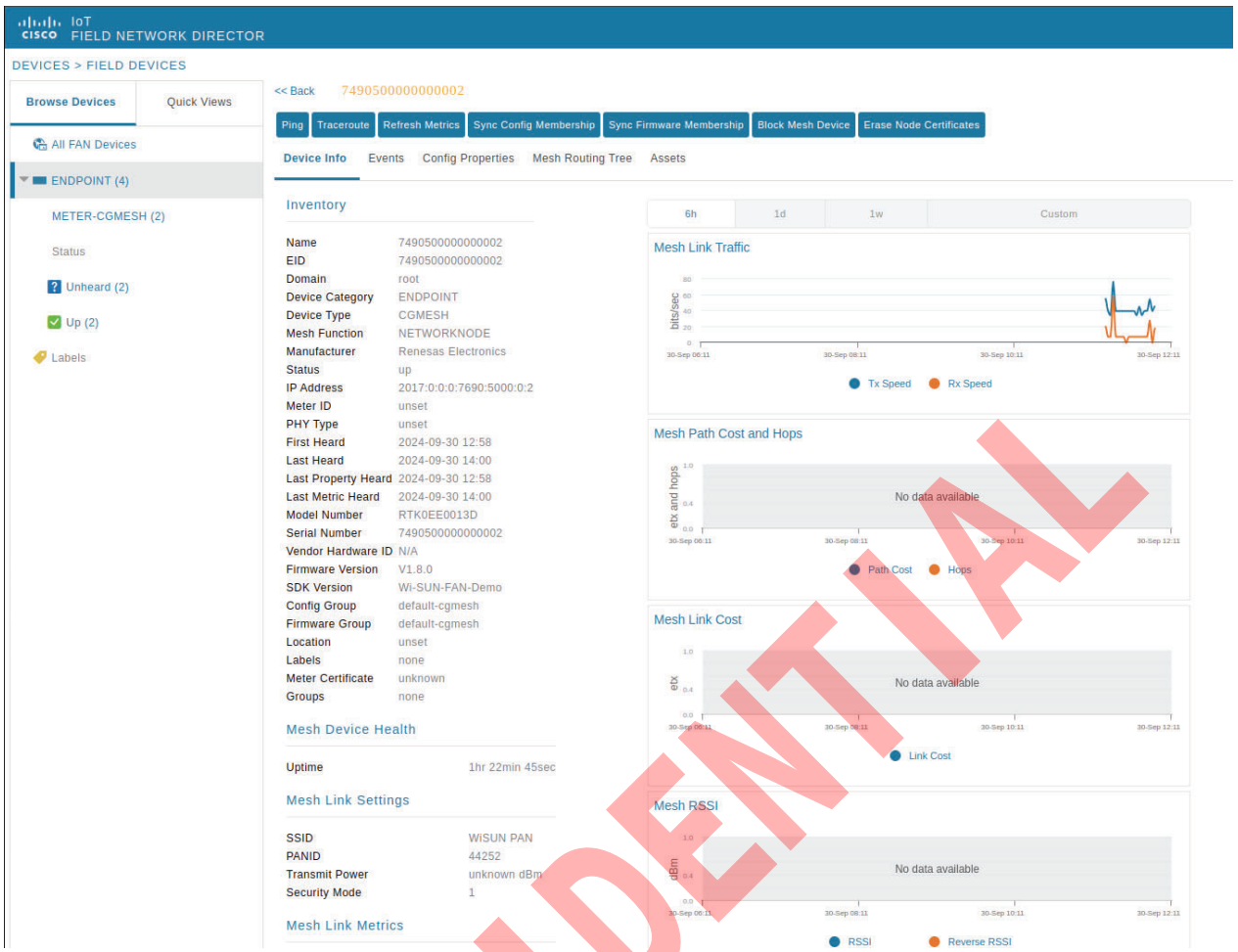


図8-14 FNDのWi-SUNルータに関するデバイス情報

ステータス情報に加えて、FND は各デバイスへの IPv6 接続をいつでも確認することもできます。

デバイス情報またはデバイスページの左上にある[Ping]ボタンをクリックすると、FND はそれぞれの Wi-SUN ノードに ICMPv6 エコー要求を送信し、ノードからの対応するエコー応答メッセージを待機します。結果は、図 8-15 に示すようにポップアップに表示されます。

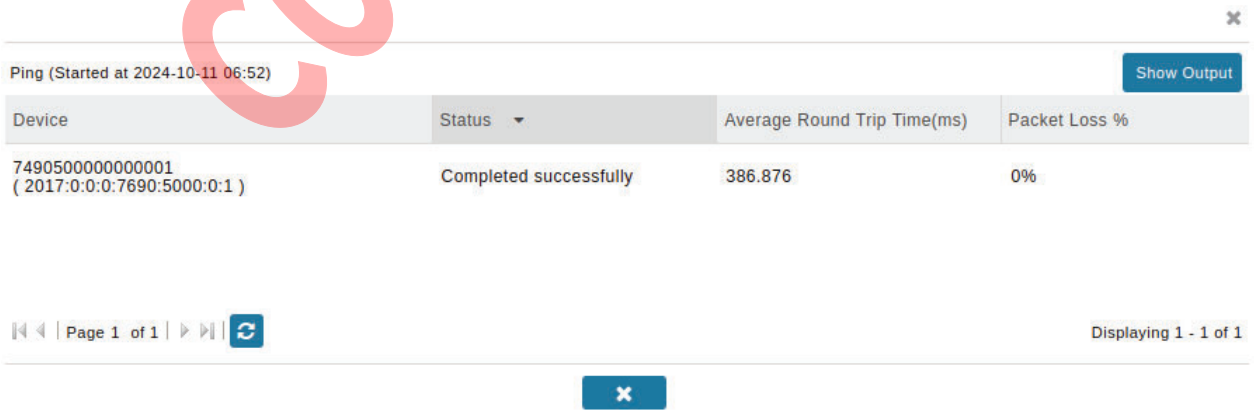


図8-15 FNDからルネサスWi-SUNルータへのping成功

8.16 生のソケットのサポート

Wi-SUN FAN プロトコル スタックは、raw ソケットと同様の機能を提供します。この動作は、IP データ表示（ *R_NWK_API_MSG_IP_DATA_IND* ）を使用して模倣され、これにより、アプリケーション層による raw IPv6 パケットの処理が可能になります。セクション 0 では、IP データ表示の構造と内容について説明します。

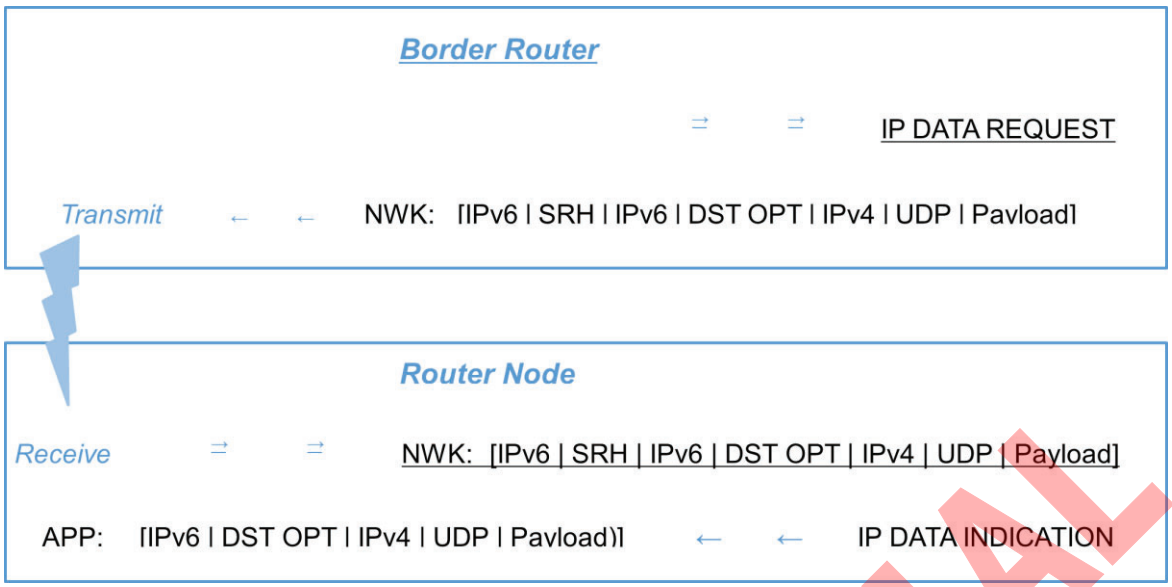
次の場合、着信 IPv6 パケットは IP データ表示としてネットワーク層からアプリケーション層に転送されます。

- BR のみ: 宛先 IPv6 アドレスが、Wi-SUN ネットワークで使用される IPv6 ネットワーク プレフィックス範囲内ではありません。
- *Next Header フィールド)*の処理中に、サポートされていない IPv6 拡張ヘッダーが検出されました。サポートされていない拡張ヘッダーとは、次の値と異なる*Next Header値*のことです。

  R_IPV6_NEXT_HDR_HOP_BY_HOP = 0x00 //!< ホップバイホップオプション拡張ヘッダー
  R_IPV6_NEXT_HDR_UDP = 0x11 //!< UDP ヘッダー
  R_IPV6_NEXT_HDR_IPV6 = 0x29 //!< IPv6 ヘッダー
  R_IPV6_NEXT_HDR_ROUTING = 0x2B //!< ルーティング ヘッダー
  R_IPV6_NEXT_HDR_AH = 0x33 //!< 認証ヘッダー
  R_IPV6_NEXT_HDR_ICMPV6 = 0x3A //!< ICMPv6 ヘッダー
  R_IPV6_NEXT_HDR_DST_OPTIONS = 0x3C //!< 宛先オプション
  R_IPV6_NEXT_HDR_NONE = 0x3B //!< 次のヘッダーなし
- RNのみ:
  ルータノードは外側のIPv6ヘッダーの宛先である
  そして
  ボーダールータは外側のIPv6ヘッダーのソースです
  そして
  ルータノードは内部 IPv6 ヘッダーの宛先ではありません

以下の例は、IPv6 でトンネルされた IPv4 パケットの送信を示しています。パケットは、ボーダールータで IP データ要求を実行することによって開始されます。生の IPv6 パケットには、IPv6 ヘッダー、宛先オプション、IPv4 ヘッダー、UDP ヘッダー、およびペイロードが含まれます。対応する IPv6 およびソース ルーティング ヘッダーはネットワーク層によって追加され、フレームはボーダールータによって送信されます。

宛先（ルータ ノード）でフレームを受信すると、ネットワーク層はアプリケーション層に IP データ表示を開始します。これは、IPv4 ヘッダーが Wi-SUN FAN プロトコルでサポートされていないためです（上記参照）。したがって、この IPv6 パケットのそれ以上の処理は Wi-SUN FAN プロトコル スタックの範囲外であり、IP データ表示を使用してアプリケーションによって処理される必要があります。着信 IPv6 パケットでサポートされていないヘッダーは常に IP データ表示となるため（上記参照）、この例は逆方向でも機能します。つまり、ルータ ノード上のアプリケーションから IP データ要求を開始します。



8.17 Wi-SUN FAN1.0 レガシーサポート

名前に「FFN-Core」という用語が含まれる新しいビルド構成/プロジェクトは、以前の FAN1.0 プロトコル標準と下位互換性があります。

FAN1.0 プロトコル標準との完全な下位互換性を確保するには、次の NWK 属性を適宜変更してください。

NWK 属性	説明	値の変更
R_NWK_lfnサポートグローバル	このボーダールータの PAN で LFN をサポートする場合は True。BR が LFN 関連の IE を送信しない場合は False。	誤り (0)
R_NWK_lfnサポートローカル	このデバイスが LFN の FFN 親になれる場合は True。LFN からの参加要求 (LPAS) を無視する場合は False。	誤り (0)
R_NWK_fanTPSVersion	この属性は、ノードがどの FAN TPS バージョンに認定されているかを示します。 1 = FAN 1.0 認定 2 = FAN 1.1 認定	1
R_NWK_mplシードIDをMplオプションに追加	有効にすると、FANの要求に応じて、各送信MPLメッセージのMPLオプションにMPLシード識別子が追加されます。 1.0。 デフォルトでは、シードIDはFAN 1.1の推奨に従って省略されます。 0 = シードID省略 1 = シードIDを含む	1

第9章 Wi-SUN FAN ブートストラップフロー

このセクションでは、ボーダールータと参加ルータ ノード間の Wi-SUN FAN ブートストラップ フローについて説明します。これには、メッセージ フロー、参加状態の遷移、およびデバイスを動作させてデバイスの状態を確認するための対応するシリアル API 呼び出しが含まれます。

通常、ボーダールータまたはルータ ノードが Wi-SUN FAN ネットワークで動作できるようにするには、対応するデバイスを適切に構成する必要があります。シリアル API は、この目的のために専用の機能を提供します。

初期化は簡素化されており、実行する必要がある主要な手順は 3 つだけです。

ステップ	*関数*	*説明*	*シリアルAPIコマンド*
1.	デバイス構成	この手順では、ボーダールータまたはルータ デバイスとしてデバイス タイプを構成することと、PanID、ネットワーク名などの FAN ネットワーク関連のパラメータについて説明します。	*DEVCONF_SETR*
		↓	
2.	PHY 構成	このステップでは、規制ドメイン、動作モード、使用するTX電力を指定してPHY構成を行います。	*PHYFN_SETR*
		↓	
3.	スケジュール構成	このステップでは、ユニキャストおよびブロードキャスト スケジュールの構成について説明します。固定チャネルまたは周波数ホッピング モードでの動作、間隔タイミング、チャネル マスキングなどを定義します。	*UCSCH_SETR* *BCSCH_SETR*

*STARTR*を実行するだけで、Wi-SUN FAN デバイスの動作が開始されます。

	デバイスの開始	デバイスの操作を開始します。ボーダールータは、動作状態、参加状態 5 で直接起動します。これにより、ルータ ノードは参加状態 1 (PAN を選択) で起動し、動作状態、参加状態 5 に到達するまでさまざまな参加状態を進みます。	STARTR

ルータ ノードのステータスの変更は、シリアル API コマンド***JSIと同じ参加状態表示によって自動的に報告されます。***さらに、ネットワーク参加プロセス中のルータ ノードの進行状況とステータスは、シリアル API コマンド STATUS_GETR を実行することによって取得できます。

	Join Sttate 表示 Device状態確認	デバイスの現在の状態を表示/取得します: ・ Reset State 0 ・ Join State 1: Select PAN ・ Join State 2: Authenticate (Note) ・ Join State 3: Acquire PAN Config ・ Join State 4: Configure Routing ・ Join State 5: Operational	JCI STATUS_GETR

(注: 証明書の処理には RL78/G1H デバイス上で膨大な計算能力が必要となるため、Join State 2 ではシリアル API コマンドの処理が制限される可能性があります)

ルータ ノード デバイスが Join State 5 に到達すると、Wi-SUN FAN ネットワークに完全に参加し、完全に動作します。Join State 5 に到達する前は、ICMP または UDP メッセージの送信は完全にはサポートされていないことに注意してください。次の図は、ブートストラップ フローを詳細に示しています。シリアル API コマンドと対応する NWK API 呼び出しの詳細については、このドキュメントの第 14 章も参照してください。

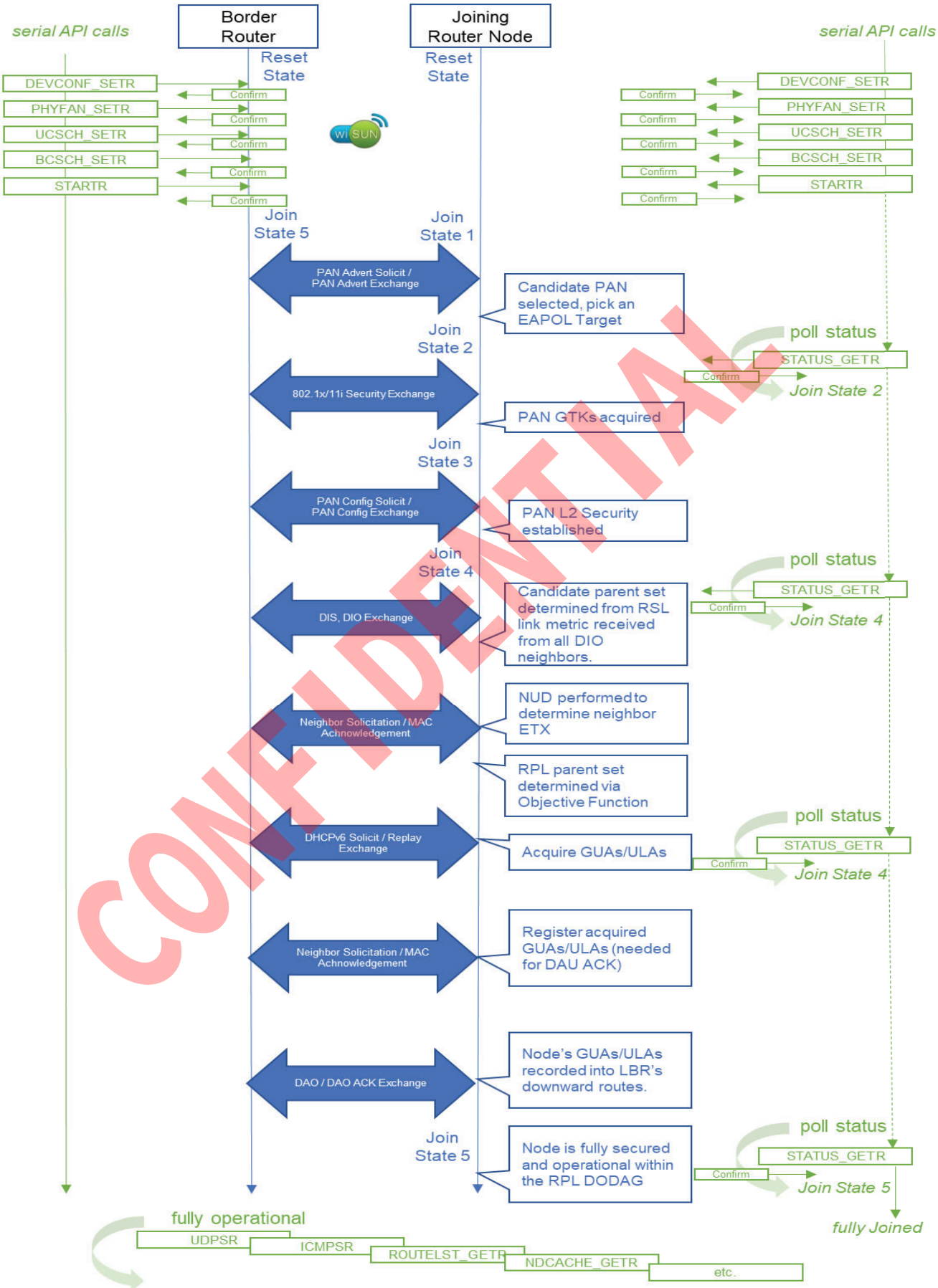


図9-1 FFNブートストラップフロー

9.1 ディフォルト設定

すべての Wi-SUN ノードには、Wi-SUN FAN プロトコル スタックとデバイス自体の既知の有効な構成パラメータの完全なセットがあり、電源サイクルごとに読み込まれます。これにはいくつかの利点があります。

- 各ノードは最初から動作構成を持っており、追加の設定なしで起動できます。構成（デフォルト構成が適切であると仮定）
- デフォルト設定を使用するノードは相互運用可能であり、追加の設定なしで相互に通信し、同じネットワークに参加できるため、デモンストレーション用の簡単なセットアップが容易になります。
- サンプル アプリケーションのコードには、デバイスのデフォルト構成パラメータを含む単一のモジュール（ヘッダー ファイル）があり、さまざまなユース ケースに簡単に適応できます。これは、自動起動機能の使用を目的としたデバイスにも役立ちます（セクション 9.2 を参照）。

デフォルトのデバイス構成パラメータは次のとおりです。

Network Name	“WiSUN PAN”
6LoWPAN MTU	1280 Bytes
PAN ID	0xACDC
PHY Mode (Fixed Channel)	North America, FSK (w/o FEC), 50kbps (m=1.0), Channel 0
PHY Mode (Frequency Hopping)	North America, FSK (w/o FEC), 150kbps (m=0.5), 64 Channels
State Maintenance Among Power Cycle	Disabled
RLog Verbosity	R_LOG_SEVERITY_ERR (1)
Regulatory Mode	Disabled
MDR	Disabled
TX Power	+5 dBm (including any FEM gain)

デフォルトの構成パラメータは、ヘッダー ファイルr_app_fan_config_defaults.hにあり、必要に応じて変更できます。このファイルには、のすべての FFN 構成パラメータが含まれています。一部の構成パラメータは相互に依存していることに注意してください。たとえば、規制を変更すると、通常、動作クラスまたはチャネル プラン ID も変更する必要があります。構成オプションとその意味、要件、または考えられる依存関係の詳細については、Wi-SUN UI または FANTPS のユーザー マニュアルの第 8 章を参照してください。

デフォルト設定の各パラメータは、設定全体が有効な限り、必要に応じて調整できます。r_app_fan_config_defaults.hの冒頭には、複数の設定パラメータを変更することで機能全体を有効にするさまざまな設定オプションがあります。

R_APP_CFG_MDR_ENABLED は、北米で必要なすべての OFDM PHY モードを新しいモードとして自動的に設定するため、これらのモー ドはいずれも、MDR を使用した実行時にノードで使用できます。たとえば、デバイスタイプを R_ROUTERNODE から R_BORDERROUTERに変更します。

9.2 自動起動

開発およびテスト中は、Wi-SUN UI などの外部ツールを使用して Wi-SUN デバイスを対話的に構成することが適切です。実稼働環境では、デバイスは電源が供給されると自動的に構成され、起動される必要があります。これは、プロジェクト設定でR_APP_AUTO_STARTプリプ ロセッサ ディレクティブを定義することで、提供されている Wi-SUN FAN サンプル アプリケーションで簡単に実現できます。このシンボ

ルが定義されている場合、サンプル アプリケーションは、通常はノードが STARTR シリアル コマンド (14.1.15 を参照) を受信したときに実行される電源サイクル後に、Wi-SUN FAN プロトコル スタックとアプリケーション タスクの関連モジュール (Wi-SUN FAN 認証モジュールなど) を初期化して起動するコードを自動的に実行します。このように、Wi-SUN ノードは、r_app_fan_config_defaults.hで提供されている構成を使用して電源が接続されると、自動的に完全に起動します。この構成は、セクション 9.1 で説明されているように、展開前に調整できます。

自動起動機能は、R_APP_CFG_DEVICE_TYPE、 R_APP_CFG_FREQ_HOP_ENABLED、 R_APP_CFG_MDR_ENABLEDなどのプロジェクト構成の他のシンボルと組み合わせることもできます。

第10章 NWK APIの説明

このセクションでは、Wi-SUN FAN NWK API について詳しく説明します。NWK API を使用すると、独自の Wi-SUN FAN ネットワークを構築し、標準のネットワーク機能を利用することができます。すべてのプリミティブは対応する関数にマップされ、コンテンツは専用の構造体に格納されます。指示および確認プリミティブは、コールバック関数として実現されます。要求構造体は、対応する関数を呼び出すことによって要求キューに追加されます。

10.1 共通API関数

次のセクションでは、Wi-SUN FAN プロトコル スタックの標準およびモデム構成でサポートされる共通 API 関数について説明します。

10.1.1 R_NWK_Init

R_NWK_Init	ネットワーク層を初期化する

構文：

r_result_t R_NWK_Init(void)

引数:

なし

説明：

この関数は、ネットワーク層とその下の層を初期化して開始します。ネットワーク スレッドと MAC スレッドが開始され、R_RESULT_SUCCESS の値が返されます。

制限事項:

なし

戻り値:

R_RESULT_SUCCESS	初期化が正常に完了しました

10.1.2 R_NWK_ResetRequest

R_NWK_ResetRequest	ネットワーク層と下位層をリセットする

構文：

r_result_t R_NWK_ResetRequest(void)

引数:

該当なし

説明：

この関数は、ネットワーク層とそれ以下の層にリセットを開始します。プリミティブID R_NWK_API_MSG_RESET_REQを持つ空のリクエスト構造をネットワーク層のメールボックスにキューイングします。正常に追加された場合、関数はR_RESULT_SUCCESSの値を返します。ネットワーク層のキューに空きがない場合は、関数はR_RESULT_SUCCESSの値を返します。
R_RESULT_INSUFFICIENT_BUFFER。リセット要求のキューへの登録が成功すると、リセットが処理されるまで、つまり対応する確認が発行されるまで、キューはそれ以上の要求に対して閉じられます。

制限事項:

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。 一時停止モード
R_RESULT_INSUFFICIENT_BUFFER	パケットがキューに追加されませんでした。メッセージ バッファを割り当てるためのメモリが足りません。

10.1.3 R_NWK_StartRequest

R_NWK_StartRequest	ネットワークを作成して開始する

構文：

```
r_result_t R_NWK_StartRequest(     uint16_t panId,
                                   uint16_t panSize、
                                uint8_t useParent_BS_IE、
                                const char* p_networkName、
                                const r_ipv6addr_t* ipAddr、
                                   uint16_t option)
```

引数:

(in) uint16_t panID	作成するネットワークの 16 ビット PAN 識別子。
(in) uint16_t panSize	PAN サイズの 16 ビットの事前初期化値。通常はゼロに設定されます。
(in) uint8_t useParent_BS_IE	ノードが親の BS-IE に従ってブロードキャストをリッスンする必要があることを示すには、8 ビットの Use Parent BS-IE を 1 に設定する必要があります。
(in) const char **p_networkName*	ネットワーク名を保持する NULL で終了する文字列へのポインター。ネットワーク名でサポートされる最大サイズは 32 文字です。
(in) const r_ipv6addr_t ipAddr* 、	ボーダールータが使用する一意の IP アドレス (GUA/ULA)。最初の 8 バイトは、RPL のネットワーク プレフィックスも定義します (プレフィックスは /64 に固定されます)。
(in) uint16_t options	ネットワーク層への16ビットオプション。ブロッキングを強制的に設定してください。これを行うには、オプションパラメータのビット8をクリアしてください。例：*オプション &= (~R_OPTIONS_NON_BLOCKING);*

説明:

R_NWK_StartRequest 関数を呼び出すと、Wi-SUN FAN ネットワークが作成され、開始されます。この関数は、プリミティブ ID R_NWK_API_MSG_NWK_START_REQ を持つ要求構造をネットワーク層メールボックスにキューイングします。正常に追加され、実行された場合、関数は R_RESULT_SUCCESS の値を返します。ネットワーク層キューに空きがなかった場合、関数は R_RESULT_FAILED の値を返します。

制限事項:

この機能は、ボーダールータ構成でのみサポートされます。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。一時停止モード
R_RESULT_INVALID_PARAMETER	無効なパラメータまたはヌルポインタが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージ バッファが利用できません

10.1.4 R_NWK_JoinRequest

R_NWK_JoinRequest	既存のネットワークに参加する

構文：

*r_result_t R_NWK_JoinRequest(const char *p_networkName,*
uint16_t options)

引数:

(in) const char *p_networkName*	ネットワーク名を保持する NULL で終了する文字列へのポインター。ネットワーク名でサポートされる最大サイズは 32 文字です。
(in) uint16_t options	ネットワーク層への16ビットオプション。ブロッキングを強制的に設定してください。これを行うには、オプションパラメータのビット8をクリアしてください。例：*オプション &= (~R_OPTIONS_NON_BLOCKING);*

説明：

R_NWK_JoinRequest 関数を呼び出すと、既存の Wi-SUN FAN ネットワークへの参加プロセスが開始されます。この関数は、プリミティブ ID R_NWK_API_MSG_NWK_JOIN_REQ を持つ要求構造をネットワーク層メールボックスにキューに入れます。正常に追加され実行された場合、関数は R_RESULT_SUCCESS の値を返します。ネットワーク層メールボックスに空きがなかった場合、関数は R_RESULT_INSUFFICIENT_BUFFER の値を返します。

制限事項:

この機能は、ルータ ノード構成でのみサポートされます。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。 一時停止モード
R_RESULT_INVALID_PARAMETER	無効なパラメータまたはヌルポインタが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージ バッファが利用できません

10.1.5 R_NWK_GetRequest

R_NWK_GetRequest	パーソナルエリアネットワーク情報ベースから値を読み取る

構文：

r_result_t R_NWK_GetRequest(uint8_t attrId,
*void *p_attrValue、*
uint16_t attrLength)

引数:

(in) *uint8_t attrId*、	読み取る PIB 属性の 8 ビット識別子。 対応するIDは「r_nwk_attr_id_t」列挙型によって定義されます。このドキュメントの第11章も参照してください。
(out) **p_attrValue*	読み取られた値を保持する属性へのポインター。
(in) uint16_t attrLength	読み取る PIB 属性の 16 ビットの長さ。

説明：

この関数は、PIB から値を読み取ります。プリミティブ ID R_NWK_API_MSG_GET_REQ を持つ要求構造を要求キューに追加します。正常に追加された場合、関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、関数は R_RESULT_FAILED の値を返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。 一時停止モード
R_RESULT_INVALID_PARAMETER	無効なパラメータまたはヌルポインタが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージ バッファが利用できません

R_RESULT_UNAVAILABLE	要求された属性/値は（まだ）利用できません。これは、いつでも失われる可能性がある値、または最初から不明な値（例: RPL 親情報または RN のブロードキャスト スケジュール）に対する一時的なエラーです。

10.1.6 R_NWK_GetRequestMultipart

R_NWK_GetRequestMultipart	パーソナルエリアネットワーク情報ベースからより大きな属性値を読み取る

構文：

r_result_t R_NWK_GetRequestMultipart(uint8_t attrId,
*void *p_attrValue、*
uint16_t attrLength、
r_boolean_t proceedPrevious)

引数:

(in) *uint8_t attrId*、	読み取る PIB 属性の 8 ビット識別子。 対応　するIDは、「r_nwk_attr_id_t」列挙子タイプ。このドキュメントの第 11 章も参照してください。
(out) *void *p_attrValue*	要求された属性値を保持する出力バッファへのポインタ。
(in) *uint16_t attrLength*	出力バッファのサイズ。
(in) *r_boolean_t proceedPrevious*	このリクエストが最後のリクエストが終了したデータ収集を続行する場合は True です。

説明：

この関数は、指定された属性の値を（潜在的に）複数の部分で取得します（大きな属性に適しています）。一般的な大きな属性には、たとえばボーダールータ上の RPL ルート（属性 R_NWK_sourceRoutes）やネイバー キャッシュ エントリ（属性 R_NWK_ndCache）などがあります。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	完全な属性値が正常に取得され、出力バッファに書き込まれた場合。
R_RESULT_SUCCESS_ADDITIONAL_DATA	データの一部が正常に取得されたが、さらにデータが利用可能な場合。
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。

R_RESULT_TIMEOUT	リクエストはタイムアウトで終了しました。2チップのみ適用されます構成。
R_RESULT_ILLEGAL_NULL_POINTER	関数への入力は NULL ポインタでした。
R_RESULT_INSUFFICIENT_BUFFER	使用可能なメッセージ バッファーがありません。
R_RESULT_INSUFFICIENT_OUTPUT_BUFFER	指定された出力バッファは結果を保持するには小さすぎます。
R_RESULT_UNAVAILABLE	要求された属性/値は（まだ）利用できません。これは、いつでも失われる可能性がある値、または最初から不明な値（例: RPL 親情報または RN のブロードキャスト スケジュール）に対する一時的なエラーです。

10.1.7 R_NWK_SetRequest

R_NWK_SetRequest	パーソナルエリアネットワーク情報ベースに値を書き込む

構文：

r_result_t R_NWK_SetRequest(uint8_t attrId,
*const void *p_attrValue、*
uint16_t attrLength)

引数:

(in) *uint8_t attrId*、	書き込む PIB 属性の 8 ビット識別子。 対応 するIDは、「r_nwk_attr_id_t」列挙子タイプ。このドキュメントの第 11 章も参照してください。
(in) const void **p_attrValue*	書き込み値を保持する属性へのポインター。
(in) uint16_t attrLength	書き込む PIB 属性の 16 ビットの長さ。

説明：

この関数は、PIB に値を書き込みます。プリミティブ ID R_NWK_API_MSG_SET_REQ を持つ要求構造をネットワーク層メールボックスにキューに入れます。正常に追加された場合、関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、関数は R_RESULT_INSUFFICIENT_BUFFER の値を返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。 一時停止モード
R_RESULT_INVALID_PARAMETER	無効なパラメータまたはヌルポインタが渡されました
R_RESULT_UNSUPPORTED_FEATURE	このデバイスでは機能がサポートされていません。つまり、デバイスはボーダールータではありません。

R_RESULT_INSUFFICIENT_BUFFER	メッセージ バッファが利用できません

10.1.8 R_ICMP_EchoRequest

R_ICMP_EchoRequest	ICMPエコー要求を送信する

構文：

```
r_result_t R_ICMP_EchoRequest(     const uint8_t* p_dstAddress,
                                                    uint16_t identifier、
                                   uint16_t sequence、
                                   const uint8_t* p_dat
                                   a,
                                   uint16_t dataLength、
                                   uint16_t options、
                                   uint8_t hoplinit)
```

注: この関数は R_NWK_IcmpEchoRequest に置き換えられました。 この関数の詳細についてはセクション 10.1.9 を参照し、この新しい関数を使用するように既存のアプリケーション コードを調整してください。

10.1.9 R_NWK_IcmpEchoRequest

R_NWK_IcmpEchoRequest	ICMPエコー要求を送信する

構文：

r_result_t R_NWK_IcmpEchoRequest(const r_nwk_icmp_echo_req_t request)*

引数（r_nwk_icmp_echo_req_t のメンバー）：

(in) *uint8_t dstAddress[16]*	IPv6 宛先アドレス。
(in) *uint16_t identifier*	ICMPv6 識別子（ICMPv6 によってエコー要求メッセージとエコー応答メッセージを照合するために使用されます）。
(in) uint16_t sequence	ICMPv6 シーケンス番号。
(in) uint16_t options	ネットワーク層への TX オプション。ブロッキング（R_OPTIONS_NON_BLOCKING）を強制的に設定してください。 デフォルトでは、ユニキャスト フレーム交換は DFE モードで実行されます。フレーム交換を EDFE モードで実行する必要がある場合は、次の TX オプションを追加してください。 R_OPTIONS_FRAME_EXCHANGE_EDFE
(in) uint8_t hoplinit	エコー要求に使用される IPv6 ホップ制限値。
(in) r_mac_mdr_params_t mdr	MDRの動作をカスタマイズするためのパラメータ（デフォルトはMAC層による自動選択。詳細はセクション7.7を参照）
(in) uint16_t dataLength	送信する ICMPv6 ペイロードの長さ。
(in) 定数 uint8_t dataPtr*	送信する ICMPv6 ペイロードを含むバッファへのポインタ。

説明：

この関数は、ICMPv6 エコー要求を開始します。この関数は、プリミティブ ID R_NWK_API_MSG_ICMP_ECHO_REQ を持つ要求構造を要求キューに登録します。

制限事項:

関数に提供される構造体ポインタは有効であり、NULL であってはなりません。ペイロードのない ICMPv6 エコー要求を送信する場合、 *dataPtr引数は NULL にすることができます（ただし、 dataLengthが 0 に設定されている*場合のみ）。

戻り値:

R_RESULT_SUCCESS	エコー要求がネットワーク層キューに正常に追加されました。
R_RESULT_FAILED	内部エラーが発生したため、エコー要求を送信できませんでした。
R_RESULT_INVALID_PARAMETER	この関数に無効なパラメータまたは NULL ポインタが渡されました。
R_RESULT_INSUFFICIENT_BUFFER	この要求に必要なバッファを割り当てるにはデバイス上のメモリが不十分です。
R_RESULT_ILLEGAL_STATE	デバイスは参加状態 5 ではないか、デバイスが一時停止されています。
R_RESULT_MAC_NO_ACK	謝辞なし
R_RESULT_MAC_CHANNEL_ACCESS_FAILURE	CSMA/CCA 失敗、チャネルビジー
R_RESULT_RPL_NO_ROUTE	宛先アドレスへのソースルートがありません

10.1.10 R_UDP_DataRequest

R_UDP_DataRequest	UDPデータ要求を送信する

構文：

```
r_result_t R_UDP_DataRequest(      const uint8_t* p_dstAddress,

uint16_t dstPort、
                                   uint16_t srcPort、
                                   const uint8_t* p_dat
a,
                                   uint16_t dataLength、
                                                        uint16_t options)
```

注：このレガシー関数は、下位互換性の理由からのみ残されています。新しい機能はサポートされなくなり、次のリリースのいずれかで削除されます。代わりに、既存のコードを移行して、R_NWK_UdpDataRequest（セクション 10.1.11 を参照）を使用してください。

引数：

(in) *uint8_t* p_dstAddress*	宛先アドレスへのポインタ。
(in) *uint16_t dstPort*	16 ビットの宛先ポート。
(in) uint16_t srcPort	16 ビットの送信元ポート。
(in) uint8_t p_data*	送信するデータへのポインタ。
(in) uint16_t dataLength	送信するデータの16ビットの長さ情報。
(in) uint16_t options	ネットワーク層への 16 ビット オプション。ブロッキングを強制的に設定してください。 注意：デフォルトでは、ユニキャスト フレーム交換は DFE モードで行われます。フレーム交換を EDFE モードで実行する必要がある場合は、次のオプションを追加してください。 R_OPTIONS_FRAME_EXCHANGE_EDFE

説明：

この関数は、UDP データ要求を開始します。プリミティブ ID R_NWK_API_MSG_UDP_DATA_REQ を持つ要求構造を要求キューに追加します。正常に追加された場合、関数はR_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、関数は R_RESULT_FAILED の値を返します。

制限事項:

関数に提供されるすべてのポインタは、「*p_data」を除いて有効であり、NULL ではない必要があります。ペイロードなしで UDP データを送信する場合は、「*p_data」を NULL に設定し、「dataLength」をゼロに設定してください。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。 一時停止モード
R_RESULT_INVALID_PARAMETER	無効なパラメータまたはヌルポインタが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージ バッファが利用できません
R_RESULT_MAC_NO_ACK	謝辞なし
R_RESULT_MAC_CHANNEL_ACCESS_FAILURE	CSMA/CCA 失敗、チャネルビジー
R_RESULT_RPL_NO_ROUTE	宛先アドレスへのソースルートがありません

10.1.11 R_NWK_UdpDataRequest

R_NWK_UdpDataRequest	UDPデータ要求を送信する

構文：

r_result_t R_NWK_UdpDataRequest(const r_nwk_udp_data_req_t request)*

引数（r_nwk_udp_data_reqのメンバー）：

(in) *r_ipv6_addr_t dstAddress*	IPv6 宛先アドレス。
(in) *uint16_t dstPort*	UDP 宛先ポート。
(in) uint16_t srcPort	UDP 送信元ポート。
(in) uint16_t options	ネットワーク層への 16 ビット オプション。ブロッキングを強制的に設定してください。 注意: デフォルトでは、ユニキャスト フレーム交換は DFE モードで行われます。フレーム交換を EDFE モードで実行する必要がある場合は、次のオプションを追加してください。 R_OPTIONS_FRAME_EXCHANGE_EDFE
(in) r_mac_mdr_params_t mdr	MDRの動作をカスタマイズするためのパラメータ（デフォルトはMAC層による自動選択。詳細はセクション7.7を参照）
(in) uint16_t dataLength	送信する UDP ペイロードの長さ。
(in) const uint8_t dataPtr*	送信する UDP ペイロードを含むバッファへのポインタ。

説明：

この関数は、UDP データ要求を開始します。プリミティブ ID R_NWK_API_MSG_UDP_DATA_REQ を持つ要求構造を要求キューに登録します。

制限事項:

関数に提供される構造体ポインタは有効であり、NULL であってはなりません。ペイロードのない UDP データを送信する必要がある場合、 *dataPtr引数は NULL にすることができます（ただし、 dataLengthが 0 に設定されている*場合に限ります）。

戻り値:

R_RESULT_SUCCESS	UDP データ要求がネットワーク層キューに正常に追加されました。

R_RESULT_FAILED	内部エラーが発生したため、UDP データグラムを送信できませんでした。
R_RESULT_INVALID_PARAMETER	この関数に無効なパラメータまたは NULL ポインタが渡されました。
R_RESULT_INSUFFICIENT_BUFFER	この要求に必要なバッファを割り当てるにはデバイス上のメモリが不十分です。
R_RESULT_ILLEGAL_STATE	デバイスは参加状態 5 ではないか、デバイスが一時停止されています。
R_RESULT_MAC_NO_ACK	NO ACK
R_RESULT_MAC_CHANNEL_ACCESS_FAILURE	CSMA/CCA 失敗、チャネルビジー
R_RESULT_RPL_NO_ROUTE	宛先アドレスへのソースルートがありません

10.1.12 R_NWK_IP_DataRequest

R_NWK_IP_DataRequest	IPv6データリクエストを送信する

構文：

r_result_t R_NWK_IP_DataRequest(const uint8_t p_data,*
uint16_t dataLength、
uint16_t options)

引数:

(in) uint8_t p_data*	生の IPv6 パケット データへのポインタ（ヘッダーを含む）
(in) uint16_t dataLength	送信するデータの 16 ビットの長さ情報。サポートされる最大サイズは 1320 バイト（1280 バイトの IPv6 MTU + 40 バイトのトンネリング ヘッダー）です。
(in) uint16_t options	ネットワーク層への 16 ビット オプション。ブロッキングを強制的に設定してください。 注意: デフォルトでは、ユニキャスト フレーム交換は DFE モードで行われます。フレーム交換を EDFE モードで実行する必要がある場合は、次のオプションを追加してください。 R_OPTIONS_FRAME_EXCHANGE_EDFE

説明：

この関数は IPv6 データ要求を開始します。この関数は、APP 層から NWK 層に生の IPv6 パケットを送信するように要求します。このメッセージは、たとえば、FAN ネットワークの外部から受信した IPv6 フレームを FAN ネットワークに転送するために使用されます。この関数は、APP 層から生の IPv6 メッセージの送信を開始するためにも使用できます。生の IPv6 メッセージの生成はユーザーの責任であることに注意してください。

プリミティブ ID R_NWK_API_MSG_IP_DATA_REQ を持つ要求構造を要求キューに追加します。正常に追加された場合、関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、関数は R_RESULT_FAILED の値を返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました

R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。
R_RESULT_INVALID_PARAMETER	NULL ポインタが渡されたか、データ長が無効です
R_RESULT_INSUFFICIENT_BUFFER	メッセージ バッファが利用できません
R_RESULT_MAC_NO_ACK	NO ACK
R_RESULT_MAC_CHANNEL_ACCESS_FAILURE	CSMA/CCA 失敗、チャネルビジー
R_RESULT_RPL_NO_ROUTE	宛先アドレスへのソースルートがありません

10.1.13 R_NWK_MulticastGroupJoinRequest

R_NWK_MulticastGroupJoinRequest	マルチキャストグループに参加する

構文：

r_result_t R_NWK_MulticastGroupJoinRequest(const r_ipv6addr_t groupAddress);*

引数:

(in) *const ipv6addr_t* groupAddress*	このデバイスが参加する必要があるマルチキャストグループを識別する IPv6 アドレス。

説明：

この関数は、指定された IPv6 アドレスによって識別されるマルチキャスト アドレスに参加するように Wi-SUN FAN プロトコル スタックに指示します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCSESS	指定されたマルチキャストグループへの参加要求が正常にキューに入れられました。デバイスがマルチキャストグループに正常に参加すると、 *R_NWK_API_MSG_MULTICAST_GROUP_IND* 指示（セクション0を参照）が送信されます
R_RESULT_ILLEGAL_NULL_POINTER	指定されたポインタがNULLでした

10.1.14 R_NWK_MulticastGroupLeaveRequest

R_NWK_MulticastGroupLeaveRequest	マルチキャストグループを脱退する

構文：

r_result_t R_NWK_MulticastGroupLeaveRequest(const r_ipv6addr_t groupAddress)*
;

引数:

(in) *const ipv6addr_t* groupAddress*	このデバイスが離脱するマルチキャスト グループを識別する IPv6 アドレス。

説明：

この関数は、指定された IPv6 アドレスによって識別されるマルチキャスト アドレスを残すようにWi-SUN FAN プロトコル スタックに指示します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	指定されたマルチキャストグループへの参加要求が正常にキューに入れられました。デバイスがマルチキャストグループから正常に離脱すると、 *R_NWK_API_MSG_MULTICAST_GROUP_IND* 指示（セクション0を参照）が送信されます
R_RESULT_ILLEGAL_NULL_POINTER	指定されたポインタがNULLでした

10.1.15 R_NWK_RplRouteRefreshRequest

R_NWK_RplRouteRefreshRequest	ソースルートを更新する

構文：

r_result_t R_NWK_RplRouteRefreshRequest()

引数:

なし

説明：

この関数は、ボーダールータが使用する宛先広告トリガー シーケンス番号 (DTSN) を増分することにより、PAN 内のすべてのルータ ノードが、その RPL の現在の親を含むボーダールータに DAO メッセージを再送信するようにします。子ノードが DTSN の増分を検出すると（ボーダールータからの次の DIO 受信時)、その子ノードは、その現在の親を含むボーダールータに DAO メッセージを送信し、自身の DTSN を増分する必要があります（詳細については、RFC 6550 を参照してください)。各着信 DAO について、ボーダールータは対応するソース ルートを更新します。この関数は、プリミティブ R_NWK_API_MSG_RPL_REFRESH_ROUTES_REQ を含む要求構造を要求キューにキューイングします。対応する確認プリミティブは、R_NWK_API_MSG_RPL_REFRESH_ROUTES_CFM です。

この機能は、たとえば電源サイクル後にソースルート情報を更新するために使用できます。

制限事項:

この機能は、ボーダールータ構成でのみサポートされます。

戻り値:

R_RESULT_SUCCESS	DTSNが増加しました
R_RESULT_FAILED	RPL モジュールでリクエストが失敗しました（内部エラー）
R_RESULT_ILLEGAL_STATE	デバイスの状態が不正なため、リクエストを実行できませんでした（参加状態 < 5 またはデバイスがボーダールータとして設定されていません）

10.1.16 R_NWK_RplGlobalRepairRequest

R_NWK_RplGlobalRepairRequest	RPLグローバル修復メカニズムを開始する

構文：

r_result_t R_NWK_RplGlobalRepairRequest()

引数:

なし

説明：

この関数により、PAN 内のすべてのルータ ノードは、ボーダールータが使用する DODAG バージョン番号と宛先広告トリガー シーケンス番号（DTSN）を増分して、親の選択を再実行し、新しい RPL 親を含む DAO メッセージをボーダールータに送信します。ボーダールータから次の DIO メッセージを受信すると、現在の DODAG の各ルータ ノードは新しい DODAG バージョン番号を検出し、現在の親を破棄して、収集されたルーティング メトリックに基づいて最適な親を選択します。ルータ ノードの新しいランクは、古い DODAG バージョン内のランクによって制約されません（グローバル修復の詳細については、RFC 6550 を参照してください）。DTSN の増分により、すべてのルータ ノードは、以前の DODAG バージョンと比較して変更されていない場合でも、現在の親をボーダールータに通知する必要があります。この関数は、プリミティブ R_NWK_API_MSG_RPL_GLOBAL_REPAIR_REQ を含む要求構造を要求キューに入れます。対応する確認プリミティブは R_NWK_API_MSG_RPL_GLOBAL_REPAIR_CFM です。

制限事項:

この機能はボーダールータ構成でのみサポートされており、デバイスが起動された後（参加状態 5）にのみ実行できます。

戻り値:

R_RESULT_SUCCESS	地球規模の修復メカニズムは正常に開始されました
R_RESULT_FAILED	RPL モジュールでリクエストが失敗しました（内部エラー）
R_RESULT_ILLEGAL_STATE	デバイスの状態が不正なため、リクエストを実行できませんでした（参加状態 < 5 またはデバイスがボーダールータとして設定されていません）

10.1.17 R_NWK_LeaveNetworkRequest

R_NWK_LeaveNetworkRequest	ネットワークを離れる（ルートノードのみ）

構文：

r_result_t R_NWK_LeaveNetworkRequest();

引数:

なし

説明：

この機能により、ルータ ノードは現在接続されているネットワークから離脱できます。動作は NWK 層のソフトウェア リセット（セクション 10.1.2 を参照）に似ていますが、この場合、デバイスは事前にネットワークを離脱する意図をネイバーに通知します。ネイバーと子ノードに通知するために、無限ランクを含む DIO メッセージによって通知されます。これにより、ネイバーは将来このデバイスを適切な親と見なすことがなくなり、子ノードは別の親に切り替わります。このデバイスのソース ルートを早期に削除できるようにするために、ボーダールータには*No-Path DAO*メッセージによって通知されます。このメッセージの Transit Information オプションには、ライフタイム 0 が含まれています。Renesas ボーダールータでは、離脱するデバイスのソース ルートの削除は、ETX エポックの次の有効期限時に行われます。これには最大 60 秒かかる場合があります。最後に、このデバイスは、ネットワークを離れ、参加状態 0 に移行する前に、ARO ライフタイムが 0 である近隣要請メッセージを送信して、自身の親で登録を解除します。これらのアドバタイズメントはすべてベストエフォート型のアプローチです。つまり、再試行や確認応答 (NS メッセージを除く) はなく、すべてのメッセージが送信されると、関数は直ちにリセット状態に移行します。DIO メッセージはマルチキャスト フレームであるため、フレームが送信されるまでの時間は、構成されたブロードキャスト間隔によって異なります。

この関数は、プリミティブ R_NWK_API_MSG_LEAVE_NETWORK_REQ を含む要求構造を要求キューに登録します。対応する確認プリミティブは R_NWK_API_MSG_LEAVE_NETWORK_CFM です。これはすぐに返されます。

制限事項:

この機能は、ネットワークに完全に参加しているルータ ノード（参加状態 5）でのみ実行できます。

戻り値:

R_RESULT_SUCCESS	デバイスはネットワークから離脱し、登録解除操作はすべて成功しました
R_RESULT_RPL_FAILED	デバイスはネットワークから離脱したが、少なくとも1つの登録解除 操作が失敗しました
R_RESULT_UNSUPPORTED_FEATURE	デバイスはルーターノードではありません
R_RESULT_ILLEGL_STATE	デバイスはまだネットワークに完全に参加していません（join state < 5）

10.1.18 R_NWK_IncreasePanVersionRequest

R_NWK_IncreasePanVersionRequest	PANバージョンを増やす

構文：

r_result_t R_NWK_IncreasePanVersionRequest()

引数:

なし

説明：

この関数は、GTKHASH IE を変更する際に必要な PAN バージョンの増分を要求します。詳細については、[Wi-SUN FAN TPS] を参照してください。この関数は、プリミティブ ID R_NWK_API_MSG_PAN_VER_INC_REQ を持つ要求構造体を要求キューに追加します。正常に追加された場合、この関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、この関数は R_RESULT_FAILED の値を返します。

制限事項:

この機能は、ボーダールータ構成でのみサポートされます。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。
R_RESULT_UNSUPPORTED_FEATURE	このデバイスでは機能がサポートされていません。つまり、デバイスはボーダールータではありません。

10.1.19 R_NWK_ResetPanTimeoutRequest

R_NWK_ResetPanTimeoutRequest	PANタイムアウトをリセット

構文：

r_result_t R_NWK_ResetPanTimeoutRequest()

引数:

なし

説明：

この関数は、PAN タイムアウト間隔のリセットを要求します。この間隔では、FAN ノードは PAN ボーダールータからの活性状態または通信能力の通知を受ける必要があります。そうでない場合は、ボーダールータが故障したとみなされます。この関数は、プリミティブ ID R_NWK_API_MSG_PAN_TIMEOUT_RESET を持つ要求構造を要求キューに追加します。正常に追加された場合、この関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、この関数は R_RESULT_FAILED の値を返します。

制限事項:

この機能は、ボーダールータ構成でのみサポートされます。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。
R_RESULT_UNSUPPORTED_FEATURE	このデバイスでは機能がサポートされていません。つまり、デバイスはボーダールータではありません。

10.1.20 R_NWK_EapolDataRequest

R_NWK_EapolDataRequest	EAPOLメッセージを送信する

構文：

```
r_result_t R_NWK_EapolDataRequest (r_mac_eapol_type_t type、
                                   const uint8_t *dst、
                                   const uint8_t *authenticator,
                                   const uint8_t *data、
                                   uint16_t dataLength);
```

引数:

(in) r_mac_eapol_type_t type	EAPOL タイプ（KMP ID）
(in) uint8_t dst*	EUI-64 宛先アドレス
(in) uint8_t authenticator*	EA-IEの認証子EUI-64（NULLの場合もある）
(in) uint8_t data*	送信する EAPOL メッセージへのポインター。
(in) uint16_t dataLength	データのサイズ（バイト単位）

説明：

この関数は、EAPOL メッセージ送信要求を開始します。プリミティブ ID R_NWK_API_MSG_EAPOL_DATA_REQ を持つ要求構造を要求キューに追加します。正常に追加された場合、関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、関数は R_RESULT_FAILED の値を返します。

制限事項:

関数に提供されるすべてのポインターは、*認証子*引数を除き、有効であり、NULL でない必要があります。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

10.1.21 R_NWK_EapolDataRequestOptimized

R_NWK_EapolDataRequestOptimized	完全に準備されたバッファに埋め込まれたEAPOLメッセージを送信する

構文：

*r_result_t R_NWK_EapolDataRequestOptimized (const uint8_t *dst,*

*uint8_t *data、*

u

int16_t dataLength);

引数:

*(in) const uint8_t *dst*	EUI-64 宛先アドレス
*(in) uint8_t *data*	送信するEAPOLメッセージを含むMPX-IEを含むMCPSデータ要求を表すデータバッファ
(in) uint16_t dataLength	データバッファのサイズ（バイト単位）

説明：

この関数は、メモリの最適化を可能にする NWK タスク用に完全に準備されたバッファに埋め込まれた EAPOL メッセージを送信します。プリミティブ ID R_NWK_API_MSG_EAPOL_DATA_REQ_RAW を持つ要求構造を要求キューに登録します。正常に追加された場合、関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、関数は R_RESULT_FAILED の値を返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

この機能は、R_MODEM_SERVER / R_MODEM_CLIENT 構成ではサポートされていません。

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

10.1.22 R_NWK_EapolDataProcessed

R_NWK_EapolDataProcessed	EAPOLメッセージが処理されました

構文：

r_result_t R_NWK_EapolDataProcessed (r_result_t status);

引数:

(in) r_result_t status	EAPOLメッセージの処理結果

説明：

EAPOL メッセージが処理されたことを NWK タスクに通知します。プリミティブ ID R_NWK_API_MSG_AUTH_MSG_PROCESSED を持つ要求構造を要求キューに追加します。正常に追加された場合、関数は R_RESULT_SUCCESS の値を返します。要求キューに空きがない場合は、関数は R_RESULT_FAILED の値を返します。

制限事項:

なし

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。

10.1.23 R_NWK_EapTlsIsStarting

R_NWK_EapTlsIsStarting	EAP-TLS交換が開始されていることをNWKタスクに通知する

構文：

r_result_t R_NWK_EapTlsIsStarting();

引数:

なし

説明：

EAP-TLS 交換が開始されていることを NWK タスクに通知します。これは、EAP-TLS が開始される前に重要でないメモリを解放するために使用できます。関数は、NWK によって適切に処理された場合は R_RESULT_SUCCESS を返します。それ以外の場合は適切なエラー コードを返します。

制限事項:

なし

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。

10.1.24 R_NWK_EapTlsIsDone

R_NWK_EapTlsIsDone	現在のEAP-TLS交換が完了したことをNWKタスクに通知する

構文：

r_result_t R_NWK_EapTlsIsDone();

引数:

なし

説明：

現在の EAP-TLS 交換が完了したことを NWK タスクに通知します。これは、EAP-TLS の終了後に重要でないメモリを再割り当てするために使用できます。この関数は、NWK によって適切に処理された場合は R_RESULT_SUCCESS を返します。それ以外の場合は適切なエラー コードを返します。

制限事項:

なし

戻り値:

R_RESULT_SUCCESS	リクエストがネットワーク層キューに正常に追加されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。

10.2 NWKの表示とメッセージ

次の表は、ネットワーク層からアプリケーション層に送信される重要なメッセージ、指示、およびデータについて説明しています。詳細については、第 12 章で参照されているソース コードと対応するヘッダー ファイル（例: 「 r_nwk_api.h 」）を参照してください。

表示 / メッセージ	説明

R_NWK_API_MSG_JOIN_STATE_IND	Join State更新表示
	
	この通知は、このデバイスの参加状態が変更されたときに送信されます。対応するデータは、次の通知構造を介して転送されます。
	
	typedef struct
	{
	r_nwk_join_state_t currentJoinState; //!< The new j
	oin state of this device
	} r_nwk_join_state_ind_t;

R_NWK_API_MSG
_DEVICE_STATUS_UPDATE_IND
	デバイスステータス更新表示
	
	この指示は、このボーダールータのネットワーク内の別のデバイスのステータスが変更されたときに送信されます。対応するデータは、次の指示構造と列挙体を介して転送されます。
	
	typedef struct
	{
	r_ipv6addr_t deviceAddress; // <! The IPv6 address to identify the d
	evice whose status has changed
	r_nwk_device_update_code_t updateType; // <! The type of status upda
	te that occurred
	} r_nwk_device_status_ind_t;
	typedef enum R_HEADER_UTILS_ATTR_PACKED
	{
	R_NWK_DEVICE_STATUS_ROUTE_ADDED = 1,
	R_NWK_DEVICE_STATUS_ROUTE_CHANGED = 2,
	R_NWK_DEVICE_STATUS_ROUTE_REMOVED = 3,
	} r_nwk_device_update_code_t;

R_NWK_API_MSG_MULTICAST_GROUP_IND	マルチキャスト グループ更新通知 この表示は、デバイスが特定の IPv6 マルチキャスト グループに正常に参加または離脱したときに送信されます。対応するデータは、次の表示構造と列挙体を介して転送されます。 typedef struct { r_ipv6addr_t groupAddress; r_nwk_multicast_group_event_t event; } r_nwk_multicast_group_ind_t; typedef enum R_HEADER_UTILS_ATTR_PACKED

	{ R_NWK_MULTICAST_GROUP_JOINED = 0, R_NWK_MULTICAST_GROUP_LEFT = 1, } r_nwk_multicast_group_event_t;

R_NWK_API_MSG_IP_DATA_IND	IPデータ表示
	対応するデータはデータ表示構造を介して転送されます。
	
	typedef struct
	{
	uint16_t packetLength; //!< The length of the IP packet
	uint8_t packet[]; //!< The IP packet (raw format)
	} r_ip_data_ind_t;

R_NWK_API_MSG_DHCP_IND	DHCPv6サーバへの受信メッセージ
	
	対応するデータは、DHCP データ表示構造を介して転送されます。
	
	typedef struct
	{
	uint8_t srcAddress[R_IPV6_ADDRESS_LENGTH]; //!< The IPv6 address ..//!< of the sender
	uint16_t srcPort ; // !< The UDP source port
	uint16_t dataLength; //!< The length of the message in bytes
	uint8_t data[]; //!< The DHCP message
	} r_dhcp_data_ind_t;

R_NWK_API_MSG_DHCP_VENDOR_OPT_DATA	DHCPv6から受信したDHCPv6ベンダー固有情報オプションデータ
	
	対応するデータは、DHCP ベンダー固有情報オプション データ表示構造を介して転送されます。
	
	typedef struct
	{
	r_eui64_t src; //!< The EUI-64 of the sender
	uint32_t enterprise_number; //!< The vendor’s registered Enterprise Number //!< as registered with IANA
	uint16_t option_data_len; //!< The length of the following option data in bytes
	uint8_t option_data[]; //!< The DHCPv6 Vendor-specific Information //!< Option data
	} r_nwk_dhcp_vend_opt_data_ind_t;

R_NWK_API_MSG_AUTH_BR_START_IND	ボーダールータ認証プロセスを開始する
R_NWK_API_MSG_AUTH_BR_PER_UPDATE	定期的な更新を実行するためのBR認証の指示

R_NWK_API_MSG_AUTH_RN_START	ルータノード認証プロセスを開始する
	対応するデータは、ルータ ノード開始表示構造を介して転送されます。
	
	typedef struct
	{
	r_eui64_t target; //!< The EUI-64 address
	of the target to authenticate with
	} r_nwk_auth_rn_start_ind_t;

R_NWK_API_MSG_AUTH_RN_RESET	ルートノード認証モジュールをリセットする
R_NWK_API_MSG_AUTH_RN_CLEAR _GTK_VALIDITY	GTK有効フラグをクリアする指示

R_NWK_API_MSG_AUTH_RN_DELETE_GTK	ルータノード認証モジュール内のGTKを削除する
	
	対応するデータは、ルータ ノードの GTK 削除指示構造を介して転送されます。
	
	typedef struct
	{
	r_nwk_gtk_index_t gtk; //!< The GTK th
	at should be deleted from //!< the Rou
	ter Node authentication module
	} r_nwk_auth_rn_gtk_del_ind_t;

R_NWK_API_MSG_EAPOL_DATA_IND	受信EAPOLメッセージ
	対応するデータは、EAPOL データ表示構造を介して転送されます。
	
	typedef struct
	{
	r_eui64_t src; //!< The EUI-64 address of the so
	urce
	r_eui64_t authenticator; //!< The EUI-64 address
	of the authenticator //!< (only if EA-IE was pre
	sent)
	r_mac_eapol_type_t type;
	uint16_t dataLength; //!< The length of the EAPO
	L payload in bytes
	uint8_t data[]; //!< The EAPOL payload
	} r_eapol_data_ind_t;

R_NWK_API_MSG_UDP_DATA_IND	UDP データ表示
	
	この通知は、このデバイスが UDP データグラムを受信したときに送信されます。次の UDP ポートは、Wi-SUN FAN スタックによって内部的に処理されるため、UDP データ通知は発生しません。
	R_DHCPV6_クライアント_ポート 546
	R_DHCPV6_サーバーポート 547
	R_AUTH_EAPOL_RELAY_PORT 10253
	
	対応するデータは、UDP データ表示構造を介して転送されます。
	
	typedef struct
	{
	uint8_t dstAddress[R_IPV6_ADDRESS_LENGTH]; //!< The 128-bit IPv6 address of
	//!< the destination of the packet
	uint16_t dstPort; //!< Has a meaning within the context of a
	//!< particular internet destination address
	uint8_t srcAddress[R_IPV6_ADDRESS_LENGTH]; //!< The 128-bit IPv6 address of
	//!< the source of the packet
	uint16_t srcPort; //!< Port of the sending process
	uint16_t dataLength; //!< The number of bytes contained in the data
	uint16_t status; //!< Rx status (secured/unsecured)
	uint8_t lqi ;
	uint8_t data[] ; // !< The actual UDP data
	} r_udp_data_ind_t;

R_NWK_API_MSG_ICMP_ECHO_IND	ICMPv6 エコー応答表示
	
	この通知は、このデバイスが ICMPv6 エコー応答メッセージを受信したときに送信されます。対応するデータは、ICMP エコー通知構造を介して転送されます。
	
	typedef struct
	{
	uint8_t dstAddress[R_IPV6_ADDRESS_LENGTH]; //!< The 128-bit IPv6 address of
	//!< the destination of the packet
	uint8_t srcAddress[R_IPV6_ADDRESS_LENGTH]; //!< The 128-bit IPv6 address of
	//!< the source of the packet
	uint16_t identifier; //!< An identifier to aid in matching Echo Replies
	uint16_t sequence; //!< A sequence number to aid in matching
	//!< Echo Replies
	uint16_t dataLength; //!< Length of data field
	uint16_t status; //!< Rx status (secured/unsecured)
	uint8_t lqi ;
	uint8_t data[] ; // !< Storage to the actual data
	} r_icmp_echo_reply_ind_t;

R_NWK_API_MSG_ICMP_ERROR_IND	ICMPv6 エラー表示
	
	この通知は、このデバイスが ICMPv6 エラー メッセージを受信したときに送信されます。対応するデータは、ICMP エラー通知構造を介して転送されます。
	
	typedef struct
	{
	uint8_t dstAddress[R_IPV6_ADDRESS_LENGTH];
	//!< The 128-bit IPv6 address of
	//!< the destination of the packet
	uint8_t srcAddress[R_IPV6_ADDRESS_LENGTH]; /
	/!< The 128-bit IPv6 address of
	//!< the source of the packet
	uint8_t type ; // !< ICMP message type
	uint8_t code; //!< Code values of for ICMP e
	rror messages
	uint32_t value; //!< Field of 32 bit, differ
	ent use depending on
	//!< error type
	uint16_t invokingPacketLength; //!< The numb
	er of bytes contained in the
	//!< invoking packet
	uint16_t status; //!< Rx status (secured/uns
	ecured)
	uint8_t lqi;
	uint8_t invokingPacket[]; //!< The actual in
	voking packet
	} r_icmp_error_ind_t;

R_NWK_API_MSG_FLASH_WRITE_IND	フラッシュ書き込み表示 これ 表示 電源投入後にデータを復元できるように、不揮発性メモリにデータを書き込む必要がある場合（つまり、それぞれのデータが変更された場合）に送信されます。対応するデータは、次の表示構造を介して転送されます。 typedef struct { r_flash_attr_id_t attrId; uint16_t dataSize; uint8_t data[]; } r_nwk_flash_write_ind_t;
R_NWK_API_MSG_NWK_WARNING_IND	NWK警告表示 対応するデータは警告表示構造を介して転送されます 。 typedef struct { uint8_t status; } r_nwk_warning_ind_t;
R_NWK_API_MSG_NWK_FATAL_ERROR_IND	NWK 致命的エラー表示 対応するデータは警告表示構造を介して転送されます 。 Typedef struct{ uint8_t status; } r_nwk_fatal_error_ind_t;

表10-1 NWKの表示とメッセージ

第11章 NWK属性

次のネットワーク属性は、Wi-SUN FAN プロトコル スタックによってサポートされており、ユーザーがアクセスできます。

DLMS/COSEM オブジェクト モデルの実装については、「dlms_wisun_mapping」スプレッドシートを参照してください。このスプレッドシートには、ルネサス Wi-SUN FAN スタックから Wi-SUN ネットワーク用の DLMS/COSEM プロファイル（Blue Book Ed. 15 バージョン 1.0 のセクション 4.18 に記載）を作成するために必要な変換が説明されています。

NWK 属性	説明	初期値
R_NWK_phyWiSunPhyConfig	この属性は、Wi-SUN準拠のPHYドライバー PHY 仕様。パラメータとして、このドキュメントの第 8.4 章（表 8.9）に従って、規制ドメイン、動作クラス、および動作モードを指定する必要があります。	{0}
R_NWK_phyFrequency	マクロ名 RP_PHY_FREQUENCY。チャネルの周波数を取得します。 （注1）	（注1）
R_NWK_phyFrequencyOffset	マクロ名 RP_PHY_FREQUENCY_FREQUENCY_OFFSET。設定された値でチャネル周波数を補正します。 （注1）	（注1）
R_NWK_phyRssiOutputOffset	マクロ名 RP_PHY_RSSI_OUTPUT_OFFSET。設定された値でRSSI値とED値を補正します。 （注1）	（注1）
R_NWK_phyCurrentChannel	マクロ名 RP_PHY_CURRENT_CHANNEL。RFチャネルを指定します。 （注1）	（注1）
R_NWK_phyChannelsSupported	マクロ名 RP_PHY_CHANNELS_SUPPORTED。IEEEのPIB phyChannelsSupportedに対応する属性ID 802.15.4g-2012仕様。 （注1）	（注1）
R_NWK_phyTransmitPower	マクロ名 RP_PHY_TRANSMIT_POWER 使用地域ごとに設定されたゲインに応じた送信出力値を設定します。（注1）	（注1）
R_NWK_phyChannelsSupportedPage	マクロ名 RP_PHY_CHANNELS_SUPPORTED_PAGE 「3.1.3 phyChannelsSupported」が有効となるチャネルの範囲を示します。（注1）	（注1）

R_NWK_phyLvlfltrVth	マクロ名 RP_PHY_LVLFLTR_VTH。 このIBを使用することで、受信レベルフィルタの閾値（オプション）を設定することができます。（注1）	（注1）

R_NWK_phyAntennaSwitchEna	マクロ名 RP_PHY_ANTENNA_SWITCH_ENA。 GPIO4ピンからANTSW信号（送信時にハイになる信号）を出力するかどうかを指定します。　（注1）	（注1）
R_NWK_phyAntennaDiversityRxEna	マクロ名 R_NWK_phyAntennaDiversityRxEna 受信アンテナダイバーシティ機能を有効にするか無効にするかを設定します。（注1）	（注1）
R_NWK_phyAntennaSelectTx	マクロ名 RP_PHY_ANTENNA_SELECT_TX このマクロにより、フレームの送受信に使用するアンテナを選択することができます。　（注1）	（注1）
R_NWK_phyRegulatoryMode	マクロ名 RP_PHY_REGULATORY_MODE。 設定送信時間を制限する規制モード。 （注1）	（注1）
R_NWK_phyRmodeTcum	マクロ名 RP_PHY_RMODE_TCUM。 規制モードの観測期間中の合計送信時間。 （注1）	（注1）
R_NWK_phyRmodeThrReleaseTcum	マクロ名 RP_PHY_RMODE_THR_RELEASE_TCUM. 規制モードの総送信時間制限における利用可能な送信時間の閾値。　（注1）	（注1）
R_NWK_phyRmodeTimeReleaseTcum	マクロ名 RP_PHY_RMODE_TIME_RELEASE_TCUM。 合計で利用可能な送信時間にかかる時間規制モードの送信時間制限が閾値以上となること。　（注１）	（注1）
R_NWK_phyRmodeTonMax	マクロ名 RP_PHY_RMODE_TON_MAX。 規制モードの最大送信時間制限 （注1）	（注1）
R_NWK_phyRmodeToffMin	マクロ名 RP_PHY_RMODE_TOFF_MIN 。 規制モードの最小一時停止期間（送信間隔）制限。（注1）	（注1）

R_NWK_phyRmodeTcumSmpPeriod	マクロ名 RP_PHY_RMODE_TCUM_SMP_PERIOD。 規制モードの観測期間あたりの累積伝送時間を計算するためのサンプリング期間。 （注1）	（注1）
R_NWK_phyRmodeTcumLimit	マクロ名 RP_PHY_RMODE_TCUM_LIMIT 。 累積送信時間制限 規制モード（注1）	（注1）

R_NWK_phyFSKOpeMode	マクロ名 RP_PHY_FSK_OPE_MODE MR-FSKモードを指定します。　（注1）	（注1）
R_NWK_phyFreqBandId	マクロ名 RP_PHY_FREQ_BAND_ID 周波数帯域を設定します。　（注1）	（注1）
R_NWK_phyCcaDuration	マクロ名 RP_PHY_CCA_DURATION CCA期間を設定します。　（注1）	（注1）
R_NWK_phyAgcWaitGainOffset	マクロ名 RP_PHY_AGC_WAIT_GAIN_OFFSET 。 これは、RX 待機ゲインと RX レベルしきい値を設定するための IB であり、外部受信 LNA を使用する場合に調整する必要があるパラメータです。 （注1）	（注1）
R_NWK_phyCcaVth	マクロ名RP_PHY_CCA_VTH。これは CCA レベルのしきい値を設定するための IB であり、外部受信 LNA を使用する場合に調整する必要があるパラメータです。 地域（規制ドメイン）とデータ レート（動作モード）に応じて、CCA レベルしきい値の最適化が推奨されます。 （注1）	0x1AA (-86dBm)
R_NWK_phyCcaVthOffset	マクロ名 RP_PHY_CCA_VTH_OFFSET。これは IBはCCAレベルの閾値オフセットを設定するためのもので、外部受信時に調整する必要があるパラメータです。 LNAを使用します。　（注1）	（注1）
R_NWK_phyAntennaSwitchEnaTiming	マクロ名 RP_PHY_アンテナ_スイッチ_ENA_TIMING。これはANTSW機能を有効にするためのIBであり、外部送信アンプを使用する場合に設定する必要があるパラメータです。　（注1）	（注1）

R_NWK_phyGpio0Setting	マクロ名 RP_PHY_GPIO0_SETTING。これは IBはGPIO0出力を設定します。　（注1）	（注1）
R_NWK_phyGpio3Setting	マクロ名 RP_PHY_GPIO3_SETTING。これは IBはGPIO3出力を設定します。　（注1）	（注1）
R_NWK_phyDataRate	マクロ名 RP_PHY_DATA_RATE。 このIBはベースバンドユニットでデータレート[kbps]を読み取るために使用できます。（注1）	（注1）
R_NWK_phyFskTransmitPower	マクロ名 RP_PHY_FSK_TRANSMIT_POWER。 (CWXのみ)	（注1）
R_NWK_phyOfdmTransmitPower	マクロ名 RP_PHY_OFDM_TRANSMIT_POWER。 (CWXのみ)	（注1）
R_NWK_mdrNewModes	基本 PHY 動作モードに加えて、このデバイスによってサポートおよびアドバタイズされる新しいモードを識別する phyModeIds のセット。(MDR が有効な場合のみ使用可能)	該当なし

R_NWK_mdrSizeThreshold	PHY モード スイッチの対象となるフレームの最小サイズ (MP-IE サイズ) (バイト単位)。送信フレームがこのサイズより小さい場合、MAC 層の自動選択では常にベース モードが優先されます。 (MDR が有効な場合のみ使用可能)	40
R_NWK_macMaxCSMABackoffs	マクロ名 RP_MAC_MAXCSMABACKOFF。 IEEEのmacMaxCsmaBackoff MIBに対応する属性ID 802.15.4e-2012仕様。　（注1）	3（注1）
R_NWK_macMinBE	マクロ名 RP_MAC_MINBE。IEEE 802.15.4e-2012 の macMinBE MIB に対応する属性 ID 仕様。　（注1）	3（注1）
R_NWK_macPANID	ボーダールータで設定された PAN の PAN ID。 タイプ: uint16_t; 範囲: 0-65535; (注2)	0xffff
R_NWK_macRxOnWhenIdle	アイドル モードの場合、PHY ドライバーを受信オン モードに自動的に変更します。タイプ: uint8_t、値: 0 = 無効、1 = 有効、	1
R_NWK_macMaxBE	マクロ名 RP_MAC_MAXBE。 IEEE 802.15.4eのmacMaxBE MIBに対応する属性ID 2012年仕様。　（注1）	5　（注1）

R_NWK_macMaxFrameRetries	マクロ名 RP_MAC_MAX_FRAME_RETRIES。 IEEEのmacMaxFrameRetries MIBに対応する属性ID 802.15.4e-2012仕様。　（注1）	5
R_NWK_macFrameCounters	MAC フレーム カウンター。タイプ: uint32_t* frame_counters[R_KEY_NUM]	すべてのカウンターがゼロ {0}
R_NWK_macFrameCounterCheckEnabled	MAC フレーム カウンターのチェックを有効/無効にする属性。 デフォルトではフレーム カウンターのチェックは無効になっています。 タイプ: uint8_t; 値: 0 = 無効; 1 = 有効、	0 = 無効
R_NWK_macFrameCounterOffset	電源投入後に復元できるように、送信フレーム カウンターが不揮発性メモリに書き込まれる間隔（フレーム単位）。(注 3)指定されたオフセットは、各フレーム カウンターの実際の値に追加され、復元されたフレーム カウンターが常に以前に格納されたフレーム カウンターよりも大きくなるようにします。 タイプ: uint16_t	100
R_NWK_macExtendedAddress	MAC 拡張アドレス。タイプ: uint8_t[8];	0x7490500000000001

R_NWK_macWhiteList	MACホワイトリスト。 アプリケーションとMACの間で渡されるMACホワイトリスト用のwhitelist_buffer_t NWK レイヤー。ホワイトリストはテスト目的であることに注意してください。ホワイトリストを利用することで、さまざまなネットワーク トポロジをシミュレートできます。詳細については、14.1.10 章も参照してください。	すべてのエントリがゼロ {0}
R_NWK_macMetricsEnabled	MAC 層のメトリックの収集を有効/無効にします。タイプ: r_boolean_t ; 値: 0 = 無効; 1 = 有効、	0 = 無効
R_NWK_macCounterOctets	MAC メトリック カウンターのサイズ（オクテット単位）。タイプ: uint32_t; (注 4)	4
R_NWK_macRetryCount	確認応答前に 1 回の再試行を必要とした送信フレームの数。タイプ: uint32_t ; (注 4)	0
R_NWK_macMultipleRetryCount	確認応答前に複数回の再試行を必要とした送信フレームの数。タイプ: uint32_t ; (注 4)	0

R_NWK_macTXFailCount	R_NWK_macMaxFrameRetries 後に確認応答が得られなかった送信フレームの数。型: uint32_t; (注4)	0
R_NWK_macTXSuccessCount	正しく確認された送信フレームの数。タイプ: uint32_t; (注4)	0
R_NWK_macFCSErrorCount	不正な FCS のために破棄されたフレームの数。型: uint32_t; (注4)	0
R_NWK_macSecurityFailureCount	MACフレームセキュリティチェックに合格しなかったフレームの数。型: uint32_t; (注4)	0
R_NWK_macDuplicateFrameCount	前のフレームと同じシーケンス番号を含むフレームの数。型: uint32_t; (注4)	0
R_NWK_macRXSuccessCount	正しく受信されたフレームの数。型: uint32_t; (注4)	0
R_NWK_nwkSchedule_Unicast	ユニキャストキャストスケジュール。ポインタ型: r_ie_wp_unicast_schedule_t *; (注 2) (注7)	{0}
R_NWK_nwkSchedule_Broadcast	ブロードキャストスケジュール。ポインターのタイプ: r_ie_wp_broadcast_schedule_t *; (注２) (注７)	{0}
R_NWK_rplDodagVersion	DODAG バージョンは、指定された DODAG ID を持つ DODAG の特定の反復 (「バージョン」) です。タイプ: uint8_t。範囲: 0-255. (注5);(注6); (BR でのみ使用可能)	240
R_NWK_rplDtsn	宛先広告トリガー シーケンス番号。タイプ: uint8_t。範囲: 0～255。 (注5)。(注6)。(BR でのみ使用可能)	240

R_NWK_rplLifetimeUnit	RPL でルートの有効期間を表すために使用される秒単位。範囲: 60-3600。タイプ: uint16_t(BR でのみ使用可能)	60
R_NWK_rplLifetimeDefault	RPL ライフタイム ユニット単位で表された RPL ルートのデフォルトのライフタイム。範囲: 1-255。タイプ: uint8_t(BR でのみ使用可能)	6
R_NWK_rplMinHopRankInc	ノードとその DODAG 親間のランクの最小増加。範囲: 32-256。タイプ: uint16_t(BR でのみ使用可能)	128

R_NWK_rplDisMaxInitialInterval	最大初期 DIS 間隔（秒単位）。実際の間隔は、I/2 から I の間のランダムな値になります。 後続の DIS 送信は、30 ～ 60 秒のランダム間隔でスケジュールされますが、これは設定できません。範囲: 0 ～ 255(LFNでは利用できません)	60
R_NWK_rplDioIntervalMin	名前はDEFAULT_DIO_INTERVAL_MINです。DIOトリクルタイマーImin値は、この値の2乗（ミリ秒単位）として定義されます。範囲：1～255。初期値は区間Iは2^(に設定される R_NWK_rplDioIntervalMin + R_NWK_rplDioIntervalDoublings) ミリ秒。 （注2）(LFNでは利用できません)	15
R_NWK_rplDioIntervalDoublings	名前 デフォルト_DIO_INTERVAL_DOUBLINGS. DIO トリクル タイマーの Imax 値。範囲: 1 ～ 8 倍。（注 2）(LFNでは利用できません)	2
R_NWK_rplDioRedundancyConstant	名前 デフォルト_DIO_REDUNDANCY_CONSTANT。DIO冗長定数のデフォルト値。 範囲: 0～1。（注2）(LFNでは利用できません)	0
R_NWK_deviceMinSens	名前はDEVICE_MIN_SENS。ベンダー特定の最小受信機感度レベル。タイプ: uint8_t; 範囲: 0 (-174 dBm) ～ 254 (+80 dBm) (TRG のみ、 CWX の場合は PHY ドライバーから値が継承されます) (注 2)	69 = (-105 dBm)
R_NWK_rplFirstParentWaitTime	最初の親を選択する前の、結合状態 4 での最小待機時間（秒単位）。時間が長いほど、より多くの DIO が受信されるため、最初の親の選択が改善される可能性があります。一方、時間が短いほど、結合プロセスが高速化されます。範囲: 0 ～ 65535(LFN では利用できません)	60
R_NWK_rplNbrMinLifetimeSecs	RPL ネイバー キャッシュ エントリの最小存続期間（秒単位）	600
R_NWK_rplNudTimeout	近隣到達不能検出を実行するまでの最大期間（秒単位）。タイプ: uint16_t、範囲: 60-65535;	1800
R_NWK_panVersion	PAN バージョンは、ボーダールータがノードに PAN 構成の変更を通知するために使用されます。タイプ: uint16_t、範囲: 0-65535; (注 5)	0

R_NWK_lfnVersion	PAN ボーダールータによって配布される現在の LFN バージョン。 タイプ: uint16_	0
R_NWK_panTimeout	名前 PAN_TIMEOUT。FAN ノードが PAN ボーダールータから活性状態または通信能力の通知を受ける必要がある間隔。そうでない場合は、ボーダールータに障害が発生したとみなされます。 タイプ: uint8_t; 範囲: 1～255 分; （注2）	90
R_NWK_targetPanId	この属性は、追加の PAN ネットワーク参加基準として使用できます。特定のターゲット PAN ID が定義されている場合、ルータ ノードは、ネットワーク名と PAN ID が一致する場合にのみネットワークに参加します。デフォルト値 0xffff は、PAN ID フィルタリングを無効にします。 タイプ: uint16_t; 範囲: 0-65535;	0xffff
R_NWK_panIdBlacklist	PAN 検出時に使用されない PAN ID のブラックリスト（uint16_t 型の 8 つの要素を持つ配列）。すべてのエントリを 0xffff に設定すると、ブラックリストが無効になります。注: GET/SET コマンドは常に完全な配列を読み取り/書き込みます。	すべてのエントリ 0xffff
R_NWK_authJoinTimerMax	認証に失敗した場合に参加状態 2 のノードが参加状態 1 に戻るまでの最大時間（秒）。タイプ: uint16_t、範囲: 60-65535。	3600
R_NWK_authJoinTimerBase	認証が失敗した場合の初期再送信時間（秒単位）。 タイプ: uint16_t	300
R_NWK_authJoinTimerBackoffFactor	認証再送信のバックオフ係数。 タイプ: uint8_t	2
R_NWK_dhcpSolicitMaxInitialDelay	このデバイスからの最初の DHCPv6 要請メッセージの最大遅延（秒単位）。この値を減らすと、このデバイスの参加時間が短縮される可能性がありますが、多くのノードが同時にネットワークに参加しようとすると衝突が発生する可能性もあります。 範囲: 0～65535秒 （注2） ; （注5）	FFNの場合は60秒 LFN の場合は 600 秒

R_NWK_dhcpSolicitInitialTimeout	このデバイスからの各初期 DHCPv6 要請メッセージの初期タイムアウト（秒単位）。このタイムアウト内に DHCPv6 応答メッセージが受信されない場合、デバイスは DHCPv6 要請メッセージを再送信します。再送信バックオフは指数関数的であり、つまり、後続の送信ではタイムアウトが 2 倍になります。 範囲: 10 ～ 65535 秒 （注2） ； （注5）	FFNの場合は60秒 LFN の場合 1800 秒
R_NWK_discIntervalImin	名前DISC_IMIN。検出トリクルタイマーImin。範囲：1～255秒。初期間隔Iは次のように設定されます。 R_NWK_discIntervalImin。（注2）	15

R_NWK_discIntervalImax	名前 DISC_IMAX。ディスカバリートリクル タイマーImax。範囲：1～8倍増；（注2）	2
R_NWK_gtkRequestIminMinutes	名前をGTK_REQUEST_IMINにします。 グループ キーハンドシェイクトリクルタイマーImin値。範囲: 1～255分。初期 送信間隔t（秒）はt=R_NWK_gtkRequestIminMinutes *に設定されます。 60 + rand([0;16]). （注2）	4
R_NWK_gtkRequestImax	名前 GTK_REQUEST_IMAX。グループ キーハンドシェイクトリクルタイマーImax値。範囲: 1～8倍増；（注2）	4
R_NWK_gtkMaxMismatchMinutes	名前 GTK_MAX_MISMATCH。SUP が GTKHASH の不一致を検出してから、SUP が認証フローの Msg1 を開始するまでの最大時間。タイプ: uint8_t、 範囲: 1～255；（注2）	64
R_NWK_lgtkMaxMismatchMinutes	名前 LGTK_MAX_MISMATCH。SUPがLGTKHASHの不一致を検出してから、 認証フローのSUP開始Msg1。 タイプ: uint8_t、 範囲: 1～120；（注2）	60
R_NWK_mplAddSeedIdToMplOption	有効にすると、「S」フィールドは 3 に設定され、FAN 1.0 の要件に従って、各送信 MPL メッセージの MPL オプションに MPL シード識別子が追加されます。デフォルトでは、「S」フィールドは 0 に設定され、FAN 1.1 の推奨に従ってシード ID は省略されます。ネットワークに FAN 1.0 ノードが含まれている場合は、このオプションを有効にする必要があります。	0

R_NWK_mplSeqNumStart	このデバイスからの MPL データ メッセージ送信のシーケンス番号の初期値。 タイプ: uint8_t、範囲: 0-255;	0
R_NWK_mplTrickleImin	名前 DATA_MESSAGE_IMIN。[MPL] トリクルタイマーImin値。タイプ: uint8_t、範囲: 1-255秒。初期間隔Iはランダムに選択されます。[Imin, 2*Imin]。 （注2）	10
R_NWK_mplTrickleImax	名前 DATA_MESSAGE_IMAX。[MPL] トリクル Imax 値。タイプ: uint8_t、範囲: 1-8 倍増; （注 2）	3
R_NWK_mplTrickleK	MPL データ メッセージ送信の冗長定数。タイプ: uint8_t	1
R_NWK_mplTrickleTimerExpirations	MPL トリクル タイマーの有効期限、このメッセージが削除されるまでのタイマーの有効期限の数。タイプ: uint8_t	3
R_NWK_mplSeedSetEntryLifetime	シード セット内の MPL エントリの最小有効期間（秒単位）。タイプ: uint16_t 範囲: 1～65535秒	1800

R_NWK_fanStackVersion	読み取り専用: この Wi-SUN FAN スタックのバージョンとコンパイル時の構成。タイプ: r_app_nwk_version_buffer_t （注 7）	{
		0xE1、0x0
		1、
		「V1.6.0」
		}

R_NWK_fanInterfaceMgmtInfo	読み取り専用: Wi-SUN FAN ネットワーク インターフェイスに関する管理情報（RFC 1213 に基づく）。タイプ: r_nwk_if_mib_t （注7）	{
		“Renesas Wi-SUN FAN Network Interface”,
		0x7490500000000001,
		{0}
		}

R_NWK_nwkSecurityEnabled	予約済み。	該当なし
R_NWK_nwkIpv6Address	デバイスのIPv6アドレス。タイプ: uint8_t[16];	0xFE8000000000000007690500000000001
R_NWK_ndCache	キャッシュエントリ。アプリケーションとNWK層の間で渡される近隣探索キャッシュのnd_cache_buffer_tへのポインタ	{0}
R_NWK_borderRouterAddress	ボーダールータ IPv6 アドレス。タイプ:	{0}

	uint8_t[16];	
R_NWK_preferredParentAddress	優先親 IPv6 アドレス。タイプ: uint8_t[16];	{0}
R_NWK_alternateParentAddress	代替親 IPv6 アドレス。タイプ: uint8_t[16];	{0}
R_NWK_sixLowpanMtu	6LoWPAN MTU サイズ。タイプ: uint16_t ; 範囲: 400-1280;	1280
R_NWK_rplDemoMode	予約済み。	該当なし
R_NWK_logVerbosity	詳細レベル（モデム サーバーのみ）。タイプ: uint8_t; 値: 0 = DISABLED、1 = ERROR、2 =WARNING、3 = DEBUG	0 = disabled
R_NWK_deviceType	デバイス タイプの構成。タイプ: uint8_t ; 値: 0 = R_BORDERROUTER; 1 = R_ROUTERNODE、	0 =R_BORDERROUTER
R_NWK_joinState	デバイスの参加状態。タイプ: r_nwk_join_state_t; 詳細については、このドキュメントの 8.17 も参照してください。（注 7）	0 = R_NWK_ JoinState0_Reset
R_NWK_panSize	PAN 広告を通じて当社の優先親会社 / BR によって配布される PAN サイズの値。	0
R_NWK_lastPanConfigRecvTime	このデバイスでの最新の [LFN] PAN 構成受信のタイムスタンプ	0
R_NWK_framesecGtk	GTKを設定または削除します。ポインターのタイプ: r_nwk_framesec_gtk_t*（注 7）	該当なし
R_NWK_framesecActiveGtk	使用する新しい GTK をアクティブにします。 タイプ: r_nwk_gtk_index_t（注7）	該当なし

R_NWK_macFrameSubscriptionEnabled	TBU用に予約済み	該当なし
R_NWK_authStateCache	FAN スタックによってキャッシュされた認証状態を示します。	該当なし
R_NWK_rplDaoIrt	各 DAO 送信の初期再送信時間（秒単位）。この属性は FANTPS の "DAO_IRT" パラメータに対応し、基礎となるアルゴリズムは [RFC3315] セクション 14 で説明されています。（注 2）（注 5） 範囲: 1～60秒; タイプ: uint8_t	3

R_NWK_rplDaoMrc	諦めるまでの DAO 再送信の最大回数、または制限がない場合は 0。この属性は FANTPS の "DAO_MRC" パラメータに対応し、基礎となるアルゴリズムは [RFC3315] セクション 14 で説明されています。(注 2) (注 5) 範囲: 0-15; タイプ: uint8_t	10
R_NWK_rplDaoMrt	DAO 再送信間の最大時間 (秒単位) (ランダム化は無視)、または制限がない場合は 0。この属性は FANTPS の "DAO_MRT" パラメータに対応し、基礎となるアルゴリズムは [RFC3315] セクション 14 で説明されています。(注 2) (注 5) 範囲: 0～600秒; タイプ: uint16_t	0
R_NWK_rplDaoMrd	DAO 交換 (すべての再送信を含む) の最大合計継続時間 (秒単位)。この時間が経過すると、諦めるか、制限がない場合は 0 になります。この属性は FANTPS の "DAO_MRD" パラメータに対応し、基礎となるアルゴリズムは [RFC3315] セクション 14 で説明されています。(注 2) (注 5) 範囲: 0～3600秒; タイプ: uint16_t	0
R_NWK_rplDaoRand	前回の再送時間に基づく DAO 再送のランダム化率。たとえば、値が 25 の場合、各再送で前回の再送タイムアウトの +/- 0.25 のランダム化係数が使用されることを意味します。この属性は FANTPS の「DAO_RAND」パラメータに対応し、基礎となるアルゴリズムは [RFC3315] セクション 14 で説明されています。 (注 2) (注 5) 範囲: 0-50%; タイプ: uint8_t	10%
R_NWK_rplInfo	WiSUN FAN スタックからさまざまな RPL 情報を取得します (読み取り専用)。タイプ: r_nwk_rpl_info_t (注 7) (LFN では使用できません)	該当なし
R_NWK_sourceRoutes	Wi-SUN FAN ボーダールータからすべての RPL ソース ルートを取得します (読み取り専用)。 タイプ: r_app_routing_table_buffer_t (注 7) (BR のみ使用可能)	{0}

R_NWK_dhcpEnableVendorOption	有効にすると、DHCPv6クライアントはSolicitメッセージにベンダー固有の情報オプションを含め、 サーバーにこのオプションを要求する（詳細は8.13を参照） タイプ: r_boolean_t 許可される値: 0=無効 1=有効	0（無効）
R_NWK_udpIpIndication	着信 UDP メッセージに対して、UDP データ表示の代わりに IP データ表示を生成します。タイプ: r_boolean_t ; 値: 0 = 無効; 1 = 有効、	0
R_NWK_lfnSupportGlobal	Trueの場合、このボーダールータのPANでLFNがサポートされます。Falseの場合、 BR は LFN 関連の IE を送信しないでください。	1（正しい）
R_NWK_lfnSupportLocal	このデバイスが LFN の FFN 親になれる場合は True。LFN からの参加要求（LPAS）を無視する場合は False。（ペアレンティングが有効になっている場合にのみ使用可能）	1（正しい）
R_NWK_fanTPSVersion	この属性は、ノードがどの FAN TPS バージョンに認定されているかを示します。 1 = FAN 1.0 認定 2 = FAN 1.1 認定	2
R_NWK_panCapacity	このBRのPANに参加できるノードの最大数。この値はPAN負荷を計算するために使用されます。 PAN 広告の JM-IE を介して配布される係数（PLF）結合メトリック。タイプ: uint16_t	64
R_NWK_jmPlfEnabled	PAN 負荷係数（JM-PLF）を計算して JMIE に含める場合は True。それ以外の場合は False。タイプ: uint8_t	1（正しい）
R_NWK_customJoinMetrics	TBU用に予約済み	該当なし
R_NWK_customHeaderIE	デモまたはテストの目的でこのデバイスによって [L]PC フレームに配布されるカスタム ヘッダー IE。タイプ: r_mac_custom_ie_t(BR でのみ使用可能)	該当なし

R_NWK_customPayloadIE	デモまたはテストの目的でこのデバイスによって [L]PC フレームで配布されるカスタム ペイロード IE。タイプ: r_mac_custom_ie_t(BR でのみ使用可能)	該当なし
R_NWK_ipAddrRegLifetime	[E]ARO 経由で親に登録する場合の GUA/ULA の有効期間（分単位）。タイプ: uint16_t	0xFFFF
R_NWK_ipAddrRegRefreshThreshold	[E]ARO を介して GUA/ULA が親に再登録される間隔（アドレスの有効期間全体に対するパーセンテージ）。範囲: 0-100。タイプ: uint8_t	80
R_NWK_ncrRetries	「登録更新要求」メッセージ [MCASTREG] の再試行回数。初期 TID は 0xFF からこの値を引いた値に設定されます。範囲: 0 ～ 127。タイプ: uint8_t (LFN では利用できません)	3
R_NWK_ncrSeriesDuration	登録更新メカニズム [MCASTREG] の初期シリーズの合計期間（つまり、すべての「登録更新要求」メッセージを送信する時間）（秒単位）。範囲: 0 ～ 255。タイプ: uint8_t (LFN では利用できません)	10
R_NWK_ncrEnabled	このデバイスが次回 PAN に完全に参加する（参加状態 5 に到達）ときに、「ネイバー キャッシュ更新」機能を有効/無効にします。有効にすると、このデバイスは「登録更新要求」メッセージ [MCASTREG] を送信して、FFN ネイバーにアドレス登録の更新を強制します。この機能は、電源サイクル後にこのデバイスの ARO ネイバー キャッシュを復元することを目的としています。値: 0 = 無効、1 = 有効。タイプ: uint8_t (LFN では利用できません)	0
R_NWK_lfnNaWaitDuration	LFN が FFN NA(EARO) 応答を待機する間隔（分）。範囲: 30-120。タイプ: uint8_t	60
R_NWK_lfnMaintainParentTime	LFN が親が失われたと想定するまでの LFN_BROADCAST_SYNC_PERIOD の数（何も受信されなかった場合）。範囲:1-60. 型: uint8_t	5
R_NWK_multicastRegLifetime	FFN 親における LFN マルチキャスト グループ登録の有効期間（分単位）。タイプ: uint16_t	10080

R_NWK_IfnBroadcastScheduleConf	ボーダールータによって配布される LFN ブロードキャスト スケジュール/構成。タイプ: r_ie_wh_lfn_broadcast_config_t (BR でのみ使用可能)	該当なし
R_NWK_IfnUnicastScheduleEapol	EAPOL 中の LFN ユニキャスト スケジュール。タイプ: r_ie_wh_lfn_unicast_schedule_t(LFN でのみ使用可能)	該当なし
R_NWK_IfnUnicastScheduleOpnl	動作状態中の LFN ユニキャスト スケジュール。タイプ: r_ie_wh_lfn_unicast_schedule_t(LFN でのみ使用可能)	該当なし

（注1）詳細については、Sub-GHz RFドライバAPI仕様を参照してください。
（注2）詳細については、Wi-SUN FAN技術プロファイル仕様書を参照してください。
（注3）データフラッシュを含むデバイスのみサポートされます。
（注4） IEEE Std 801.15.4-2015の8.4.2.6章に準拠したMACパフォーマンスメトリック。
（注5）リセット状態でのみ書き込み可能です。
（注6）動作状態でのみ読み取り可能です。
（注7）詳細については本書の第12章を参照してください。

表11-1 NWK属性

第12章 定義、列挙型、構造体

Wi-SUN FANプロトコルでサポートされている定義、列挙型、構造の詳細については、
、ライブラリ パッケージのディレクトリ「 Wi_SUN_FAN_RX_FW_LIB¥stack¥inc 」にあります。

- r_nwk_api.h
- r_nwk_api_base.h
- r_nwk_global.h
- r_nwk_inter_config.h
- r_mac_ie.h
- r_dhcpv6.h

第13章 Wi-SUN FANスタックインタラクションのためのAPP API

このセクションでは、Wi-SUN FAN プロトコル スタックの動作に関連するアプリケーション レイヤーのさまざまなモジュールの API について説明します。たとえば、Wi-SUN FAN の認証モジュールは、柔軟性と制御性を高めるために、完全にアプリケーション レイヤーに常駐します（つまり、FAN スタック ライブラリの一部ではありません）。提供されているサンプル アプリケーションは、これらの API を使用して、Wi-SUN FAN に必要な動作を実現します。この章では、これらの API の最も重要な機能について説明します。これらのモジュールを変更することは可能ですが、WiSUN FAN プロトコル スタックとのやり取りを中断したり、FANTPS で定義されている動作に違反したりしないように注意する必要があります。

13.1.1 R_DHCPV6_ServerInit

R_DHCPV6_ServerInit	DHCPv6サーバーを初期化する

構文：

*r_result_t R_DHCPV6_ServerInit (const uint8_t *eui64,*
*r_dhcpv6_server_lookup_cb lookup、 void *lookupUser);*

引数:

(in) *const uint8_t *eui64*	ノードの EUI-64。
(in) *r_dhcpv6_server_lookup_cb lookup*	ルックアップ関数。
*(in) void *lookupUser*	ルックアップ関数に渡される不透明なユーザーポインター。

説明：

この関数は、DHCPv6 サーバーを初期化して起動します。DHCP サーバーが正常に起動された場合、この関数は R_RESULT_SUCCESS を返し、それ以外の場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	DHCP サーバーが正常に起動しました
R_RESULT_FAILED	DHCPサーバーを起動できませんでした。 DHCPサーバポートを登録できません
R_RESULT_ILLEGAL_NULL_POINTER	無効なパラメータまたはヌルポインタが渡されました

13.1.2 R_DHCPV6_ServerProcessMsg

R_DHCPV6_ServerProcessMsg	DHCPv6サーバのプロセスメッセージ

構文：

*r_result_t R_DHCPV6_ServerProcessMsg (const uint8_t *src,*
*const uint8_t *in、uint16_t in_size、uint8_t *out、ui*
nt16_t outSize);

引数:

(in) const *uint8_t *src*	受信したメッセージの IPv6 送信元アドレス。
(in) const *uint8_t *in*	受信したメッセージ バッファ。
(in) uint16_t in_size	受信したメッセージのサイズ。
*(出力) uint8_t *out*	応答メッセージを保持する出力バッファ。
(入力) uint16_t outSize	出力バッファのサイズ（バイト単位）。

説明：

この関数は、DHCPv6 サーバーの受信 DHCP クライアント メッセージを処理し、対応する DHCP サーバー メッセージを生成します。対応するプリミティブ ID は*R_NWK_API_MSG_DHCP_IND*です。サーバー メッセージが正常に処理された場合、この関数は R_RESULT_SUCCESS を返し、それ以外の場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	サーバーメッセージは正常に処理されました
R_RESULT_FAILED	パケットはキューに追加されませんでした。つまり、サスペンドモードが原因です。
R_RESULT_ILLEGAL_NULL_POINTER	無効なパラメータまたはヌルポインタが渡されました
R_RESULT_ILLEGAL_STATE	DHCPサーバーが初期化されていません

R_RESULT_INVALID_PARAMETER	DHCP クライアント メッセージに無効なパラメータが見つかりました
R_RESULT_INSUFFICIENT_BUFFER	生成されたDHCPサーバーメッセージに十分なバッファがありません

13.1.3 R_DHCPV6_ServerStop

R_DHCPV6_ServerStop	DHCPv6サーバーを停止する

構文：

void R_DHCPV6_ServerStop (void)

引数:

なし

説明：

この関数は、DHCPv6 サーバーを停止します。

13.1.4 R_AUTH_BR_Reset

R_AUTH_BR_Reset	BR認証子の管理構造をリセットする

構文：

void R_AUTH_BR_Reset (r_apl_global_t nwk)*

引数:

(in) r_apl_global_t* nwk	グローバルネットワーク情報構造

説明：

この関数は、すべてのメモリを解放し、すべての内部状態を削除することで、ボーダールータ/認証子の管理構造をリセットまたは初期化します。この関数は、R_AUTH_BR_Start の前に呼び出す必要があります。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

13.1.5　R_AUTH_BR_Start

R_AUTH_BR_Start	BR認証システムを起動します

構文：

void R_AUTH_BR_Start (　　r_apl_global_t nwk,*
*　　r_boolean_t restoreFromNvm);*

引数:

(in) r_apl_global_t* nwk	グローバルネットワーク情報構造
(in) r_boolean_t restoreFromNvm	認証状態を不揮発性メモリから復元する場合は true に設定します。それ以外の場合は false に設定します。

説明：

この関数は、認証モジュールを起動します。R_AUTH_BR_Reset を除く他のすべての R_AUTH_BR_* 関数呼び出しの前に呼び出す必要があります。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

13.1.6 R_AUTH_BR_ProcessDirectMessage

R_AUTH_BR_ProcessDirectMessage	受信したEAPOLダイレクトメッセージを処理する

構文：

r_result_t R_AUTH_BR_ProcessDirectMessage (r_apl_global_t nwk,*
r_mac_eapol_type_t type、const uint8_t source、const void* data*
、uint16_t size)

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造
(in) *r_mac_eapol_type_t type*	KMP ID
(in) *const uint8_t* source*	受信メッセージのEUI-64送信元アドレス
(in) *const void* data*	メッセージ
(in) *uint16_t size*	メッセージのサイズ

説明：

この関数は、着信 EAPOL ダイレクト メッセージを処理します。対応するプリミティブ ID は R_NWK_API_MSG_EAPOL_DATA_IND です。関数は、成功した場合は R_RESULT_SUCCESS を返し、それ以外の場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_ILLEGAL_STATE	認証モジュールの状態が異なり、リクエストを許可しません
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.7 R_AUTH_BR_ProcessRelayMessage

R_AUTH_BR_ProcessRelayMessage	受信EAPOLリレーメッセージを処理する

構文：

r_result_t R_AUTH_BR_ProcessRelayMessage (r_apl_global_t nwk,*
const uint8_t source、const uint8_t* dest、const uint8_t* data、uint16_t size)*

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造
(in) *const uint8_t* source*	受信メッセージの IPv6 送信元アドレス
(in) *const uint8_t* dest*	受信メッセージの IPv6 宛先アドレス
(in) *const void* data*	メッセージ
(in) *uint16_t size*	メッセージのサイズ

説明：

この関数は、着信 EAPOL リレー メッセージを処理します。対応するプリミティブ ID は R_NWK_API_MSG_UDP_DATA_IND です。関数は、成功した場合は R_RESULT_SUCCESS を返し、それ以外の場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_ILLEGAL_STATE	認証モジュールの状態が異なり、リクエストを許可しません
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.8　R_AUTH_BR_PeriodicUpdate

R_AUTH_BR_PeriodicUpdate	GTKの定期的な更新を実行する

構文：

r_boolean_t R_AUTH_BR_PeriodicUpdate　(r_apl_global_t nwk)*

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造

説明：

この関数はGTKの定期的な更新を実行します。少なくとも1分ごとに呼び出される必要があります。PANバージョンを増やす必要がある場合、この関数はTRUEを返します。対応するプリミティブIDはR_NWK_API_MSG_AUTH_BR_PER_UPDATE です。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

TRUE	PAN バージョンを増やす必要があります。
FALSE	PAN バージョンの増分は必要ありません。

13.1.9　R_AUTH_BR_RevokeGTK

R_AUTH_BR_RevokeGTK	既存のGTKを取り消す

構文：

r_result_t R_AUTH_BR_RevokeGTKs（　　r_apl_global_t nwk,*
const uint8_t new_gtk）*

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造
(in) *const uint8_t* new_gtk*	インストールすべき新しいGTK

説明：

この関数は、既存の GTK を取り消して、提供された新しい GTK をインストールします。[FANWG-6.5.2.5 ノード アクセスの取り消し] で説明されているように、GTK に関する手順を実行し、呼び出し元に PAN バージョンの更新を要求します。関数は、成功した場合は R_RESULT_SUCCESS を返し、そうでない場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.10　R_AUTH_BR_RevokeSupplicant

R_AUTH_BR_RevokeSupplicant	既存のサプリカントを取り消す

構文：

r_result_t R_AUTH_BR_RevokeSupplicant (r_apl_global_t nwk,*
const uint8_t supplicant)*

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造
(in) *const uint8_t* supplicant*	取り消されるサプリカントのMACアドレス

説明：

この関数は、既存のサプリカントを取り消します。この関数は、PMK、PTK、および GTK のタイムアウトを 0 に設定して EAP-TLS 交換を強制し、呼び出し元に PAN バージョンの更新を要求します。この関数は、成功した場合は R_RESULT_SUCCESS を返し、そうでない場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.11 R_AUTH_SUP_Reset

R_AUTH_SUP_Reset	サプリカント管理構造をリセットする

構文：

r_result_t R_AUTH_SUP_Reset (r_apl_global_t nwk)*

引数:

(in) r_apl_global_t* nwk	グローバルネットワーク情報構造

説明：

この関数はサプリカント管理構造をリセットします。R_AUTH_SUP_Init は保持されます。対応するプリミティブ ID は R_NWK_API_MSG_AUTH_RN_RESET です。関数は成功した場合は R_RESULT_SUCCESS を返し、そうでない場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.12 R_AUTH_SUP_Init

R_AUTH_SUP_Init	サプリカント管理構造を初期化する

構文：

r_result_t R_AUTH_SUP_Init (r_apl_global_t nwk,*
const uint8_t encoded_cets、uint8_t cert_index、size_*
t mac_offset)

引数:

(in) r_apl_global_t* nwk	グローバルネットワーク情報構造
(in) *const uint8_t* encoded_certs*	encode-certsツールによってエンコードされた証明書
(入力) *uint8_t cert_index*	使用する秘密鍵と証明書チェーンのインデックス
(in) *size_t mac_offset*	MAC層に渡されるバッファの先頭に追加する必要があるバイト数

説明：

この関数は、サプリカント管理構造を初期化します。他の R_AUTH_SUP_* 関数の前に呼び出す必要があります。関数は、成功した場合は R_RESULT_SUCCESS を返し、そうでない場合は適切なエラーコードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.13 R_AUTH_SUP_StartJoin

R_AUTH_SUP_StartJoin	認証プロセスを開始/続行する

構文：

r_result_t R_AUTH_SUP_StartJoin (r_apl_global_t nwk、const uin t8_t* target)*

引数:

(in) r_apl_global_t* nwk	グローバルネットワーク情報構造
(in) *const uint8_t* target*	認証のために選択されたターゲットの EUI-64 アドレス、または以前のターゲットで認証を再開する場合は NULL

説明：

この関数は、Wi-SUN 初期 EAPOL-Key パケットを送信して、サプリカントの認証プロセスを開始/継続します。対応するプリミティブ ID は R_NWK_API_MSG_AUTH_RN_START です。関数は、成功した場合は R_RESULT_SUCCESS を返し、それ以外の場合は適切なエラー コードを返します。

制限事項:

関数に提供される nwk ポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.14 R_AUTH_SUP_ProcessEapolMessage

R_AUTH_SUP_ProcessEapolMessage	受信EAPOLメッセージを処理する

構文：

r_result_t R_AUTH_SUP_ProcessEapolMessage (r_apl_global_t nwk,*
r_mac_eapol_type_t type、const uint8_t source、*
const uint8_t authenticator、*
uint8_t data、uint16_t size)*

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造
(in) *r_mac_eapol_type_t type*	KMP ID
(in) *const uint8_t* source*	受信メッセージのEUI-64送信元アドレス
(in) *const uint8_t* authenticator*	EAPOLAuthenticator-EUI-64 IE が存在する場合の認証子の EUI-64 (NULL の場合もあります)
(in) *const void* data*	メッセージ。これは、AUTH ヒープ上の動的に割り当てられたメモリへのポインタである必要があります。バッファは処理中に繰り返し縮小され、最終的に解放されるため、この関数を呼び出した後はポインタを使用してはいけません。
(in) *uint16_t size*	メッセージのサイズ

説明：

この関数は、着信 EAPOL メッセージを処理します。対応するプリミティブ ID は R_NWK_API_MSG_EAPOL_DATA_IND です。関数は、成功した場合は R_RESULT_SUCCESS を返し、それ以外の場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは、*認証*ポインターを除き、有効であり、NULL でない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました

R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.15 R_AUTH_SUP_ProcessRelayMessage

R_AUTH_SUP_ProcessRelayMessage	受信EAPOLリレーメッセージを処理する

構文：

r_result_t R_AUTH_SUP_ProcessRelayMessage (r_apl_global_t nwk、 const uint8_t* source、const uint8_t* dest、const uint8_t* data、uint16_t size)*

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造
(in) *const uint8_t* source*	受信メッセージの IPv6 送信元アドレス
(in) *const uint8_t* dest*	受信メッセージの IPv6 宛先アドレス
(in) *const void* data*	メッセージ
(in) *uint16_t size*	メッセージのサイズ

説明：

この関数は、着信 EAPOL リレー メッセージを処理します。対応するプリミティブ ID は R_NWK_API_MSG_UDP_DATA_IND です。関数は、成功した場合は R_RESULT_SUCCESS を返し、それ以外の場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

13.1.16 R_AUTH_SUP_ClearGtkValidity

R_AUTH_SUP_ClearGtkValidity	サプリカントのGTK有効性フラグをクリアする

構文：

void R_AUTH_SUP_ClearGtkValidity (r_apl_global_t nwk);*

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造

説明：

この関数は、サプリカントの GTK 有効性フラグをクリアします。GTK 有効性フラグは、どの GTK がまだ使用できるかを示します。この関数は、それらすべてをクリアします。対応するプリミティブ ID は、R_NWK_API_MSG_AUTH_RN_CLEAR_GTK_VALIDITY です。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

13.1.17 R_AUTH_DeleteGTKs

R_AUTH_DeleteGTKs	指定されたGTKを削除する

構文：

r_result_t R_AUTH_DeleteGTKs (r_apl_global_t nwk,*
uint8_t mask)

引数:

(in) *r_apl_global_t* nwk*	グローバルネットワーク情報構造
(in) *uint8_t mask*	削除する GTK を指定するビット マスク（ビット 0 はインデックス 0 を表し、ビット 3 はインデックス 3 を表します）

説明：

この関数は、指定された GTK を削除します。指定された GTK は、MAC 層からも削除されます。内部 GTK-HASH IE はゼロになります。BR の場合、これにより、PAN Configs で送信される GTK-HASH IE も変更されます。対応するプリミティブ ID は R_NWK_API_MSG_AUTH_RN_DELETE_GTK です。この関数は、成功した場合は R_RESULT_SUCCESS を返し、そうでない場合は適切なエラー コードを返します。

制限事項:

関数に提供されるすべてのポインターは有効であり、NULL ではない必要があります。

戻り値:

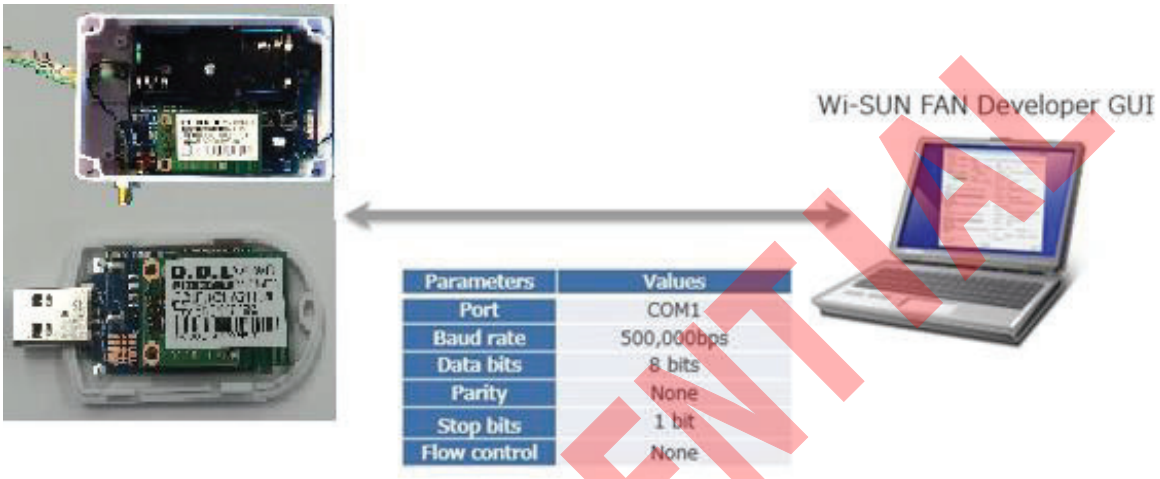
R_RESULT_SUCCESS	関数は正常に実行されました
R_RESULT_FAILED	関数の実行中にエラーが発生しました
R_RESULT_ILLEGAL_NULL_POINTER	関数に無効なヌルポインタが渡されました
R_RESULT_INVALID_PARAMETER	関数に無効な入力パラメータが渡されました
R_RESULT_INSUFFICIENT_BUFFER	メッセージを生成するためのバッファが不足しています

第14章 シリアルAPIコマンドの説明

14.1 デモンストレーターシリアルAPI

この章では、外部ツール（Wi-SUN UI など）が無線モジュール評価機とUSBドングルと通信するために使用するシリアル API コマンドについて説明します。

Wi-SUN FAN プロトコル スタック、Wi-SUN UI <-> RL78/G1H / RX65x シリアル通信 API コマンドの説明



。

図14-1 デモンストレータのシリアルインターフェース

Wi-SUN UI と RL78/G1H または RX65x デバイス間の通信は、特定の API 関数を使用するシリアル プロトコルに基づいています。すべてのシリアル コマンドは、第 14.2 章で説明されているように、シリアル HDLC エンコード フレームに埋め込まれたペイロードとして送信されます。Wi-SUN UI はノードにシリアル要求を送信し、一定のタイムアウト内に対応する確認を期待します。次の要求および確認シリアル コマンドは、シリアル API によってサポートされています。

シリアル HDLC エンコード フレームに埋め込まれたシリアル コマンド文字列の末尾には、常に改行文字 '¥n' が存在する必要があることに注意してください。そうでない場合、コマンドは破棄されます。

さらに、受信した UDP フレーム、ICMP エコー応答、またはフレーム サブスクリプションのデータ表示は、RL78/G1H または RX65x デバイスによって開始される専用の表示機能によって処理されます。原則として、これらの表示は Wi-SUN UI によって確認されませんでした。また、これらの表示は、第 14.2 章で説明されているように、シリアル HDLC エンコード フレームに埋め込まれたペイロードとして送信されます。

シリアル API コマンドで使用される NWK 属性とパラメータの詳細については、このドキュメントの第 10 章と第 11 章を参照してください。

14.1.1　ステータス取得

参照される NWK 属性:　*R_NWK_joinState*
R_NWK_macExtendedAddress
R_NWK_nwkIpv6Address

参照される NWK 関数:　*R_NWK_GetRequest*

ステータス取得要求コマンド: (GUI -> RX651)

```
STATUS_GETR 01 00
            | | |
            | |  -- <1 バイト> オプション、将来の使用のために予約済み
            | -- <1 バイト> ハンドル ID
            -- <文字列> リクエストコマンド
```

ステータス取得確認コマンド: (GUI <- RX651)

```
STATUS_GETC 01 00 05 7490500000000012 FE80000000000000769050000000012
            | | | |  |                  |
            | | | |  |                  -- <文字列> IPv6 アドレス
            | | | |  -- <文字列> MACアドレス
            | | |  -- <1 バイト> 参加状態情報
            | |  -- <1 バイト> 合格/不合格の戻りコード
            | -- <1 バイト> ハンドル ID
            -- <文字　 列> コマンドの確認
```

14.1.2　デバイス構成セット

この関数は、デバイス タイプ (ボーダールータまたはルータ ノード)、PAN ID と PAN サイズ、ネットワーク名、および 6 つの低 PAN MTU サイズなど、デバイスの基本構成を設定します。

参照される NWK 属性:　*R_NWK_rplデモモード*
R_NWK_sixLowpanMtu
R_NWK_deviceType

参照されるNWK関数:　*R_NWK_SetRequest*

デバイス構成設定要求コマンド: (GUI -> RX651)

```
DEVCONF_SETR 02 00 00 ACDC 0010 01 576953554E2050414E 0400 00 01
             |    |  | | |      |    |    |                 |    | |
             |    |  | | |      |    |    |                 |    | -- <1 バイト> 電源サイクルによる状態維持
             |    |  | | |      |    |    |                 |    -- <1 バイト > デモ モード設定
             |    |  | | |      |    |    |                 -- <2 バイト> 6 つの低パン MTU サイズ
             |    |  | | |      |    |    -- <文字列> ネットワーク名
             |    |  | | |      |    -- <1 バイト> UseParent_BS_IE
             |    |  | | |      -- <2 バイト> PAN サイズ
             |    |  | | -- <2 バイト> PAN ID
             |    |  | -- <1 バイト> デバイスタイプ
             |    |  -- <1 バイト> オプション、将来の使用のために予約済み
             |    -- <1 バイト> Handle ID
             -- <文字列> リクエストコマンド
```

デバイス構成設定確認コマンド: (GUI <- RX651)

```
DEVCONF_SETC 02 00
             |    | |
             |    | -- <1 バイト> 合格/不合格の戻りコード
             |    -- <1 バイト> ハンドル ID
             -- <文字列> コマンドの確認
```

14.1.3　デバイス構成の取得

この関数は、デバイスの基本構成を取得します。

参照される NWK 属性:　*R_NWK_deviceType*
R_NWK_macPANID
R_NWK_macExtendedAddress
R_NWK_nwkIpv6Address
R_NWK_rplDemoMode
R_NWK_sixLowpanMtu

参照されるNWK関数:　*R_NWK_GetRequest*

デバイス構成取得要求コマンド: (GUI -> RX651)

```
DEVCONF_GETR 03 00
             |   | |
             |   | -- <1 バイト> オプション、将来の使用のために予約済み
             |   -- <1 バイト> ハンドル ID
             -- <文字列> リクエストコマンド
```

デバイス構成取得確認コマンド: (GUI <- RX651)

```
DEVCONF_GETC 03 00 00 ACDC 0010 01 576953554E2050414E 0400 00 01 7490500000000012 FE80000000000000769050000000012
             |   | | | | |    |    | |                  |    | | | |                  |
             |   | | | | |    |    | |                  |    | | | |                  -- <文字列> IPv6 アドレス
             |   | | | | |    |    | |                  |    | | -- <文字列> MAC アドレス
             |   | | | | |    |    | |                  |    | -- <1 バイト> 電源サイクルによる状態維持
             |   | | | | |    |    | |                  |    -- <1 バイト> デモ モード設定
             |   | | | | |    |    | |                  -- <2 バイト> 6 つの低パン MTU サイズ
             |   | | | | |    |    | -- <文字列> ネットワーク名
             |   | | | | |    |    -- <1 バイト> UseParent_BS_IE
             |   | | | | |    -- <2 バイト> PAN サイズ
             |   | | | |  -- <2 バイト> PAN ID
             |   | | |  -- <1 バイト> デバイスタイプ
             |   | |  -- <1 バイト> 合格/不合格の戻りコード
             |   |  -- <1 バイト> ハンドル ID
             -- <文字列> 確認コマンド
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.4　PHY構成セット

この関数は、規制ドメイン、PHY チャネル プラン ID、PHY モード ID、送信電力、および PHY の規制モードを構成します。この機能は、CWX と TRG の両方の PHY ドライバーでサポートされています。

参照される NWK 属性:
R_NWK_phyWiSunPhyConfig
R_NWK_phyTransmitPower
R_NWK_phyRegulatoryMode
R_NWK_phyAntennaSelectTx
R_NWK_phyCcaVth
R_NWK_phyDataRate
R_NWK_DeviceMinSens

参照されるNWK関数:
R_NWK_SetRequest
R_NWK_GetRequest

PHY（FAN モード）構成設定要求コマンド:（GUI -> RX651）

```
PHYFAN_SETR 04 01 01 01 02 00 00
            | | | | | | | |
            | | | | | | |  -- <1 バイト> PHY 規制モード
            | | | | | | --  <1 バイト> PHY 送信電力
            | | | | | --  <1 バイト> PHY モード ID
            | | | | --  <1 バイト> PHY チャネル プラン ID
            | | | --  <1 バイト> PHY 規制ドメイン
            | | --  <1 バイト> idFormat、01 に設定する必要があります
            | --  <1 バイト> ハンドル ID
            --  <文字列> リクエストコマンド
```

PHY（FAN モード）構成設定確認コマンド:（GUI <- RX651）

```
PHYFAN_SETC 04 00
            | | |
            | |  -- <1 バイト> 合格/不合格の戻りコード
            | --  <1 バイト> ハンドル ID
            --  <文字列> コマンドの確認
```

PHY 構成セット（レガシー）

この関数は、PHY の周波数帯域、動作クラス、動作モード、送信電力、および規制モードを構成します。この機能は、TRG PHY ドライバーでのみサポートされます。

参照される NWK 属性: *R_NWK_phyWiSunPhyConfig*
R_NWK_phyTransmitPower
R_NWK_phyRegulatryMode
R_NWK_phyAntennaSelectTx
R_NWK_phyCcaVth
R_NWK_phyDataRate
R_NWK_deviceMinSens

参照される NWK 関数: *R_NWK_SetRequest*
R_NWK_GetRequest

PHY（FAN モード）構成設定要求コマンド:（GUI -> RX651）

```
PHYFAN_SETR 04 00 01 01 1B 0B 00
            | | | | | | | |
            | | | | | | | -- <1 バイト> PHY 規制モード
            | | | | | | -- <1 バイト> PHY 送信電力
            | | | | | -- <1 バイト> PHY 動作モード
            | | | | -- <1 バイト> PHY 動作クラス
            | | | -- <1 バイト> PHY 規制ドメイン
            | | -- <1 バイト> idFormat、00 に設定する必要があります
            | -- <1 バイト> ハンドル ID
            -- <文字列> リクエストコマンド
```

PHY（FAN モード）構成設定確認コマンド:（GUI <- RX651）

```
PHYFAN_SETC 04 00
            | | |
            | | -- <1 バイト> 合格/不合格の戻りコード
            | -- <1 バイト> ハンドル ID
            -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.5　PHY構成の取得

この関数は、規制ドメイン、PHY チャネル プラン ID、PHY モード ID、送信電力、規制モードなどの PHY の構成を返します。この関数は、CWX と TRG の両方の PHY ドライバーでサポートされています。

参照される NWK 属性:　*R_NWK_phyWiSunPhyConfig*
R_NWK_phyTransmitPower
R_NWK_phyRegulatoryMode

参照される NWK 関数:　*R_NWK_GetRequest*

PHY（FAN モード）構成取得要求コマンド:（GUI -> RX651）

```
PHYFAN_GETR 05 00
            | | |
            | |  ― <1 バイト> オプション、将来の使用のために予約済み
            | ― <1 バイト> ハンドル ID
            ― <文字列> リクエストコマンド
```

PHY（FAN モード）構成取得確認コマンド:（GUI <- RX651）

```
PHYFAN_GETC 05 00 01 01 02 0B 00
            | | | | | | | |
            | | | | | | |  ― <1 バイト> PHY 規制モード
            | | | | | |  ― <1 バイト> PHY 送信電力
            | | | | |  ― <1 バイト> PHY モード ID
            | | | |  ― <1 バイト> PHY チャネル プラン ID
            | | |  ― <1 バイト> PHY 規制ドメイン
            | |  ― <1 バイト> 合格/不合格の戻りコード
            | ― <1 バイト> ハンドル ID
            ― <文字列> コマンドの確認
```


DO NOT DUPLICATE 無断複製
DDL CONFIDENTIAL AND PROPRIETARY

PHY 構成の取得（レガシー）

この関数は、周波数帯域、動作モード、動作クラス、送信電力、規制モードなどの PHY の構成を返します。
この関数はTRG PHYドライバでのみサポートされています。この関数は、PHY構成が以下の条件に基づいて行われた場合にのみ、動作モードと動作クラスを返します。 動作モードと動作クラスのパラメータ。

参照される NWK 属性: *R_NWK_phyWiSunPhyConfig*
R_NWK_phyTransmitPower
R_NWK_phyRegulatoryMode

参照される NWK 関数: *R_NWK_GetRequest*

PHY（FAN モード）構成取得要求コマンド：(GUI -> RX651)

```
PHYFAN_GETR 05 00
            | | |
            | |  -- <1 byte> option, reserved for future use
            |  -- <1 byte> Handle ID
            -- <string> Request Command
```

PHY（FAN モード）構成取得確認コマンド：(GUI <- RX651)

```
PHYFAN_GETC 05 00 01 01 1B 0B 00
            | | | | | | | |
            | | | | | | |  -- <1 byte> PHY Regulatory Mode
            | | | | | | |  -- <1 byte> PHY Tx Power
            | | | | |  |  -- <1 byte> PHY Operating Mode
            | | | |  |  -- <1 byte> PHY Operating Class
            | | |  | -- <1 byte> PHY Regulatory Domain
            | |  -- <1 byte> pass/fail return code
            |  -- <1 byte> Handle ID
            -- <string> Confirm Command
```

14.1.6　ユニキャストスケジュールセット

この関数は、デバイスのユニキャスト スケジュールを設定します。

参照される NWK 属性:　*R_NWK_phyCurrentChannel*
R_NWK_nwkSchedule_Unicast

参照される NWK 関数:　***R_NWK_SetRequest***

ユニキャストスケジュールセット要求コマンド: (GUI -> RX651)

```
UCSCH_SETR 06 00 FA FF 00 00 00 00 01 01 000DC438 00 81 00
           | | |  |  |  | |  | | | |  |          | |  |
           | | |  |  |  | |  | | | |  |          | |  |-- <1 バイト> 固定チャネル
           | | |  |  |  | |  | | | |  |          | |-- <1 バイト> チャネル数
           | | |  |  |  | |  | | | |  |          |-- <1 バイト> チャネル間隔
           | | |  |  |  | |  | | | |  |-- <4 バイト> チャネル 0
           | | |  |  |  | |  | | | |-- <1 バイト> 動作クラス
           | | |  |  |  | |  | | |-- <1 バイト> 規制ドメイン
           | | |  |  |  | |  | |  -- <1 バイト> 除外チャネル制御
           | | |  |  |  | |  |  -- <1 バイト> チャネル関数
           | | |  |  |  | |  -- <1 バイト> チャネル プラン
           | | |  |  |  |  -- <1 バイト> タイミング精度
           | | |  |  |  -- <1 バイト> クロックドリフト
           | | |  |  -- <1 バイト> 滞留間隔
           | |  -- <1 バイト> オプション、将来の使用のために予約済み
           |-- <1 バイト> ハンドル ID
           -- <文字列> リクエストコマンド
```

例1: 除外チャネル範囲を含むユニキャストスケジュールセット要求コマンド:

```
UCSCH_SETR 06 00 FA FF 00 00 02 01 01 01 000DC438 00 81 00 04 04 06 08 09 0A 0D 7E 7F
                                  |                       | | | | | | | | |-- <1 バイト> 終了チャネル番号 (範囲 4)
                                  |                       | | | | | | |  -- <1 バイト> 開始チャネル番号 (範囲 4)
                                  |                       | | | | | |  -- <1 バイト> 終了チャネル番号 (範囲 3)
                                  |                       | | | | |  -- <1 バイト> 開始チャネル番号 (範囲 3)
                                  |                       | | | |  -- <1 バイト> 終了チャネル番号 (範囲 2)
                                  |                       | | |  -- <1 バイト> 開始チャネル番号 (範囲 2)
                                  |                       | |  -- <1 バイト> 終了チャネル番号 (範囲 1)
                                  |                       |  -- <1 バイト> 開始チャネル番号 (範囲 1)
                                  |                       -- <1 バイト> 範囲の数、つまり 4
                                  -- <1 バイト> 除外チャネル制御 = 範囲の変更
```

例 2: 除外チャネル マスクを含むユニキャスト スケジュール セット要求コマンド:

```
UCSCH_SETR 06 00 FA FF 00 00 02 02 01 01 000DC438 00 81 00 FE FF 81 06 00 00 00 00 00 00 00 00 00 00 00 00 00
                                   |                          | | | | | | | | | | | | | | | | | | | | | |
                                   |                          |-----------------------------------------
                                   |                          |
                                   |                          --<n バイト> n = チャネルプラン内のチャネル数/8 を切り上げた値
                                   -- <1 バイト> 除外チャネル制御 = マスクの変更
```

除外されたチャネル範囲を含むユニキャスト get コマンドと、チャネル マスク情報を除外するブロードキャスト get/set コマンドの形式は同一であることに注意してください。

ユニキャストスケジュール設定確認コマンド: (GUI <- RX651)

```
UCSCH_SETC 06 00
           | |  |
           | |  -- <1 バイト> 合格/不合格の戻りコード
           | -- <1 バイト> ハンドル ID
           -- <文字列> コマンドの確認
```

14.1.7　ユニキャストスケジュール取得

この関数は、デバイスに設定されたユニキャスト スケジュールを返します。

参照される NWK 属性:　R_NWK_nwkSchedule_Unicast

参照される NWK 関数:　R_NWK_GetRequest

ユニキャストスケジュール取得要求コマンド: (GUI -> RX651)

```
UCSCH_GETR 07 00
           | | |
           | |  -- <1 バイト> オプション、将来の使用のために予約済み
           |  -- <1 バイト> ハンドル ID
           -- <文字列> リクエストコマンド
```

ユニキャストスケジュール取得確認コマンド: (GUI <- RX651)

```
UCSCH_GETC 07 00 FA FF 00 00 00 00 01 01 000DC438 00 81 00
           | | |  |  |  |  |  |  |  |  |  |         |  |  |
           | | |  |  |  |  |  |  |  |  |  |         |  |  |-- <1 バイト> 固定チャネル
           | | |  |  |  |  |  |  |  |  |  |         |  |-- <1 バイト> チャネル数
           | | |  |  |  |  |  |  |  |  |  |         |-- <1 バイト> チャネル間隔
           | | |  |  |  |  |  |  |  |  |  |-- <4 バイト> チャネル 0
           | | |  |  |  |  |  |  |  |  |-- <1 バイト> 動作クラス
           | | |  |  |  |  |  |  |  |-- <1 バイト> 規制ドメイン
           | | |  |  |  |  |  |  |-- <1 バイト> 除外チャネル制御
           | | |  |  |  |  |  |-- <1 バイト> チャネル関数
           | | |  |  |  |  |-- <1 バイト> チャネル プラン
           | | |  |  |  |-- <1 バイト> タイミング精度
           | | |  |  |-- <1 バイト> クロックドリフト
           | | |  |-- <1 バイト> 滞留間隔
           | |  -- <1 バイト> 合格/不合格の戻りコード
           |  -- <1 バイト> ハンドル ID
           -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.8　Broadcastスケジュール設定

この機能は、デバイスのブロードキャスト スケジュールを設定します。ブロードキャスト スケジュールの設定はボーダールーターにのみ必要であることに注意してください。

参照される NWK 属性:　*R_NWK_phyCurrentChannel*
R_NWK_nwkSchedule_Broadcast

参照される NWK 関数:　*R_NWK_SetRequest*

ブロードキャストスケジュール設定要求コマンド: (GUI -> RX651)

```
BCSCH_SETR 08 00 00000190 0001 78 FF 00 00 00 00 01 01 000DC438 00 81 00
           | | | |          |    | | | | | | | | |          | | |
           | | | |          |    | | | | | | | | |          | | |-- <1 バイト> 固定チャネル
           | | | |          |    | | | | | | | | |          | |-- <1 バイト> チャネル数
           | | | |          |    | | | | | | | | |          |-- <1 バイト> チャネル間隔
           | | | |          |    | | | | | | | | |          |-- <4 バイト> チャネル 0
           | | | |          |    | | | | | | | |  |-- <1 バイト> 動作クラス
           | | | |          |    | | | | | | |  |-- <1 バイト> 規制ドメイン
           | | | |          |    | | | | | |  |-- <1 バイト> 除外チャネル制御
           | | | |          |    | | | | |  |-- <1 バイト> チャネル関数
           | | | |          |    | | |  |-- <1 バイト> チャネル プラン
           | | | |          |    | |  |-- <1 バイト> タイミング精度
           | | | |          |    |  |-- <1 バイト> クロックドリフト
           | | | |          |    |-- <1 バイト> 滞留間隔
           | | | |          |-- <2 バイト> ブロードキャストスケジュール識別子
           | | |  |-- <4 バイト> ブロードキャスト間隔
           | |  |-- <1 バイト> オプション、将来の使用のために予約済み
           |-- <1 バイト> ハンドル ID
           -- <文字列> リクエストコマンド
```

放送スケジュール設定確認コマンド: (GUI <- RX651)

```
BCSCH_SETC 08 00
           | | |
           | |  |-- <1 バイト> 合格/不合格の戻りコード
           |-- <1 バイト> ハンドル ID
           -- <文字列> コマンドの確認
```

14.1.9　Broadcastスケジュールの取得

この関数は、デバイスの現在のブロードキャスト スケジュールを返します。

参照される NWK 属性:　*R_NWK_nwkSchedule_Broadcast*

参照される NWK 関数:　*R_NWK_GetRequest*

ブロードキャストスケジュール取得要求コマンド: (GUI -> RX651)

```
BCSCH_GETR 09 00
           | | |
           | |  -- <1 バイト> オプション、将来の使用のために予約済み
           | -- <1 バイト> ハンドル ID
           -- <文字列> リクエストコマンド
```

ブロードキャストスケジュール取得確認コマンド: (GUI <- RX651)

```
BCSCH_GETC 09 00 00000190 0001 78 FF 00 00 00 00 01 01 000DC438 00 81 00
           | | |  |        |    | |  |  |  |  |  |  |         | |  |
           | | |  |        |    | |  |  |  |  |  |  |         | |  -- <1 バイト> 固定チャネル
           | | |  |        |    | |  |  |  |  |  |  |         | |-- <1 バイト> チャネル数
           | | |  |        |    | |  |  |  |  |  |  |         |-- <1 バイト> チャネル間隔
           | | |  |        |    | |  |  |  |  |  |  |-- <4 バイト> チャネル 0
           | | |  |        |    | |  |  |  |  |  |-- <1 バイト> 動作クラス
           | | |  |        |    | |  |  |  |  |-- <1 バイト> 規制ドメイン
           | | |  |        |    | |  |  |  |-- <1 バイト> 除外チャネル制御
           | | |  |        |    | |  |  |-- <1 バイト> チャネル関数
           | | |  |        |    | |  |-- <1 バイト> チャネル プラン
           | | |  |        |    | |-- <1 バイト> タイミング精度
           | | |  |        |    |-- <1 バイト> クロックドリフト
           | | |  |        |-- <1 バイト> 滞留間隔
           | | |  |-- <2 バイト> ブロードキャストスケジュール識別子
           | | |-- <4 バイト> ブロードキャスト間隔
           | |-- <1 バイト> 合格/不合格の戻りコード
           |-- <1 バイト> ハンドル ID
           -- <文字列> コマンドの確認
```

14.1.10　IPv6 アドレスセット

この機能は、ボーダールータ デバイス上の FAN ネットワーク インターフェイスの静的 IPv6 構成を実行します。注: この機能は、ボーダールータ構成でのみサポートされます。

IPv6 アドレス設定要求コマンド: (GUI -> RX651)

```
IPADDR_SETR 00 20170000000000000000000000000001
            | | |
            | |  -- <16 バイト> BR FAN インターフェースの IPv6 アドレス (プレフィックス長は常に 64 ビット)
            | -- <1 バイト> ハンドル ID
            -- <文字列> リクエストコマンド
```

IPv6 アドレス設定確認コマンド: (GUI <- RX651)

```
IPADDR_SETC 00 00
            | | |
            | |  -- <1 バイト> 合格/不合格の戻りコード
            | -- <1 バイト> ハンドル ID
            -- <文字列> コマンドの確認
```

14.1.11　外部認証サーバーセット

この機能は、各 Wi-SUN ノードの EAP-TLS 交換を実行するために使用する外部認証サーバーを設定します。注: この機能は、統合 EAP-TLS サーバーが無効になっているボーダールーター構成でのみサポートされます。

外部認証サーバー設定要求コマンド: (GUI -> RX651)

```
AUTHSERVER_SETR 01 FD00CAFECAFE00000000000000000001 6D79536563726574
                | | |                                  |
                | | |                                  -- <文字列> 認証サーバーのシークレット
                | | -- <16 バイト> 外部認証サーバーの GUA/ULA
                | -- <1 バイト> ハンドル ID
                -- <文字列> リクエストコマンド
```

外部認証サーバー設定確認コマンド: (GUI <- RX651)

```
AUTHSERVER_SETC 00 00
                | | |
                | |  -- <1 バイト> 合格/不合格の戻りコード
                | -- <1 バイト> ハンドル ID
                -- <文字列> コマンドの確認
```

14.1.12　外部 DHCPv6 サーバー セット

この機能は、各 Wi-SUN ノードの IPv6 アドレスを取得するために使用する外部 DHCPv6 サーバーを設定します。注: この機能は、ボーダールータ設定でのみサポートされます。

外部 DHCPv6 サーバー設定要求コマンド: (GUI -> RX651)

```
DHCPV6SERVER_SETR 02 FD00CAFECAFE00000000000000000001
                  | | |
                  | |  -- <16 バイト> 外部 DHCPv6 サーバーの GUA/ULA
                  | -- <1 バイト> ハンドル ID
                  -- <文字列> リクエストコマンド
```

外部 DHCPv6 サーバー設定確認コマンド: (GUI <- RX651)

```
DHCPV6SERVER_SETC 00 00
                  | | |
                  | |  -- <1 バイト> 合格/不合格の戻りコード
                  | -- <1 バイト> ハンドル ID
                  -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.13　ホワイトリスト設定

この機能は、デバイスのホワイトリストを設定します。すべてのエントリを空のままにすると（デフォルト設定）、ホワイトリスト機能が無効になります。MAC アドレスをホワイトリストに追加すると、登録された MAC アドレスから受信したフレームのみがデバイスによって受信され、処理されます。その他のフレームはすべて破棄されます。

参照される NWK 属性: ***R_NWK_macWhitelist***

参照される NWK 関数: ***R_NWK_SetRequest***

ホワイトリスト設定要求コマンド: (GUI -> RX651)

```
WHTLST_SETR 0A 00 7490500000000000 7490500000000001 7490500000000002 7490500000000003 7490500000000004 7490500000000005
            | | | 7490500000000006 7490500000000007 7490500000000008 7490500000000009 749050000000000A 749050000000000B
            | | | 749050000000000C 749050000000000D 749050000000000E 749050000000000F
            | | | |                |                |                |                |                |
            | | | ------------------------------------------------------------------------------------------
            | | | |
            | | |  -- <16 x 文字列> MAC アドレス
            | |  -- <1 バイト> オプション、将来の使用のために予約済み
            |  -- <1 バイト> ハンドル ID
             -- <文字列> リクエストコマンド
```

ホワイトリスト設定確認コマンド: (GUI <- RX651)

```
WHTLST_SETC 0A 00
            | | |
            | |  -- <1 バイト> 合格/不合格の戻りコード
            |  -- <1 バイト> ハンドル ID
             -- <文字列> コマンドの確認
```


DO NOT DUPLICATE 無断複製
DDL CONFIDENTIAL AND PROPRIETARY

14.1.14　　ホワイトリストを取得

この関数はデバイスのホワイトリストを返します。

参照される NWK 属性:　　*R_NWK_macWhitelist*

参照される NWK 関数:　　*R_NWK_GetRequest*

ホワイトリスト取得要求コマンド: (GUI -> RX651)

```
WHTLST_GETR 0B 00
            | |  |
            | |  |-- <1 バイト> オプション、将来の使用のために予約済み
            | -- <1 バイト> ハンドル ID
            -- <文字列> リクエストコマンド
```

ホワイトリスト取得確認コマンド: (GUI <- RX651)

```
WHTLST_GETC 0B 00 7490500000000000 7490500000000001 7490500000000002 7490500000000003 7490500000000004 7490500000000005
            | |  |  7490500000000006 7490500000000007 7490500000000008 7490500000000009 749050000000000A 749050000000000B
            | |  |  749050000000000C 749050000000000D 749050000000000E 749050000000000F
            | |  |  |                |                |                |                |                |
            | |  |  ------------------------------------------------------------------------------------
            | |  |  |
            | |  |  -- <16 x 文字列> MAC アドレス
            | |  -- <1 バイト> 合格/不合格の戻りコード
            | -- <1 バイト> ハンドル ID
            -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.15　開始リクエスト

この機能はデバイスを起動します。ボーダールータとして動作している場合、内部認証および DHCP サーバ、RPL モジュールが起動され、デバイスは参加状態 5「動作中」に変わります。ルータ ノード構成の場合、デバイスは参加状態 1「PAN を選択」に変わり、参加プロセス フローを開始します。

参照される NWK 属性:　*R_NWK_デバイスタイプ*
R_NWK_mac拡張アドレス
R_NWK_macPANID

参照されるNWK関数:　*R_NWK_Getリクエスト*
R_NWK_参加リクエスト
R_NWK_開始リクエスト

開始要求コマンド: (GUI -> RX651)

```
STARTR 0B 0000
       | |  |
       | |  -- <2 バイト> オプション、将来の使用のために予約済み
       | -- <1 バイト> ハンドル ID
       -- <文字列> リクエストコマンド
```

開始確認コマンド: (GUI <- RX651)

```
STARTC 0B 00 00 ACDC 576953554E2050414E 7490500000000012
       | |  | |  |      |                      |
       | |  | |  |      |                      -- <文字列> MAC アドレス
       | |  | |  |      -- <文字列> ネットワーク名
       | |  | |  -- <2 バイト> PAN ID
       | |  | -- <1 バイト> デバイスタイプ
       | |  -- <1 バイト> 合格/不合格の戻りコード
       | -- <1 バイト> ハンドル ID
       -- <文字列> コマンドの確認
```

14.1.16　リセット要求

この関数はデバイスとすべての内部状態をリセットします。デバイスのステータスは Join State 0、「リセット」に変わります。

参照される NWK 関数:　　*R_NWK_GetRequest*

リセット要求コマンド: (GUI -> RX651)

RSTR 0C 0000
　| | |
　| | -- <2 バイト> オプション、将来の使用のために予約済み
　| -- <1 バイト> ハンドル ID
　-- <文字列> リクエストコマンド

リセット確認コマンド: (GUI <- RX651)

RSTC 0C 00
　| | |
　| | -- <1 バイト> 合格/不合格の戻りコード
　| -- <1 バイト> ハンドル ID
　-- <文字列> コマンドの確認

14.1.17　フラッシュフォーマットリクエスト

この機能は、このデバイスの不揮発性メモリをフォーマットします。つまり、内部に保存されているすべてのデータが削除されます。

フラッシュフォーマット要求コマンド: (GUI -> RX651)

FFORMATR 0C 0000
　　| |　|
　　| |　── <2 バイト> オプション、将来の使用のために予約済み
　　| ── <1 バイト> ハンドル ID
　　── <文字列> リクエストコマンド

フラッシュフォーマット確認コマンド: (GUI <- RX651)

FFORMATC 0C 00
　　| |　|
　　| |　── <1 バイト> 合格/不合格の戻りコード
　　| ── <1 バイト> ハンドル ID
　　── <文字列> コマンドの確認

14.1.18　フレームサブスクリプションセット

この機能により、フレーム サブスクリプションのサポートが有効になります。この機能は、テスト ベッド ユニットとして使用されるデバイスでのみ使用できることに注意してください。

参照される NWK 属性: *R_NWK_macFrameSubscriptionEnabled*

参照される NWK 関数: *R_NWK_SetRequest*

フレームサブスクリプションセット要求コマンド: (GUI -> RX651)

```
SUBSCP_SETR 0D 00 01
            | |  | |
            | |  |  -- <1 バイト> サブスクリプションの有効化/無効化
            | |  -- <1 バイト> オプション、将来の使用のために予約済み
            | -- <1 バイト> ハンドル ID
            -- <文字列> リクエストコマンド
```

フレームサブスクリプションセット確認コマンド: (GUI <- RX651)

```
SUBSCP_SETC 0D 00
            | |  |
            | |  -- <1 バイト> 合格/不合格の戻りコード
            | -- <1 バイト> ハンドル ID
            -- <文字列> コマンドの確認
```

フレームサブスクリプション表示コマンド: (GUI <- RX651)

```
SUBMSGI 0D 00 001A 10810000112233445566770301020002020101030500000000005
        | |  |  |    |
        | |  |  |     -- <(n) バイト> フレーム
        | |  |  -- <2 バイト> フレーム長 (n)
        | |  -- <1 バイト> 合格/不合格情報
        | -- <1 バイト> ハンドル ID
        -- <文字列> 指示コマンド
```


DO NOT DUPLICATE 無断複製
DDL CONFIDENTIAL AND PROPRIETARY

14.1.19　詳細モードの設定

この関数は、デバイスの詳細なログ レベルを設定します。サポートされている詳細なログ レベルは、無効、エラー、警告、デバッグ メッセージです。

参照される NWK 属性:　*R_NWK_logVerbosity (2 チップ構成のみ)*

参照される NWK 関数:　*R_NWK_SetRequest*

詳細モード設定要求コマンド: (GUI -> RX651)

```
VERBOSE_SETR 0E 00 03
             | | |  |
             | | |  -- <1 バイト> 詳細ログのレベル、0=無効、1=エラー、2=警告、3=デバッグ メッセージ
             | |  -- <1 バイト> オプション、将来の使用のために予約済み
             |  -- <1 バイト> ハンドル ID
             -- <文字列> リクエストコマンド
```

詳細モード設定確認コマンド: (GUI <- RX651)

```
VERBOSE_SETC 0E 00 1043F1ABE5486FB9D3421E5318265A4F
             | |  | |
             | |  |  -- <16 バイト> RLog エンコーダー バージョン ID
             | |   -- <1 バイト> 合格/不合格の戻りコード
             |  -- <1 バイト> ハンドル ID
             -- <文字列> コマンドの確認
```

14.1.20　UDP送信リクエスト

この関数は、UDP 送信要求を開始します。

参照される NWK 関数:　　*R_UDP_DataRequest*

UDP 送信要求コマンド: (GUI -> RX651)

```
UDPSR 0F 20170000000000000769050000000B0A 20170000000000007690500000000001 0E1A 0E1A 00 0008 0102030405060708 0000 00 00
      | | |                                |                                |    |    |  |  |                |    | |
      | | |                                |                                |    |    |  |  |                |    | |-- <1 バイト> PhyModeID
      | | |                                |                                |    |    |  |  |                |    |-- <1 バイト> MDR 選択
      | | |                                |                                |    |    |  |  |                |-- <2 バイト> オプション
      | | |                                |                                |    |    |  |  |-- <(n) バイト> ペイロード
      | | |                                |                                |    |    |  |-- <2 バイト> ペイロード サイズ (n)
      | | |                                |                                |    |    |-- <1 バイト> フレーム交換モード DFE/EDFE
      | | |                                |                                |    |-- <2 バイト> 送信元ポート
      | | |                                |                                |-- <2 バイト> 宛先ポート
      | | |                                |-- <文字列> IPv6 送信元アドレス
      | | |-- <文字列> IPv6 宛先アドレス
      | |-- <1 バイト> ハンドル ID
      |-- <文字列> リクエストコマンド
```

UDP 送信確認コマンド: (GUI <- RX651)

```
UDPSC 0F 00
      | | |
      | | |-- <1 バイト> 合格/不合格の戻りコード
      | |-- <1 バイト> ハンドル ID
      |-- <文字列> コマンドの確認
```

14.1.21　UDPデータ表示

この関数は、UDP フレームの受信を示します。

UDP データ表示コマンド: (GUI <- RX651)

```
RCVI 0F 00 0011 20170000000000000769050000000B0A 20170000000000007690500000000001 0E1A 0E1A 0004 01020304 0080 7E
     ||  |  |     |                                |                                |    |    |    |    |    |
     ||  |  |     |                                |                                |    |    |    |    |    -- <1バイト> lqi
     ||  |  |     |                                |                                |    |    |    |    -- <2 バイト> 更新ステータス
     ||  |  |     |                                |                                |    |    |    -- <(n) バイト> ペイロード
     ||  |  |     |                                |                                |    |    -- <2 バイト> ペイロード サイズ (n)
     ||  |  |     |                                |                                |    -- <2 バイト> 送信元ポート
     ||  |  |     |                                |                                -- <2 バイト> 宛先ポート
     ||  |  |     |                                -- <文字列> IPv6 送信元アドレス
     ||  |  |     -- <文字列> IPv6 宛先アドレス
     ||  |  -- <2 バイト> IPv6 プロトコル ID (0x11 = UDP)
     ||  -- <1 バイト> 合格/不合格情報
     |-- <1 バイト> ハンドル ID
     -- <文字列> 指示コマンド
```


複製禁
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.22　ICMPv6エコー送信

この関数は、ICMPv6 エコー送信要求を開始します。

参照される NWK 関数:　　*R_NWK_IcmpEchoRequest*

ICMPv6 エコー要求送信コマンド: (GUI -> RX651)

```
ICMPSR 10 20170000000000007690500000000B0A 20170000000000007690500000000001 FF 80 0000 0000 00 0004 01020304 0000 00 00
       | | |                                |                                | | | |    | |  |         |    | |  |
       | | |                                |                                | | | |    | |  |         |    | |  |-- <1 バイト> PhyModeID
       | | |                                |                                | | | |    | |  |         |    |  -- <1 バイト> MDR 選択
       | | |                                |                                | | | |    | |  |         |     -- <2 バイト> オプション
       | | |                                |                                | | | |    | |  |          -- <(n) バイト> ペイロード
       | | |                                |                                | | | |    | |   -- <2 バイト> ペイロード サイズ (n)
       | | |                                |                                | | | |    |  -- <1 バイト> 交換モード DFE/EDFE
       | | |                                |                                | | | |     -- <<2 バイト> シーケンス
       | | |                                |                                | | |  -- <2 バイト> 識別子
       | | |                                |                                |  -- <1 バイト> フレーム タイプ ICMP エコー要求
       | | |                                |                                -- <1 バイト> ホップ制限
       | | |                                -- <文字列> IPv6 送信元アドレス
       | |  -- <文字列> IPv6 宛先アドレス
       |  -- <1 バイト> ハンドル ID
       -- <文字列> リクエストコマンド
```

ICMPv6 Send Confirm Command: (GUI <- RX651)

```
ICMPSC 0F 00
       | | |
       | |  -- <1 byte> 合格/不合格の戻りコード
       |  -- <1 byte> Handle ID
       -- <string> コマンドの確認
```

14.1.23　ICMPエコー応答表示

この関数は、ICMP エコー応答フレームの受信を示します。

ICMPエコー応答表示コマンド: (GUI <- RX651)

```
RCVI 02 00 003A 20170000000000000769050000000001 20170000000000000769050000000B0A 81 0000 0000 0002 6162 0080 7D
     || | |    |                                |                                | |    |    |    |    |    |
     || | |    |                                |                                | |    |    |    |    |    -- <1バイト> lqi
     || | |    |                                |                                | |    |    |    |    -- <2 バイト> icmp ステータス
     || | |    |                                |                                | |    |    |    -- <(n) バイト> ペイロード
     || | |    |                                |                                | |    |    -- <2 バイト> ペイロード サイズ (n)
     || | |    |                                |                                | |    -- <2 バイト> シーケンス
     || | |    |                                |                                |  -- <2 バイト> 識別子
     || | |    |                                |                                -- <1 バイト> ICMP タイプ エコー応答
     || | |    |                                -- <文字列> IPv6 送信元アドレス
     || | |    -- <文字列> IPv6 宛先アドレス
     || |  -- <2 バイト> IPv6 プロトコル ID (0x3A = ICMP)
     ||  -- <1 バイト> 合格/不合格情報
     |-- <1 バイト> ハンドル ID
     -- <文字列> 指示コマンド
```


複製禁止
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.24　RPL情報取得

この関数は、デバイスからさまざまな RPL 情報を取得します。

参照される NWK 属性:　　*R_NWK_rplInfo*

RPL 情報取得要求

コマンド: (GUI -> RX651)

```
RPLINFO_GETR 12 00
             | | |
             | |  ― <1 バイト> オプション、将来の使用のために予約済み
             |  ― <1 バイト> ハンドル ID
              ― <文字列> リクエストコマンド
```

RPL 情報取得確認

コマンド: (GUI <- RX651)

```
RPLINFO_GETC 19 00 00 20170000000000000769050000000000 F0 00 01 80 F0
             | | | | |                                 |  |  | | |
             | | | | |                                 |  |  | | |― <1 バイト> DTSN
             | | | | |                                 |  |  |  ― <1 バイト> ランク
             | | | | |                                 |  |   ― <1 バイト> 動作モード
             | | | | |                                 |   ― <1 バイト> DODAG 設定
             | | | | |                                  ― <1 バイト> DODAG バージョン
             | | | |   ― <16 バイト> DODAG ID
             | | |  ― <1 バイト> インスタンス ID
             | |  ― <1 バイト> 合格/不合格の戻りコード
             |  ― <1 バイト> ハンドル ID ― <文字列>
確認コマンド
```

14.1.25　　ルートリストの取得

この関数は、デバイスのルーティング リストを取得します。ボーダールータとして動作している場合は、ソース ルートのリストが返されます。ルータ ノードとして動作している場合は、優先される親と、該当する場合は代替の親が返されます。

参照される NWK 属性:　　*R_NWK_sourceRoutes*
R_NWK_preferredParentAddress
R_NWK_alternateParentAddress
R_NWK_nwkIpv6Address

参照されるNWK関数:　　*R_NWK_GetRequest*
R_NWK_GetRequestMultipart

ルートリスト取得要求コマンド: (GUI -> RX651)

```
ROUTELIST_GETR 11 00
               | | |
               | |  -- <1 バイト> オプション、将来の使用のために予約済み
               | -- <1 バイト> ハンドル ID
               -- <文字列> リクエストコマンド
```

ルートリスト取得確認コマンド: (GUI <- RX651)

```
ROUTELST_GETC 11 00 01 20170000000000000769050000000000001 20170000000000000769050000000B0A 00000000000000000000000000000000 FFFFFFFF
               | | | |  |                                  |                                 |                                 |
               | | | |  |                                  |                                 |                                 -- <8 バイト>
               | | | |  |                                  |                                 |                                    Lifetime
               | | | |  |                                  |                                 -- <文字列> 代替親アドレス
               | | | |  |                                  -- <文字列> 優先親アドレス
               | | | |  -- <文字列> 宛先アドレス
               | | |  -- <1 バイト> ルート情報 (0x01 = アップリンク、0x03 = ダウンリンク)
               | |  -- <1 バイト> 合格/不合格の戻りコード
               | -- <1 バイト> ハンドル ID
               -- <文字列> コマンドの確認
```

例: ボーダールータで複数のソースルートを取得するルートリスト取得確認コマンドを実行する

```
ROUTELST_GETC 09 00 03 20170000000000000769050000000000002 20170000000000000769050000000B0A 00000000000000000000000000000000 00001BAA
                       |    20170000000000000769050000000B0B 20170000000000000769050000000B0A 20170000000000000769050000000002 00001BBC
                       |      20170000000000000769050000000003 20170000000000000769050000000B0A 20170000000000000769050000000002 00001BB0
                       |  |                                  |                                 |                                 |
                       |  |                                  |                                 |                                 |
                       |  ------------------------------------------------------------------------------------------------------------
                       |  |
                       |  -- ソースルート 0
                       |    -- ソースルート 1
                       |      -- ソースルート 2
```

```
                -- <1 バイト> ルート情報 (0x03 = ダウンリンク)
```

14.1.26　近隣キャッシュ取得

この関数は、デバイスのネイバー キャッシュ リストを取得します。
参照される NWK 属性:　　***R_NWK_ndCache***
参照されるNWK関数:　　***R_NWK_GetRequestMultipart***

ネイバーキャッシュ取得要求コマンド: (GUI -> RX651)

```
NDCACHE_GETR 12 00
             | | |
             | |  -- <1 バイト> オプション、将来の使用のために予約済み
             |  -- <1 バイト> ハンドル ID
             -- <文字列> リクエストコマンド
```

ネイバーキャッシュ取得確認コマンド: (GUI <- RX651)

```
NDCACHE_GETC 12 00 7490500000000B0B 0080 007D 0000 0010 0080 0080 20170000000000007690500000000B0B 2CC8 0001
             | | | |                |    |    |    |    |    |    |                                    |      |
             | | | |                |    |    |    |    |    |    |                                    |      |-- <2 バイト> ステータス
             | | | |                |    |    |    |    |    |    |                                    |-- <2-4 バイト> 最終受信秒数
             | | | |                |    |    |    |    |    |    |-- <文字列> IPv6 アドレス
             | | | |                |    |    |    |    |    |-- <2 バイト> ルーティング コスト
             | | | |                |    |    |    |    |-- <2 バイト> rpl ランク
             | | | |                |    |    |    |-- <2 バイト> パンサイズ
             | | | |                |    |    |-- <2 バイト> rssi (常にゼロ、互換性の理由からのみサポートされます)
             | | | |                |    |-- <2 バイト> ノード rsl の隣接
             | | | |                |-- <2 バイト> リンク メトリック
             | | |  -- <文字列> MACアドレス
             | |  -- <1 バイト> 合格/不合格の戻りコード
             |  -- <1 バイト> ハンドル ID
             -- <文字列> コマンドの確認
```

例: 複数のネイバー キャッシュ エントリを含むネイバー キャッシュ取得確認コマンド

```
NDCACHE_GETC 05 00 7490500000000B0B 00C5 00A1 0000 0012 0081 00A8 20170000000000007690500000000B0B   2710   0000
                   7490500000000001 03F9 007B 0000 0010 0079 0090 20170000000000007690500000000001 2EE0 0000
                     7490500000000002 04F9 0075 0000 0011 0080 0090 20170000000000007690500000000002 2D40 0000
                   |                |    |    |    |    |    |    |                                  |      |
                   |                |    |    |    |    |    |    |                                  |      |
                   ------------------------------------------------ ----------------------------------
                   |
                   -- 近隣キャッシュエントリ 0
                      -- 隣接キャッシュエントリ 1
                         -- 隣接キャッシュエントリ 2
```

14.1.27　キーの取り消し

このコマンドは、デバイス上のすべての GTK または LGTK (アクティブなものを除く) を取り消し、指定された代替 GTK をインストールします。注: このコマンドは、ボーダールータ構成でのみサポートされます。

参照される NWK 属性:　*R_NWK_joinState*
R_NWK_ndCache

参照される NWK 関数:　*R_NWK_GetRequest*
R_NWK_IncreasePanVersionRequest

キー要求コマンドの取り消し: (GUI -> RX651)

```
REVOKEKEYSR 13 00 BB0608572CE14D7BA2D155499CC8519B
            | | |  |
            | | |  -- <文字列> 他の [L]GTK が取り消された後にインストールする必要がある代替 [L]GTK
            | | -- <1 バイト> 取り消すべき GTK のタイプ (0=GTK、1=LGTKS)
            | -- <1 バイト> ハンドル ID
            -- <文字列> リクエストコマンド
```

キー確認コマンドの取り消し: (GUI <- RX651)

```
REVOKEKEYSC 13 00
            | | |
            | | -- <1 バイト> 合格/不合格の戻りコード
            | -- <1 バイト> ハンドル ID
            -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.28　　GTK セット

この関数は、デバイス上の対応する GTK を設定します。この関数は、ボーダールータ構成でのみサポートされます。

参照される NWK 属性:　　*R_NWK_joinState*

参照される NWK 関数:　　*R_NWK_GetRequest*
R_NWK_IncreasePanVersionRequest
R_AUTH_SetGTK

GTKS セット要求コマンド: (GUI -> RX651)

```
GTKS_SETR 15 00 03 0102030405060708090A0B0C0D0E0F10 BB0608572CE14D7BA2D155499CC8519B
          | | | |  |                                |
          | | | |  |                                -- <文字列> GTK 1
          | | | |  -- <文字列> GTK 0
          | | |  -- <1 バイト> キー マスク (ビット単位の GTK マスク)、つまり 0x03 -> GTK0 および GTK1
          | |  -- <1 バイト> オプション、将来の使用のために予約済み
          |  -- <1 バイト> ハンドル ID
          -- <文字列> リクエストコマンド
```

例: すべてのGTKを設定するためのGTKS設定要求コマンド

```
GTKS_SETR 15 00 0F 11111111111111111111111111111111 22222222222222222222222222222222 33333333333333333333333333333333 44444444444444444444444444444444
          | |  | |  |                                |                                |                                |
          | |  | |  |                                |                                |                                -- <文字列> GTK3
          | |  | |  |                                |                                -- <文字列> GTK2
          | |  | |  |                                -- <文字列> GTK1
          | |  | |  -- <文字列> GTK0
          | |  |  -- <1 バイト> キーマスク = 0x0F -> GTK0、GTK1、GTK2、GTK3
          | |  -- <1 バイト> オプション、将来の使用のために予約済み
          | -- <1 バイト> ハンドル ID
          -- <文字列> リクエストコマンド
```

GTKS 設定確認コマンド: (GUI <- RX651)

```
GTKS_SETC  15 00
           | | |
           | | -- <1 バイト> 合格/不合格の戻りコード
           |  -- <1 バイト> ハンドル ID
           -- <文字列> コマンドの確認
```

14.1.29　GTKs 取得

この関数はデバイスの GTK を返します。

GTKS 取得要求コマンド: (GUI -> RX651)

```
GTKS_GETR 14 00
          | | |
          | | -- <1 バイト> オプション、将来の使用のために予約済み
          | -- <1 バイト> ハンドル ID
          -- <文字列> リクエストコマンド
```

GTKS 取得確認コマンド: (GUI <- RX651)

```
GTKS_GETC 14 00 03 0102030405060708090A0B0C0D0E0F10 BB0608572CE14D7BA2D155499CC8519B
          | | | | |                                 |
          | | | | |                                 -- <文字列> GTK
          | | | | -- <文字列> GTK 0
          | | | -- <1 バイト> キー情報が有効 (ビット単位の GTK が有効)、つまり 0x03 -> GTK0 と GTK1 が有効
          | | -- <1 バイト> 合格/不合格の戻りコード
          | -- <1 バイト> ハンドル ID
          -- <文字列> コマンドの確認
```

例: すべてのGTKを取得するGTKS取得確認コマンド

```
GTKS_GETC 14 00 0F 11111111111111111111111111111111 22222222222222222222222222222222 33333333333333333333333333333333 44444444444444444444444444444444
          || | | | |                                 |                                 |                                 |
          || | | | |                                 |                                 |                                 -- <文字列> GTK3
          || | | | |                                 |                                 -- <文字列> GTK2
          || | | | |                                 -- <文字列> GTK1
          || | | | -- <文字列> GTK0
          || | | -- <1 バイト> キー情報有効 = 0x0F -> GTK0、GTK1、GTK2、GTK3
          || -- <1 バイト> オプション、将来の使用のために予約済み
          |-- <1 バイト> ハンドル ID
          -- <文字列> リクエストコマンド
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.30　　キーの有効期間の設定

この機能を使用すると、デバイスのキーの有効期間を変更できます。この機能は、ボーダールータの設定でのみサポートされます。このコマンドの個々のパラメータの詳細については、Wi-SUN FAN TPS を参照してください。

キーの有効期間設定要求コマンド：(GUI -> RX651)

```
KEYKIFETIMES_SETR 15 00 0002A300 00015180 0000000A 02D0 001E 14 000BDD80 0007E900 00000014 02D0 001E 14
                  | | | |       |        |        |    |    | |          |        |        |    |    |
                  | | | |       |        |        |    |    | |          |        |        |    |    |-- <1 バイト> LGTK_NEW_INSTALL_REQUIRED (FANTPS を参照)
                  | | | |       |        |        |    |    | |          |        |        |    |-- <2 バイト> LFN 失効有効期間短縮率
                  | | | |       |        |        |    |    | |          |        |        |-- <2 バイト> LGTK の新しいアクティベーション時間の割合
                  | | | |       |        |        |    |    | |          |        |-- <4 バイト> LTK の有効期間
                  | | | |       |        |        |    |    | |          |-- <4 バイト> LFN PTK の有効期間
                  | | | |       |        |        |    |    | |-- <4 バイト> LFN PMK の有効期間
                  | | | |       |        |        |    |    -- <1 バイト> GTK_NEW_INSTALL_REQUIRED
                  | | | |       |        |        |    -- <2 バイト> FFN 失効有効期間短縮率
                  | | | |       |        |        -- <2 バイト> GTK の新しいアクティベーション時間の割合
                  | | | |       |        -- <4 バイト> GTK の有効期間
                  | | | |       -- <4 バイト> PTK の有効期間
                  | | |  -- <4 バイト> PMK の有効期間
                  | |  -- <1 バイト> オプション、将来の使用のために予約済み
                  |  -- <1 バイト> ハンドル ID
                  -- <文字列> リクエストコマンド
```

キーの有効期間設定確認コマンド：(GUI <- RX651)

```
KEYLIFETIMES_SETC 15 00
                  | | |
                  | |  -- <1 バイト> 合格/不合格の戻りコード
                  |  -- <1 バイト> ハンドル ID
                  -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.31　IPv6 ネットワークプレフィックスセット

この関数は、デバイスが使用する IPv6 ネットワーク プレフィックスを設定します。この機能は、ボーダールータ構成でのみサポートされます。

参照される NWK 属性: ***R_NWK_joinState***

参照される NWK 関数: ***R_NWK_GetRequest***

IPv6 プレフィックス設定要求コマンド: (GUI -> RX651)

IPV6PREFIX_SETR 15 201800DEADBEEF00
| | |
| | — <8 バイト> IPv6 ネットワーク プレフィックス
| — <1 バイト> ハンドル ID
— <文字列> リクエストコマンド

IPv6 プレフィックス設定確認コマンド: (GUI <- RX651)

IPV6PREFIX_SETC 15 00
| | |
| | — <1 バイト> 合格/不合格の戻りコード
| — <1 バイト> ハンドル ID
— <文字列> コマンドの確認

14.1.32　　バージョン取得

この機能は、ハードウェア プラットフォームと、デバイス上で実行されている Wi-SUN FAN スタックおよびアプリケーションの名前、バージョン、サポートされている機能を表示します。

バージョン取得要求コマンド: (GUI -> RX651)

```
VERSION_GETR 16 00
             | | |
             | |  -- <1 バイト> オプション、将来の使用のために予約済み
             | -- <1 バイト> ハンドル ID
             -- <文字列> リクエストコマンド
```

バージョン取得確認コマンド: (GUI <- RX651)

```
VERSION_GETC 00 00 MB-RX604S-02 Wi-SUN-FAN-DemoApp V1.4.1 000F 0121 7490500000000000
             | | | |              |                    |      |    |   |
             | | | |              |                    |      |    |   ||-- <文字列> MAC アドレス
             | | | |              |                    |      |    -- <2 バイト> Wi-SUN FAN スタック機能マスク
             | | | |              |                    |      -- <2 バイト> アプリケーション機能マスク
             | | | |              |                   -- <文字列> Wi-SUN FAN スタック バージョン
             | | | |              -- <文字列> アプリケーション名
             | | |  -- <文字列> ハードウェアプラットフォーム名
             | |  -- <1 バイト> 合格/不合格の戻りコード
             | -- <1 バイト> ハンドル ID
             -- <文字列> コマンドの確認
```

14.1.33　　PIB セット

この関数は指定された PIB 属性を設定します。

参照される NWK 関数:　　*R_NWK_SetRequest*

PIB セット要求コマンド: (GUI -> RX651)

```
PIB_SETR 00 b2 01 0001 1E
        | |  | |  |    |
        | |  | |  |    -- <(n) バイト> PIB属性値
        | |  | |  -- <2 バイト> PIB 属性の長さ
        | |  |  -- <1 バイト> 数値。数値型 (char、short、int など) の場合は 01 に設定し、それ以外の場合は 00 に設定してください。
        | |  -- <1 バイト> 16 進数の PIB 属性、つまり 0xb2 -> R_NWK_panTimeout
        | -- <1 バイト> ハンドル ID
        -- <文字列> リクエストコマンド
```

例: 属性 R_NWK_sixLowpanMtu を 1576 10 進数に設定する PIB 設定要求コマンド

```
PIB_SETR 00 e0 01 0002 0628
        | |  | |  |    |
        | |  | |  |    -- <(n) バイト> PIB 属性値 16 進数 0628 -> sixLowPanMTU サイズを 1576 10 進数に設定
        | |  | |  -- <2 バイト> PIB 属性の長さ 0002 -> 2 バイト
        | |  |  -- <1 バイト> 数値、数値型 01
        | |  -- <1 バイト> PIB 属性、R_NWK_sixLowpanMtu -> 0xe0
        | -- <1 バイト> ハンドル ID
        -- <文字列> リクエストコマンド
```

PIB 設定確認コマンド: (GUI <- RX651)

```
PIB_SETC  00  00
        |  |  |
        |  |  -- <1 バイト> 合格/不合格の戻りコード
        |  -- <1 バイト> ハンドル ID
        -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.34　　PIB 取得

この関数は指定された PIB 属性を取得します。

参照される NWK 関数:　　***R_NWK_GetRequest***

PIB 取得要求コマンド: (GUI -> RX651)

```
PIB_GETR 00 e0 01
         | |  |  |
         | |  |  -- <1 バイト> 数値。数値型 (char、short、int など) の場合は 01 に設定し、それ以外の場合は 00 に設定してください。
         | |  -- <1 バイト> 16 進数の PIB 属性、つまり 0xe0 -> R_NWK_sixLowpanMtu
         | -- <1 バイト> ハンドル ID
         -- <文字列> リクエストコマンド
```

PIB 取得確認コマンド: (GUI <- RX651)

```
PIB_GETC 00 00 0002 0628
         | | | |    |
         | | | |    -- <(n) バイト> PIB 属性値を 16 進数で表したもの。つまり 0628 -> sixLowPanMTU サイズは 1576 10 進数で表したものを取得します。
         | | | -- <2 バイト> PIB 属性の長さ、つまり 2 バイト
         | | -- <1 バイト> 合格/不合格の戻りコード
         | -- <1 バイト> ハンドル ID
         -- <文字列> コマンドの確認
```

14.1.35　IPデータ要求

この関数は、生の IPv6 パケット送信要求を開始します。

参照される NWK 属性: *R_NWK_joinState*

参照される NWK 関数: *R_NWK_GetRequest*
R_NWK_IP_DataRequest

IPデータ要求コマンド: (GUI -> RX651)

```
IPSR 0F 0020 000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f 0000
    | |  |    |                                                                     |
    | |  |    |                                                                     -- <2 バイト> オプション
    | |  |    -- <(n) バイト> 生の IP パッケージ
    | |  -- <2 バイト> 生の IP パッケージ長 (n)
    | -- <1 バイト> ハンドル ID
    -- <文字列> IP データ要求コマンド
```

IP データ要求確認コマンド: (GUI <- RX651)

```
IPSC 0F 00
    |  |  |
    |  |  -- <1 バイト> 合格/不合格の戻りコード
    |  -- <1 バイト> ハンドル ID
    -- <文字列> コマンドの確認
```

14.1.36　IPデータ表示

この関数は、Wi-SUN FAN プロトコル スタックで処理できなかった生の IPv6 パケージを受信したことを示します。

IPデータ表示コマンド: (GUI <- RX651)

```
IPFI 0F 00 0020 000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f
  | |  |  |    |
  | |  |  |    -- <(n) バイト> 生の IP パケージ
  | |  |  -- <2 バイト> 生の IP パケージ長 (n)
  | |  -- <1 バイト> 合格/不合格情報
  | -- <1 バイト> ハンドル ID
  -- <文字列> IP データ表示コマンド
```

14.1.37　ログエントリ要求

この関数は、デバイスの内部ログ バッファーからログ エントリを取得します。

ログエントリ要求コマンド: (GUI -> RX651)

```
LOGENTRIES_GETR 01 00
                | | |
                | |  -- <1 バイト> オプション、将来の使用のために予約済み
                | -- <1 バイト> ハンドル ID
                -- <文字列> ログエントリ要求コマンド
```

ログエントリ要求確認コマンド: (GUI <- RX651)

```
LOGENTRIES_GETC 01 00 E300CF040000000000007490500000000003CA7EDF01CF04000000000000807E0902CF040000000000032010000787E
                | | | |
                | | |  -- <(n) バイト> HDCLエンコードされたログステートメント
                | |  -- <1 バイト> 合格/不合格情報
                |  -- <1 バイト> ハンドル ID
                -- <文字列> ログエントリ要求確認コマンド
```


DO NOT DUPLICATE 無断複製
DDL CONFIDENTIAL AND PROPRIETARY

14.1.38　RPLグローバル修復

この機能は、DODAG バージョンを増やすことで RPL のグローバル修復をトリガーします。この機能は、ボーダールータ設定でのみサポートされます。

RPL グローバル修復要求コマンド: (GUI -> RX651)

```
RPL_GLOBALREPAIRR 13
                  | |
                  | -- <1 バイト> ハンドル ID
                  -- <文字列> リクエストコマンド
```

RPL グローバル修復確認コマンド: (GUI <- RX651)

```
RPL_GLOBALREPAIRC 13 00
                  | | |
                  | |  -- <1 バイト> 合格/不合格の戻りコード
                  | -- <1 バイト> ハンドル ID
                  -- <文字列> コマンドの確認
```

14.1.39　ソースルートの更新

この関数は、すべてのルートに DAO を送信するように強制することで、すべてのソース ルートの更新をトリガーします。この関数は、ボーダールータ構成でのみサポートされます。

RPL グローバル修復要求コマンド: (GUI -> RX651)

```
RPL_ROUTES_REFRESHR 13
                    | |
                    | -- <1 バイト> ハンドル ID
                    -- <文字列> リクエストコマンド
```

RPL グローバル修復確認コマンド: (GUI <- RX651)

```
RPL_ROUTES_REFRESHC 13 00
                    | | |
                    | | -- <1 バイト> 合格/不合格の戻りコード
                    | -- <1 バイト> ハンドル ID
                    - - <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.40　ネットワークからデバイスをキックする

この機能は、対応するデバイスをネットワークからキックします。この機能は、ボーダールータ構成でのみサポートされます。

キックデバイス要求コマンド: (GUI -> RX651)

```
DEVICEKICKR 13 BB0608572CE14D7BA2D155499CC8519B
            | | |
            | |  -- <文字列> キックするルータノードの IPv6 アドレス
            | -- <1 バイト> ハンドル ID
            -- <文字列> リクエストコマンド
```

キックデバイス確認コマンド: (GUI <- RX651)

```
DEVICEKICKC 13 00
            | | |
            | |  -- <1 バイト> 合格/不合格の戻りコード
            | -- <1 バイト> ハンドル ID
            -- <文字列> コマンドの確認
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.41　ネットワークを離れる

この機能により、このデバイスは直ちに近隣デバイスに通知し、ネットワークを離れます。この機能は、ネットワークに完全に参加しているルータ ノードに対してのみサポートされます。

ネットワーク離脱要求コマンド: (GUI -> RX651)

DEVICEKICKR 13
　　| |
　　| ― <1 バイト> ハンドル ID
　　― <文字列> リクエストコマンド

ネットワーク離脱確認コマンド: (GUI <- RX651)

DEVICEKICKC 13 00
　　| | |
　　| | ― <1 バイト> 合格/不合格の戻りコード
　　| ― <1 バイト> ハンドル ID
　　― <文字列> コマンドの確認

14.1.42　参加状態更新表示

この関数は、このデバイスの FAN 参加状態の更新を示します。

参加状態更新指示コマンド：（GUI <- RX651）

```
JSI 13 05
 | | |
 | | -- <1 バイト> 新しい結合状態の値 (0-5)
 | -- <1 バイト> ハンドル ID
 -- <文字列> 指示コマンド
```

14.1.43　マルチキャスト グループ要求（参加/離脱）

この機能は、デバイスに特定の IPv6 マルチキャスト グループに参加または離脱するように指示します。

マルチキャストグループ要求コマンド：（GUI -> RX651）

```
MULTICAST_GROUPRR 13 FF030000000000000000000000000042 01
                  | | |                                  |
                  | | |                                  -- <1 バイト> True の場合、指定されたグループに参加します。False の場合、グループを脱退します。
                  | | -- <文字列> マルチキャストグループを識別するIPv6アドレス
                  | -- <1 バイト> ハンドル ID
                  -- <文字列> リクエストコマンド
```

14.1.44　マルチキャストグループ更新通知

この関数は、デバイスがグループに参加したか、グループから離脱したかなど、特定の IPv6 マルチキャスト グループの更新を示します。

マルチキャスト グループ更新表示コマンド：（GUI <- RX651）

```
MCI 07 00 FF030000000000000000000000000042
   | | | |
   | | | -- <文字列> マルチキャストグループを識別するIPv6アドレス
   | | -- <1 バイト> 更新イベント (セクション 0 の R_NWK_API_MSG_MULTICAST_GROUP_IND を参照)
   | -- <1 バイト> ハンドル ID
   -- <文字列> 指示コマンド
```


無断複製 DO NOT DUPLICATE DDL CONFIDENTIAL AND PROPRIETARY

14.1.45　ネットワークインターフェーステーブルの取得

この関数は、デバイスのネットワーク インターフェイスに関するさまざまな管理情報を取得します。個々のフィールドとその意味の詳細については、RFC 1213 を参照してください。

参照される NWK 属性:　　*R_NWK_fanInterfaceMgmtInfo*

NWK インターフェース テーブル取得要求コマンド: (GUI -> RX651)

```
NWKIFTABLE_GETR 12 00
                | | |
                | |  -- <1 バイト> オプション、将来の使用のために予約済み
                | -- <1 バイト> ハンドル ID
                -- <文字列> リクエストコマンド
```

NWK インターフェース取得確認コマンド: (GUI <- RX651)

```
NWKIFTABLE_GETC 7D 00 52656E657361732057692D53554E2046414E204E6574776F726B20496E7465726661636520562312E342E31 7490500000000001
                | |  |  |0000E4C4 0000001A 0000001F 00000000 00000000 0001060F 00000010 00000032 00000000 00000001
                | |  |  ||          |         |         |         |         |         |         |         |         |
                | |  |  ||          |         |         |         |         |         |         |         |         |-- <8 バイト> EUI-64
                | |  |  ||          |         |         |         |         |         |         |         |-- <4 バイト> ifoutQlen
                | |  |  ||          |         |         |         |         |         |         |-- <4 バイト> ifouterrors
                | |  |  ||          |         |         |         |         |         |-- <4 バイト> ifoutnucastpackets
                | |  |  ||          |         |         |         |         |-- <4 バイト> ifoutucast パケット
                | |  |  ||          |         |         |         |-- <4 バイト> ifoutoctets
                | |  |  ||          |         |         |-- <4 バイト> ifinunknownprotos
                | |  |  ||          |         |-- <4 バイト> ifinerrors
                | |  |  ||          |-- <4 バイト> ifinnucastpackets
                | |  |  ||-- <4 バイト> ifinucastpackets
                | |  |  |-- <4 バイト> ifinoctets
                | |  |  -- <16進文字列> インターフェースの説明/名前
                | |  -- <1 バイト> 合格/不合格の戻りコード
                | -- <1 バイト> ハンドル ID

                -- <文字列> 確認コマンド
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.46　新しいモードセット

この関数は、変調およびデータ レート (MDR) ネゴシエーションのためにデバイスによってサポートおよびアドバタイズされる MDR モードを設定します。

参照される NWK 属性:　R_NWK_mdrNewModes

新しいモード設定要求コマンド: (GUI -> RX651)

```
NEWMODE_SETR 01 02 05 24 54
             | | | |  | |
             | | | |  |  -- <1 バイト> PhyModeID
             | | | |  -- <1 バイト> PhyModeID
             | | |  -- <1 バイト> PhyModeID
             | |  -- <1 バイト> PhyModeID
             | -- <1 バイト> ハンドル ID
             - - <文字列> リクエストコマンド
```

新しいモード設定確認コマンド: (GUI <- RX651)

```
NEWMODE_SETC 01 00
             | |  |-- <1 バイト> 合格/不合格の戻りコード
             | |-- <1 バイト> ハンドル ID
             |-- <文字列> 確認コマンド
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.47　　CSMP 設定 セット

この 関数 セット の CSMP 構成 パラメータ それ は 使用済み に 登録する これ デバイス で 1つの CSMP サーバ (FND)。 これ 指示 は のみ CSMPが有効になっているデバイスでサポートされます。

CSMP 設定 セット リクエスト 指示： (GUI -> RX651)

```
CSMPCONFIG_SETR 01 FD00CAFECAFE00000000000000000CCCC 000A 0078
                |  |  |                                  |    |
                |  |  |                                  |    ── <2 バイト> 最大 登録 間隔
                |  |  |                                  ── <2 バイト> 最小 登録 間隔
                |  |  ── <16 バイト> IPv6 住所 の CSMP サーバ (FND)
                |  ── <1 バイト> ハンドル ID
                ── <文字列> リクエスト 指示
```

CSMP 設定 セット 確認する 指示： (GUI <- RX651)

```
CSMPCONFIG_SETC 01 00
                |  |  ── <1 バイト> 合格/不合格 戻る コード
                |  ── <1 バイト> ハンドル ID
                |── <文字列> 確認コマンド
```


無断複製
DO NOT DUPLICATE
DDL CONFIDENTIAL AND PROPRIETARY

14.1.48　　CSMP 設定 得る

この 関数 取得する の CSMP 構成 パラメータ それ は 使用済み に 登録する これ デバイス で 1つの ネットワーク 管理 サーバ (NMS)、 どれの は 通常 の 実例 の 1つの シスコ IoT FND。
これ 指示 は のみ サポートされている のために デバイス と CSMP 有効。

CSMP 設定 得る リクエスト 指示：　(GUI -> RX651)

```
CSMPCONFIG_GETR 0B 00
                |   | |-- <1 バイト> オプション、予約済み のために 未来 使用
                |   -- <1 バイト> ハンドル ID
                -- <文字列> リクエスト 指示
```

CSMP 設定 得る 確認する 指示：　(GUI <- RX651)

```
CSMPCONFIG_GETC 01 00 FD00CAFECAFE00000000000000000CCCC 000A 0078
                | | | |                                  |    |
                | | | |                                  |    -- <2 バイト> 最大 登録 間隔
                | | | |                                   -- <2 バイト> 最小 登録 間隔
                | | | -- <16 バイト> IPv6 住所 の ネットワーク 管理 サーバ (NWS)
                | | -- <1 バイト> 合格/不合格 戻る コード
                | -- <1 バイト> ハンドル ID
                -- <文字列> 確認コマンド
```

14.2 追加コマンド

ＢＲ／ＲＮ，ＬＦＮそれぞれにコマンドを追加している。以下にコマンド例を記述する。

14.2.1　ＢＲ／ＲＮ用（RX65X）

デバイスのsimpleRFTestモードに移行するためのコマンドです。

```
RFTEST_SETR 00 01          ->テストモード
            |  |-  00: 通常動作、01:テストモード
            |-HANDLE
```

※１

```
RFTEST_SETC 00 00 01
            |  |  |- future
            |  |-  00: 成功、0以外失敗
            |-HANDLE
```

アンテナ切替コマンド。

```
ANT_SETR 00 00
         |  |-  00:外部アンテナ　01:内蔵アンテナ
         |-HANDLE
```

※１

```
ANT_SETC 00 00 00
         |  |  |-  00:外部アンテナ　01:内蔵アンテナ
         |  |-00: 成功、0以外失敗
         |-HANDLE
```

リセットコマンド、モジュールの再起動を行う。

```
RESET 00
      |-HANDLE
```

マックアドレス取得

```
MAC_GETR 00
         |-HANDLE

MAC_GETC 00 00 00C14FFFFE007000
         |  |  |-MACアドレス
         |  |-00: 成功、0以外失敗
         |-HANDLE
```

シリアル番号取得

```
SERIAL_GETR 00
            |-HANDLE

SERIAL_GETC 00 00 0000000000000007
            |  |  |-シリアル番号　0000007
            |  |-00: 成功、0以外失敗
            |-HANDLE
```

ソフトウェアバージョン
SWVER_GETR 00
　　|-HANDLE

SWVER_GETC 00 0110
　　|　|- Version　01.10
　　|-HANDLE

※１　このコマンド設定後、有効にするためにはRESETコマンドを実行する必要があります。

simpleRFTestモードから通常モードへに移行コマンド
STESTR 00 00
　　|　|-- 00:通常モード　01：RFTestモード
　　|--HANDLE
注２　上記コマンド実行後RESETコマンドを実行する必要があります。(RESET)

RN用自動接続用コマンド
PAN ID設定コマンド
PARAM_SETR 00 00 ACDC
　　|　|　|---Pan ID "0xACDC"の場合
　　|　|--種別 00:PAN ID 01:NetworkName 02:PhyMode 03:Tx Power
　　|--HANDLE

PARAM_SETC 00 00 ACDC 00
　　|　|　|　　|-00: 成功、0以外失敗
　　|　|　|--PanID
　　|　|--種別
　　|-HANDLE

NetWork名の設定コマンド
PARAM_SETR 00 01 576953554E2050414E
　　|　|　|---NetworkName "WiSUN PAN"の場合
　　|　|--種別 00:PAN ID 01:NetworkName 02:PhyMode 03:Tx Power
　　|--HANDLE

PARAM_GETC 00 01 00 576953554E2050414E
　　|　|　|　|---NetworkName "WiSUN PAN"の場合
　　|　|　|---00: 成功、0以外失敗
　　|　|--種別 00:PAN ID 01:NetworkName 02:PhyMode 03:Tx Power
　　|--HANDLE

送信モード設定コマンド
PARAM_SETR 00 02 02
　　|　|　|---PhyMode ※１
　　|　|--種別 00:PAN ID 01:NetworkName 02:PhyMode 03:Tx Power
　　|--HANDLE

PARAM_SETC 00 02 02 00
　　|　|　|　|--00: 成功、0以外失敗
　　|　|　|---PhyMode ※１

```
          |  |--種別 00:PAN ID 01:NetworkName 02:PhyMode 03:Tx Power
          |--HANDLE
```

※1
02:FSK 50Kbps
04:FSK 100Kbps
05:FSK 150Kbps
33:OFDM Option2 MCS3 400Kbps
34:OFDM Option2 MCS4 600Kbps
35:OFDM Option2 MCS5 800Kbps
36:OFDM Option2 MCS6 1200Kbps
44:OFDM Option3 MCS4 300Kbps
45:OFDM Option3 MCS5 400Kbps
46:OFDM Option3 MCS6 600Kbps
54:OFDM Option4 MCS4 150Kbps
55:OFDM Option4 MCS5 200Kbps
56:OFDM Option4 MCS6 300Kbps

送信出力値設定コマンド

```
PARAM_SETR 00 03 EA
           |  |  |---Tx Power ※2
           |  |--種別 00:PAN ID 01:NetworkName 02:PhyMode 03:Tx Power
           |--HANDLE

PARAM_SETC 00 03 EA 00
           |  |  |  |--00: 成功、0以外失敗
           |  |  |---Tx Power ※2
           |  |--種別 00:PAN ID 01:NetworkName 02:PhyMode 03:Tx Power
           |--HANDLE
```

※2
E6:-26(-13.0dBm)
E7:-25(-12.5dBm)
E8:-24(-12.0dBm)
E9:-23(-11.5dBm)
EA:-22(-11.0dBm)
EB:-21(-10.5dBm)
EC:-20(-10.0dBm) FSK選択時の最大値
ED:-19(-9.5dBm)
EE:-18(-9.0dBm)
EF:-17(-8.5dBm)
F0:-16(-8.0dBm) OFDM選択時の最大値

14.2.2 LFN用 (RL78/G1H)

デバイスのsimpleRFTestモードに移行するためのコマンドです。

```
DDL_TEST_MODE 00 00
              |  |- -<1バイト>オプション値
              |- -<1バイト>Handle ID

DDL_TEST_MODEC 00 00
               |  |- -<1バイト>00はコマンド実行成功、それ以外は失敗
               |- -<1バイト>Handle ID
```

マックアドレス取得
MAC_GETR 00 00
　　|　|- -<1バイト>オプション値
　　|- -<1バイト>Handle ID

MAC_GETC 00　00 XXXXXXFFFEXXXXXX
　　|　|　|- -<8バイト>MACアドレス
　　|　|- -<１バイト>00はコマンド実行成功、それ以外は失敗
　　|- -<1バイト>Handle ID

シリアル番号取得
SERIAL_GETR 00 00
　　|　|- -<1バイト>オプション値
　　|- -<1バイト>Handle ID

SERIAL_GETC 00　00 XXXXXXXX
　　|　|　|- -<8バイト>シリアル番号
　　|　|　- -<1バイト>00で取得成功、それ以外はコマンド実行失敗　　|
　　| - -<1バイト>Handle ID

データ送信先アドレス設定
データを送信する親機のMACアドレスを登録します。
MACアドレスの長さを06にするとFFFEを補完してくれます。
08にすると8バイト分入力します。
DSTMAC_SETR 00 06 00C14F000000
　　|　|　|- -<6バイトまたは8バイト>MACアドレス(06ならFFFE不要、08なら手動で入力)
　　|　|- -<1バイト>MACアドレスの長さ（06または08のみ）
　　|- -<1バイト>Handle ID

DSTMAC_SETR 00 08 00C14FFFFE000000
　　|　|　|- -<6バイトまたは8バイト>MACアドレス(06ならFFFE不要、08なら手動で入力)
　　|　|- -<1バイト>MACアドレスの長さ（06または08のみ）
　　|- -<1バイト>Handle I

DSTMAC_GETC 00　00 XXXXXXFFFEXXXXXX
　　|　|　|- -<8バイト>MACアドレス
　　|　|- -<8バイト>00はコマンド実行成功、それ以外は失敗
　　|- -<1バイト>Handle ID

データ送信先アドレス取得
DSTMAC_GETR 00 00
　　|　|- -<1バイト>オプション値
　　|- -<1バイト>Handle ID

DSTMAC_SETC 00 00 00C14F000000
　　|　|　|- -<8バイト>MACアドレス
　　|　|- -<1バイト>00でコマンド実行成功、それ以外は失敗
　　|- -<1バイト>Handle ID

simpleRFTestモードから通常モードへに移行コマンド
DDL_NORMAL 00
　　|- -<1バイト>handle ID

DDL_NORMALC 00 00
　　|　|- -<1バイト>00はコマンド実行成功、それ以外は失敗
　　|- -<1バイト>Handle ID

14.3 モデムシリアルAPI

シリアル API は、Wi-SUN FAN スタックが 1 つの MCU 上で実行され、シリアル インターフェイスを介して他の MCU 上のアプリケーション ファームウェアと通信する、2 つのマイクロコントローラを組み合わせて使用するユース ケースをサポートするように定義されています。

14.3.1 シリアルAPIアーキテクチャ

シリアル インターフェイスは、シングル チップ実装（RL78/G1H または RX65x ベース）と 2 チップ実装（RL78/G1H + ホスト コントローラ）の両方で Wi-SUN FAN スタックの API を透過的に使用できるように設計されています。これにより、アプリケーション ソース コードは、シングル チップ ソリューションと 2 チップ ソリューションの両方で変更せずに使用できます。

Renesas は必要なソース コードをすべてサンプル コードとして提供しているため、すべてのシリアル API 機能を透過的に使用できることに注意してください。つまり、シリアル API 機能はすでに組み込まれています。

シングル チップ実装の場合、アプリケーション タスクとネットワーク タスク間の通信はキューを介して実現されます。つまり、API 関数呼び出しを介して提供されるアプリケーション データはシリアル化され、ネットワーク タスクによる後続の処理のためにキュー バッファに格納されます。図 14-2 に示すこのアーキテクチャでは、NWK API 関数は、パラメータからメッセージ（C 構造体）を構築し、キューを介してアプリケーション タスクとネットワーク タスク間の通信を行います。

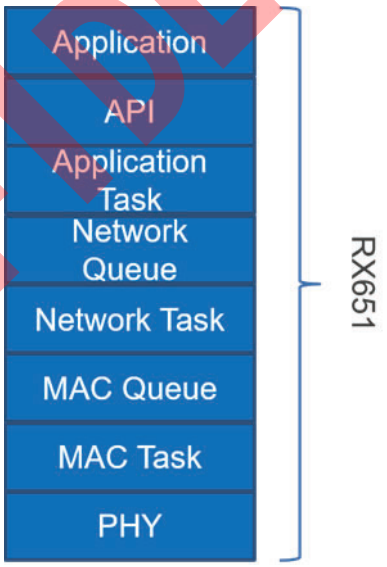


図14-2 シングルチップAPIアーキテクチャ

2 チップ アーキテクチャの場合に透過的な使用を実現するには、API のシリアル化は変更されませんが、シリアル化された API 呼び出しをネットワーク層キューに直接転送するのではなく、コマンドはシリアル HDLC エンコード フレームに埋め込まれたペイロードとして送信されます。抽出されたペイロードは、ネットワーク タスク キューに格納され、さらに処理されます。この機能をサポートするために、NWK API ソース コードには、内部メッセージ キューまたはシリアル インターフェイスをサポートするようにローカライズされた違いがあります。結果として得られるアーキテクチャを図 14-3 に示します。

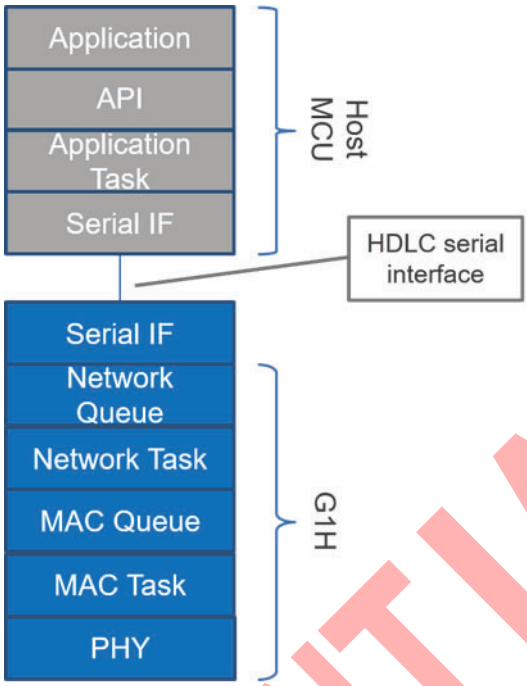


図14-3 2チップAPIアーキテクチャ

API コマンドのプラットフォームに依存しないシリアル化を可能にするために、 *r_nwk_api.h*で定義された構造体は次の機能を利用します。

- 固定サイズのスカラー データ型のみ（uint8_t、uint16_t、...）
- パディングを無効にする（プラグマ経由）
- インラインペイロード、例：UDPメッセージ用（柔軟な配列メンバー経由’’uint8_t’]’）

14.3.2　HDLCフレームフォーマット

シングルチップ実装（RL78/G1H または RX65x ベース）と Wi-SUN UI を実行しているホスト PC 間の通信、および RL78/G1H モデム デバイスとホスト コントローラ間の通信は、次のセクションで説明する堅牢なプロトコルによって実現されます。使用されているプロトコルは十分に実証されており、ルネサスの電力線通信モデムにも使用されています。

14.3.2.1　コマンドパケットフォーマット

堅牢性を確保するために、コマンド ストリームは 2 バイトのヘッダーと 4 バイトの CRC を含む HDLC タイプのパケットにカプセル化されます。すべてのデータは、開始フラグ バイト 0x7E と終了フラグ バイト 0x7E の間にカプセル化されます。図 14-4 と表 14-1 に、コマンド パケットの形式を示します。

0x7E	Reserve	Type	Payload	CRC	0x7E

IDC	IDA	IDP	CMD	DAT

図14-4 コマンドパケットフォーマット図

	Sコード	ヘッダ		ペイロード					CRCP	Sコード
		予約する	タイプ	IDC	IDA	IDP	CMDF	DAT		
価値	0x7E	0x00	-	-	-	-	-	-	-	0x7E
長さ	1バイト	1バイト	1バイト	1ビット	3ビット	4ビット	1バイト	N-2バイト	4バイト	1バイト
ノート	-	-	-	-	-	-	-	-	Big エンディアン	-

表14-1 コマンドパケットフォーマット表

14.3.2.2 HDLCエスケープシーケンス

中間データ文字のいずれかの値が 0x7E の場合、その前にエスケープ バイト 0x7D が付き、その後に元の文字とバイト 0x20 の XOR 演算から得られたバイトが続きます。文字ストリーム内に 0x7D がある場合も同様です。このような場合の例を次に示します。

送信メッセージ: 0x01 0x02 0x7E 0x03 0x04 0x7D 0x05 0x06
実際の送信シーケンス: 0x01 0x02 0x7D 0x5E 0x03 0x04 0x7D 0x5D 0x05 0x06
エスケープシーケンス
エスケープシーケンス

14.3.2.3 CRC計算

フレームの末尾にある 32 ビットの CRC は、「ヘッダー」フィールドと「ペイロード」フィールドの両方をカバーします。CRC は、送信される元のデータ、つまり前述のエスケープ シーケンスのバイト スタッフィングが実行される前のデータに対して計算されます。入力多項式 M(x) は、チェックされるデータのビットを係数とする多項式として形成されます（チェックする最初のビットは最高次の係数で、チェックする最後のビットは 0 次の係数です）。CRC の生成多項式は次のとおりです。

G(x)=x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1。剰余R(x)は、M(x)·x32をG(x)で割った剰余として計算されます。剰余の係数が結果として得られるCRCになります。表14-2にCRCの設定を示します。

アイテム	設定
幅	32ビット (CRC-32)
多項式	0x4C11DB7
方向をシフト	左
初期値	0
出力XOR	0x00000000
最終決定	ビット反転ではありません。

表14-2 CRC設定

14.3.2.4　コマンドパケットフォーマットの詳細

14.3.2.4.1 Sコード

S-CODE は、パケットの開始点と終了点を識別するためのパケット同期区切り文字です。S-CODE は常に 0x7E です。

14.3.2.4.2 予備

このフィールドは将来の使用のために予約されています。

14.3.2.4.3 タイプ

このフィールドは、表 14-3 に示されているプロトコル タイプを示します。
Wi-SUN FAN の場合、TYPE = 0x06 のみが使用されます。

タイプ	説明
0x00	システムブロック
0x01	予約する
0x06	Wi-SUNファン
0x07-0xFF	予約する

表14-3 TYPEの定義

14.3.2.4.4 CRCC

セクション14.3.2.3を参照してください。

14.3.2.4.5 情報通信技術

このフィールドは、表14-4に示すようにチャネルインデックスを示します。
Wi-SUN FAN の場合、IDC = 0 のみが使用されます。

IDC	説明
0x0	チャンネル0
0x1	チャンネル1

表14-4 IDCの定義

14.3.2.4.6 IDA

このフィールドは、表 14-5 に示すように SAP のアクセス タイプを示します。

IDA	説明
0x0	リクエスト（モデム設定にのみ使用）
0x1	確認　　（モデム設定にのみ使用）
0x2	表示　　（モデム設定にのみ使用）
0x3	予約する
0x4	Printfおよびデモンストレータコマンドに使用
0x5-0x7 の	予約する

表14-5 IDAの定義

14.3.2.4.7 IDP

このフィールドは、表 14-6 に示すように、アクセス ターゲット レイヤーを示します。標準構成（シングル チップ構成）の Wi-SUN FAN では、IDP = 0x00 のみが使用されます。2 チップ構成では、IDP = 0x06 がクライアントとサーバー モジュール間のネットワーク レイヤー通信をさらにサポートします。

IDP	説明
0x0	コントローラ
0x1	予約する
0x2	予約する
0x3	UMAC レイヤー
0x4	ADP レイヤー
0x5	EAP レイヤー
0x6	ネットワーク層（モデム構成に使用）
0x7-0xF	予約する

表14-6 IDPの定義

14.3.2.4.8 コマンド

このフィールドはコマンド ID を示します。各コマンドの ID は、ヘッダー ファイル*r_nwk_api.h*で定義されている*列挙体r_nwk_msg_type_tと一致します。*

14.3.2.4.9 DAT

、*r_nwk_api.h*で定義されているシリアル化されたコマンド構造で構成される実際のコマンド ペイロードが含まれます。

- r_***_req_t　リクエスト構造
- r_***_cfm_t　確認構造
- r_***_ind_t　指示構造

※参考資料

HDLフレーム構成

Wi-SUN FAN1.1で扱うコマンドのHDLCフレーム構成は以下のようになっています。

		予約タイプ	ヘッダ		ペイロード(Nバイト)					CRCP	予約タイプ
		S-CODE	Reserve	Type	IDC	IDA	IDP	CMD	DAT	CRC	S-CODE
	値	0x7E固定	0x00固定	0x06固定	0固定	100(4)固定	0000(0)固定	任意の1バイト値	実際のコマンド部分	ヘッダからペイロードまでをCRC32で変換した値	0x7E
	長さ	1バイト	1バイト	1バイト	1ビット	3ビット	4ビット	1バイト	N-2バイト	4バイト	1バイト
	CRCの値はヘッダ部分とペイロード部分をCRC32処理を行ったハッシュ値になります。先頭の予約タイプは含めません。										
	詳細はWi-SUN FAN Protocol Stack User Manualの　14.2.2 HDLC Frame Formatを参照下さい。										

例として、「STATUS_GETR 00 00」コマンドの場合は以下のようなフレーム構成になります(シングルチップの場合)。

		ヘッダ		ペイロード					CRCP	
	予約タイプ	Reserve	type	IDC	IDA	IDP	CMD	DAT	CRC	予約タイプ
値	0x7E	0x00	0x06	0	100	0000	0x00	STATUS_GETR 00 00¥n	*0x90 0xBB 0xC8 0x17	0x7E
0x表記	0x7E	0x00	0x06	0x40			0x00	0x53 0x54 0x41 0x54 0x55 0x53 0x5F 0x47 0x45 0x54 0x52 0x20 0x30 0x30 0x20 0x30 0x30 0x0A	*0x90 0xBB 0xC8 0x17	0x7E
*DAT部分のハンドル値やCMDの値を変更すると変化します。STATUS_GETR 00 00¥nと入力したときのみのCRCです。										
DAT部のコマンドはWi-SUN FAN Protocol Stack User ManualのChapter 14 Serial API Command Descriptionを参照下さい。										

第15章シンプルなRFテストツール

このドキュメントで説明されている Wi-SUN FAN サンプル アプリケーションの代わりに、各評価ボードでシンプルな RF テスト アプリケーションを実行できます。このアプリケーションを使用すると、専用のシリアル コマンドを介してサブ GHz RF ドライバーに直接アクセスすることで、シンプルな RF テストを簡単にセットアップして実行できます。Wi-SUN FAN サンプル アプリケーションのデモ コマンドとは異なり、これらのテキストベースのシリアル コマンドは HDLC エンコードされていないため、標準のターミナル プログラム（Tera Term、 PuTTYなど）を使用して、シンプルな RF テスト アプリケーションと通信および対話できます。

「RF テスト」アプリケーションの使用方法と操作の詳細については、リリース パッケージの一部である対応する仕様（ドキュメント

「R30UW0075EJ0103_Simple_RFTest_Cmd_Spec.pdf」）を参照してください。

シンプルテストモードと通常FANスタックを切り替えられるようなコマンドをBR,RN、LFNにそれぞれ追加しております。

テストモードへの入り方：（BR、RN）（RX65X）

Wi-SUN UIアプリケーションにて

Start Developer Modeを選択し、 Send Serial API Commandから次のコマンドを入力する。（図15-1参照）

RFTEST_SETR 00 01 → テストモード設定

RESET 00 → リセット動作

この後、UIツールを終了し、Teratermで接続する。（この段階でテストモードに入っている）

必要なテスト終了後、通所動作に戻る為には、下記コマンドを入力する。

STESTR 00 00 → 通常モード設定

RESET → リセット動作

テストモードへの入り方：（LFN(RL78)）

Wi-SUN UIアプリケーションにて

Start Developer Modeを選択し、 Send Serial API Commandから次のコマンドを入力する。（図15-1参照）

DDL_TEST_MODE 00 00 → テストモード設定

※コマンド実行後にテストモードへ。リセットなどは不要

この後、UIツールを終了し、Teratermで接続する。

必要なテスト終了後、通所動作に戻る為には、下記コマンドを入力する。

DDL_NORMAL 00 → 通常モードへ

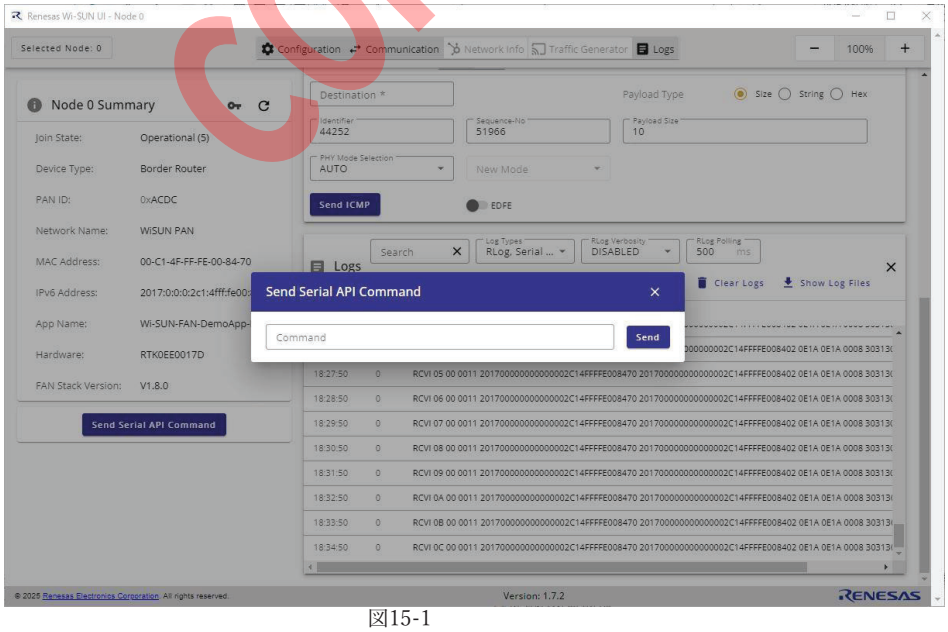


図15-1

改訂履歴

Rev.	日付	説明	
		ページ	まとめ
1.00	2025年3月21日	-全章	初版発行