モノプラットフォーム β版
SIPF Firmware for nRF9160
UART インターフェイス仕様

概要

nRF9160 用の SIPF Firmware をデバイスに書き込んだ時に用意される、UART インターフェイスの仕様です。

ユーザーMCU はこの UART インターフェイスを使用して、SIPF Firmware にアクセスすることで、さくらのモノプラットフォームが用意する各種機能の利用、nRF9160 の制御を、容易に行うことができます。

このインターフェイスで利用できる SIPF が提供する現時点での各種機能には、代表的な対応機能として、オブジェクト送受機能、ファイル送受機能等があります。

SIPF Firmware 自体は下記 URL にて提供されています。

https://github.com/sakura-internet/sipf-std-client_nrf9160

SIPF Firmware のコードは公開されておりユーザー側でコードの追加/変更などが可能ですが、その場合はこの仕様書との一致性は一切保証されません。

本資料について

SIPF Firmware が提供する UART 部分のみ解説しています。

この UART インターフェイスを利用するには SIPF Firmware 及び、さくらのモノプラットフォームサービス全体、各さくらのモノプラットフォーム機能の理解が必要です。各仕様書を参照の上ご利用ください。

本資料ではこのインターフェイスの利用手段の説明のみになります。このインターフェイス経由での各機能を利用した応用例などはアプリケーションノートなどを参照してください。

注意事項

基本的に現状の実装されたところのみ整備しており、最終的な実装機能全てを網羅しているものではありません。また、一部現在未実装の将来提供予定の内容が説明の都合などで記載されていることがあります。

今後の実装の進捗状況により変更が発生する場合があります。

目次

アーキテクチャ

本 UART インターフェイスでは、コマンドベースのプロトコルを採用しており、コマンドとして、ヒューマンリーダブルな便利 ASCII モードを用意しています。

この便利 ASCII コマンドでは、人間の可読性操作性を優先した設計になっており、試用時などに手打ちなどにより容易に試してみることができるようになっています。ただし、このコマンド体系では代表的な主要機能に実装が絞られています。将来的にモノプラットフォームや SIPF Firmware の細かな機能を利用できたり、MCU などからの制御をより容易とするコマンド体系も用意することを計画しています。

この UART インターフェースを利用いただくことで様々なユーザーのアプリケーションからこのモノプラットフォームや SIPF Firmware を快適に利用いただくことができます。

対応モノプラットフォーム機能

モノプラットフォームが提供するデバイス側への機能の内、下記に対応しています。

- 認証機能

  セキュアモバイルコネクトの利用によるモノプラットフォームへの接続認証に対応
- オブジェクト送受信機能

  基本的な送受に対応
- ファイル送受信機能

  ファイル送受、およびファイル送受機能を利用した SIPF Firmware 自体の更新機能に対応

UART インターフェイス利用時の注意事項

コマンド実行前にユーザーMCU 側の UART の送受バッファをクリアすることをおすすします。バッファに残留していたゴミデータによりコマンドの失敗や意図しない動作を防ぐことにつながります。

ドキュメント中用語、表現等定義

進数表記

２進数 : 0b を頭に表記　例）0b1010

16 進数 : 0x を頭に表記　例）0xF0F0

10 進数 : そのまま表記　例）125

ただし、上記は数値としての記載の場合に限り、文字列についての記載の場合は何進数の場合でも実際に使用する文字のみ記載します。

例：16 進数 2 桁の文字列のサンプルは「FF」と記載され「0x」は付加されません

スペース文字

ドキュメント中、明示的にスペースを挿入すべきことを指示する際には「␣」を用いる。

改行文字

ドキュメント中、明示的に改行を挿入すべきことを指示する際には「↵ 」を用いる。

UART ライン上で使用する改行文字としては基本的に CR+LF(0x0D0A)を用いる。

物理層

電気的仕様については別資料として電気仕様書として提供しています。そちらを参照願います。

項目	方式
ボーレート	標準：115200bps
データビット	8bit
スタートビット	1bit
ストップビット	1bit
パリティ	なし
改行コード	送受共に CR+LF
フロー制御	なし
文字間必要遅延	0ms
行間必要遅延	前コマンド行レスポンス完了後 0ms

コマンド/レスポンスの形式

ユーザーMCU と SIPF Firmware との間では、UART ライン上でコマンド文字列/レスポンス文字列として通信が行われます。ユーザーMCU から SIPF Firmware 方向の通信がコマンド文字列、逆方向がレスポンス文字列となります。ここではそれぞれのコマンド/レスポンスのフォーマットを定義します。

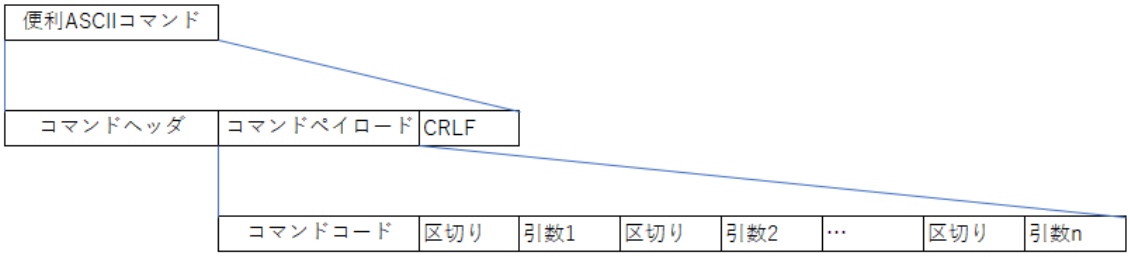
現在便利 ASCII コマンドのみ定義されています。

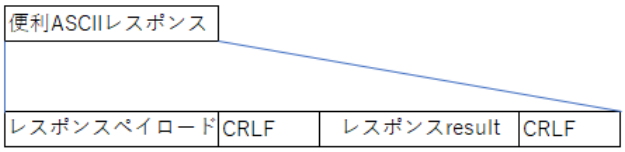
コマンド形式	対応レスポンス形式	コマンド概要
便利 ASCII コマンド	便利 ASCII レスポンス	ヒューマンリーダブル優先 規定処理の簡易実行に特化 利用できる機能は代表的/基本的なものに限定 SIPF Firmware 内を意識せず利用

また基本的に通信の開始はユーザーMCU 側から行います。SIPF Firmware 側からの通信開始がないことで、常時ユーザーMCU 側で待ち受ける必要がなく、これにより、ユーザーデバイスの処理負荷状況や、電力事情などに応じて適切なタイミングでの送受信処理を行うことができます。

便利 ASCII コマンド

ユーザーMCU から SIPF Firmware に対して人の読める形の ASCII 文字列で送信する際の通信フォーマットで、典型的な処理を容易に利用できるようにしたものです。





便利 ASCII コマンド文字列構成定義

フィールド	文字数	文字列	概要
便利 ASCII コマンドヘッダ	2	$$	便利 ASCII コマンド開始
便利 ASCII コマンドペイロード	n	2044 文字以下の ASCII 文字からなるコマンドペイロード文字列	
CRFL	2	CRLF	便利 ASCII コマンド終了

※便利 ASCII コマンド全体の最大長は 2048 文字以下に制限されています。あくまでもこの UART インターフェイスの実装上の制限であり、モノプラットフォームとの接続プロトコル上の制限ではないことに注意してください。

- 便利 ASCII コマンドヘッダ

  「$$」の文字列で 2 文字を便利 ASCII コマンドの開始文字とします。この文字で始まらないものは便利 ASCII モードのコマンドとはみなしません。(行頭が「$$」の必要があるわけではなく「$$」以降をコマンドとしてパースしていく形になります。「$$」の直前に無関係な文字があったとしても読み捨てられ、「$$」以降をコマンドとして認識します。)

- 便利 ASCII コマンドペイロード

  2044 文字以下の ASCII 文字です。

  SIPF Firmware に対する指示を表す文字列が入ります。

  ASCII 文字のうち、「0～9」、「A～Z」、「a～z」、「␣.-_」のいずれかのみを用います。

  詳細は ペイロードの構成-便利 ASCII コマンドペイロード、および各コマンドの項を参照してください。

- CRLF

  「CRLF」の文字列で 2 文字を 便利 ASCII コマンド終了文字とします。

  コマンドのパースの開始基準になります。この CRLF が正しく SIPF Firmware に伝わらなかった場合、SIPF Firmware 側の UART バッファにゴミが残るなどの可能性があり、次のコマンドがエラーになることがあります。

便利 ASCII レスポンス

便利 ASCII コマンドに対する、ユーザーMCU への SIPF Firmware からの応答の通信フォーマットです。

便利 ASCII レスポンス文字列構成定義

フィールド	文字数	文字列	概要
便利 ASCII レスポンスペイロード	n	0 文字以上の ASCII 文字からなるレスポンス文字列	
CRLF	0 or 2	CRLF	便利 ASCII ペイロードがあった場合に付加される
便利 ASCII レスポンス Result	2	OK / NG どちらか	便利 ASCII コマンドの実行結果や、コマンドとして成り立っているかを応答します。
CRLF	2	CRLF	便利 ASCII レスポンス終了

※便利 ASCII レスポンスの最大長は定義されていません。各便利 ASCII コマンド毎に便利 ASCII レスポンスペイロードの最大長が定義されますので、利用するコマンドのうち最大長の ASCII レスポンスペイロードを含む ASCII レスポンスを受信できるよう設計してください。

- 便利 ASCII レスポンスペイロード

  SIPF Firmware からのコマンドに対する応答の文字列が入ります。コマンドにより無い場合もあります。

  詳細はペイロードの構成の項を参照してください。

  0byte 以上の ASCII 文字。

  ASCII 文字のうち、「0～9」、「A～Z」、「a～z」、「␣.-_/%」、「CRLF」のいずれかのみを用います。

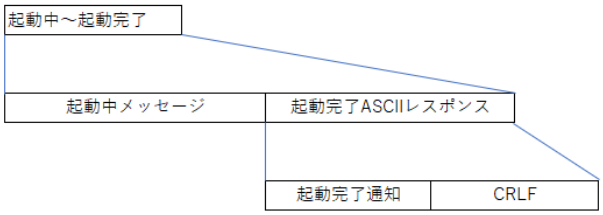
  詳細は ペイロードの構成-便利 ASCII レスポンスペイロード、および各コマンドの項を参照してください。

  便利 ASCII レスポンスペイロードの最大長は定義されていません。各便利 ASCII コマンド毎に便利 ASCII レスポンスペイロードの最大長が定義されますので、利用するコマンドのうち最大長のものを受信できるよう設計してください。

- CRLF
  「CRLF」文字列はペイロードがあった場合(0 文字ではなかった場合)に末尾に付加されます。
  ペイロード末尾の区切りを示します。(ペイロード内にも CRLF が含まれることがあり、その場合は便利 ASCII レスポンス Result の直前の CRLF のみがこの区切りに対応することに注意してください)
- 便利 ASCII レスポンス Result
  コマンドとして正常かどうかや、コマンドの処理の結果を表します。
  「OK」/「NG」のどちらかが返ってきます。
  NG の詳細な原因はコマンドによっては便利 ASCII レスポンスペイロード内で返されることもあります。
  また、この便利 ASCII レスポンス Result ＋ CRLF が返ってくるまでの間は、SIPF Firmware は基本的に busy であり、次のコマンドを送り付けても正常に処理される保証はありません。
- CRLF
  「CRLF」文字列はレスポンス末尾の区切りを示します。

起動中～起動完了メッセージ

SIPF Firmware が起動中にユーザーMCU へ送出する SIPF Firmware からの通信フォーマットです。



メッセージ文字列構成定義

フィールド	文字数	文字列	概要
起動中メッセージ	n	任意の ASCII 文字列(文字種の限定はありません) 起動時のメッセージで内容/長さの規定はありません。 起動完了 ASCII レスポンスの手前までになります。	

起動中のパラメータ表示などが行われます。Ver などにより表示内容は異なります。

起動完了 ASCII レスポンス

SIPF Firmware が起動完了時にユーザーMCU へ送出する SIPF Firmware からの通信フォーマットです。

レスポンス文字列構成定義

フィールド	文字数	文字列	概要
起動完了通知	13	+++␣Ready␣+++	起動完了を示します
CRLF	2	CRLF	起動完了レスポンス終了

SIPF Firmware の起動完了を通知します。このメッセージ送出後、SIPF Firmware はコマンドを受け付けることができるようになります。電源投入やリセットなどの後、この起動完了 ASCII レスポンスが返ってくるまでの間は、SIPF Firmware は基本的に busy であり、その完了前に送出したコマンドなどの動作の保証はありません。また、このメッセージは、SIPF Firmware のアップデート処理時の完了通知メッセージを兼ねています。このメッセージの受信を持って、Firmware 更新後の再起動完了と UPDATE コマンドの終了を判断できます。

ペイロードの構成

各コマンド/レスポンスで使用されるペイロードフィールド内の構成を示します。

便利 ASCII コマンドペイロード

ユーザーMCU から SIPF Firmware に対しての指示を表します。

0 文字以上の ASCII 文字。「0～9」、「A～Z」、「a～z」、「␣.-_」のいずれかのみを用いて表現されます。

ペイロード内の要素は␣で区切られます。

コマンドペイロード文字列構成定義

フィールド		文字数	
便利 ASCII コマンドコード		1～16	SIPF Firmware への指示種別を表す
引数 1	区切り	1	引数が必要なコマンドの場合区切りとして␣を挿入
	便利 ASCII コマンド引数	1～n	SIPF Firmware への指示への付加情報を表す
…		…	…
引数 m	区切り	1	引数が必要なコマンドの場合区切りとして␣を挿入
	便利 ASCII コマンド引数	1～n	SIPF Firmware への指示への付加情報を表す

※コマンドペイロード全体で 1020byte 以下に制限する必要があります

例：

便利 ASCII コマンドコード␣便利 ASCII 引数 1␣便利 ASCII 引数 2␣…␣便利 ASCII 引数 m

- 便利 ASCII コマンドコード

  「0～9」、「A～Z」、「a～z」のいずれかのみを用いて表現されます。最大 16 文字。

  SIPF Firmware への指示種別を表します。

  定義コマンドや詳細は、便利 ASCII コマンドコード項を参照してください。

- 便利 ASCII コマンド引数

  オプション。コマンドにより必要個数は変動します。

  SIPF Firmware への指示への付加情報を表します。

  「0～9」、「A～Z」、「a～z」、「␣.-_」のいずれかのみを用いて表現されます。

  詳細は、各便利 ASCII コマンドコード項を参照してください。

便利 ASCII レスポンスペイロード

ユーザーMCU への SIPF Firmware からの応答を表します。

基本的には 0 文字以上の ASCII 文字。「0～9」、「A～Z」、「a～z」、「␣.-_/%」、「CRLF」、のいずれかのみを用いて表現されます。

便利 ASCII レスポンスペイロードには、便利 ASCII コマンド便利 ASCII コマンドの実行結果の応答や、エラー内容などが格納されます。コマンドの種類や実行結果により無いこともあります。 詳細は各コマンドコード毎の定義を参照してください。

このレスポンスペイロードの最大長は各コマンド等により異なるため定義されていません。各ユーザーごとに使用するコマンドの組み合わせから個別の最大長を算出し、その最大長のレスポンスペイロードを受け入れられるようユーザーMCU のバッファなどを調整する必要があります。詳細は各コマンド毎の定義を参照してください。

レスポンスペイロード文字列構成定義

フィールド	文字数	
レスポンスペイロード	0～n	便利 ASCII コマンドに対する応答データ

- 便利 ASCII レスポンス Result が OK の時

  便利 ASCII コマンドに対する応答データはコマンド実行結果など各コマンドに対応したユーザーMCU への SIPF Firmware からの応答を表します。コマンドにより空のこともあります。

  基本的には「0～9」、「A～Z」、「a～z」、「␣.-_/%」、「CRLF」のいずれかのみを用いて表現されます。

  どのような応答が返るかは各コマンドコードの項を参照してください。

- 便利 ASCII レスポンス Result が NG の時

  便利 ASCII コマンドに対する応答データはコマンド実行に失敗した時のエラー内容や正常なコマンドでないときのユーザーMCU への SIPF Firmware からの応答を表します。便利 ASCII コマンドの構文が異常な時などや、コマンドの処理結果に問題がある場合に、便利 ASCII レスポンスデータの代わりとして、その問題についての示すメッセージが返ります。コマンドにより空のこともあります。

  「0～9」、「A～Z」、「a～z」、「␣.-_/%」、「CRLF」のいずれかのみを用いて表現されます。どのような応答が返るかは各コマンドコードの項を参照してください。

メッセージ	内容
ILLEGAL PARAMETER	便利 ASCII コマンドペイロードが不正です
コマンド特有メッセージ	各コマンドの項を参照

  コマンドの処理結果に問題がある場合に、どのような応答が返るかは各コマンドの項を参照してください。

便利 ASCII コマンドコード

便利 ASCII モード時の使用できるコマンドとそのレスポンスについて定義しています。

便利 ASCII コマンドは基本的にヒューマンリーダブル性、ターミナルへの手打ち性を優先しています。ごく一般的な動作を簡単に試せるよう設計されています。

コマンド一覧

カテゴリ	コマンド名	コマンドコード文字列	概要
オブジェクト送受	オブジェクト送信	TX	引数の複数のオブジェクトを 1 オブジェクト転送単位としてモノプラットフォームへ送信。
オブジェクト送受	オブジェクト受信	RX	モノプラットフォームから 1 オブジェクト転送単位のオブジェクトセットを受信し、表示する。
ファイル送受	ファイル送信	FPUT	ユーザー指定の FILE_ID 文字列をターゲットとして、XMODEM プロトコルでファイルを送信する。
ファイル送受	ファイル受信	FGET	ユーザー指定の FILE_ID 文字列でファイルを指定して、XMODEM プロトコルでファイルを受信する。
管理	管理コマンドのロックを解除	UNLOCK	管理コマンド系が誤って実行されないようにかかっているロックを一時的に解除する。
管理	SIPF Firmware の更新	UPDATE	ファイル送受機能を使用して、指定した FILE_ID 文字列の FW ファイルをダウンロードして SIPF Firmware 本体を更新します。
TBD	TBD	TBD	今後追加予定

TX コマンド

SIPF のデバイスアダプターに対してオブジェクトプロトコルを用いて、引数として渡すオブジェクトをオブジェクト転送単位としてまとめモノプラットフォームに対して自動的に送信するコマンドです。このコマンドでは複数のオブジェクトを同時に送信することもできます。

「便利 ASCII コマンド Result」はデバイスアダプターへの送信が正常に完了したかどうかを示します。送信したオブジェクトがサービスアダプターを介してユーザーサーバーへの到着などを示すものではないことにご注意ください。

OK の場合のみ「便利 ASCII コマンドレスポンスペイロード」として「OTID(オブジェクト転送単位 ID)」が応答されます。

NG の場合は「便利 ASCII コマンドレスポンスペイロード」として失敗となった原因が応答されます。

オブジェクトの送信から「OTID(オブジェクト転送単位 ID)」の取得までの複数の送受を連続で自動的に行うため、応答が完了するまでに、環境や状況次第で数十～数千 ms 程度の時間がかかります。

今現在本コマンドでは、オブジェクトに対してユーザが任意の時刻情報を付与できるパラメータ Timestamp_src(RX 時の USER_SEND_DATETIME の逆向きに相当)機能は対応しておりません。

便利 ASCII コマンドペイロード構成

基本構成

要素			コマンド文字列	文字数	概要
便利 ASCII コマンドコード			TX	2	TX コマンドを表します。
区切り			␣	1	
引数 1	オブジェクト 1	TAG_ID	16 進数 2 桁	2	送信オブジェクトの TAG_ID を指定します。 00～FF の 16 進数 2 桁で表記します。ユーザーのシステム設計に合わせて任意に設定できます。
区切り			␣	1	
引数 2		TYPE	16 進数 2 桁	2	送信オブジェクトの型指定 ID を指定します。00～FF の 16 進数 2 桁で表記します。詳細は型毎詳細を参照してください。
区切り			␣	1	
引数 3		VALUE	送信するオブジェクトの値を指定します。最大 510 文字。 引数 2 により指定した型指定 ID により 16 進数の桁数のフォーマットが変わります。 詳細は型毎詳細を参照してください。		
区切り			␣	1	
引数 4	オブジェクト 2	TAG_ID	16 進数 2 桁	2	オブジェクト 1 と同様
区切り			␣	1	
引数 5		TYPE	16 進数 2 桁	2	オブジェクト 1 と同様
区切り			␣	1	
引数 6		VALUE	オブジェクト 1 と同様		
⋮	⋮	⋮	⋮	⋮	⋮
区切り	␣		␣	1	
引数 m-2	オブジェクト n	TAG_ID	16 進数 2 桁	2	オブジェクト 1 と同様
区切り			␣	1	
引数 m-1		TYPE	16 進数 2 桁	2	オブジェクト 1 と同様
区切り			␣	1	
引数 m		VALUE	オブジェクト 1 と同様		

※コマンドペイロード全体で 2044 文字以下に制限する必要があります

- 便利 ASCII コマンドコード 「TX」
- 必要引数　3 x n 個。
- 必要オブジェクト n 個
  - 引数 3n+1　TAG_ID

    送信するオブジェクトの TAG_ID を指定します。

    １６進表記２桁(00～FF)の文字列で表現します。

    小さい値でも必ず頭に 0 を付加し 2 桁とする必要があります。この ID はオブジェクトの VALUE が何を表す値なのかを、ユーザーのデバイスとサーバーの間で共有するための情報で、ユーザーのシステム設計に合わせて任意に設定できます。
  - 引数 3n+2　TYPE

    送信するオブジェクトの VALUE の型を ID で指定します。

    １６進表記２桁(00～FF)の文字列で表現します。

    指定できる型は型指定一覧の表を参照してください。
  - 引数 3n+3　VALUE

    送信するオブジェクトの値を指定します。

    １６進表記の文字列で表現します。表記法は指定した TYPE により異なります。詳細は型指定一覧の表を参照してください。

TX コマンド例：符号なし 32bit 整数値 38177486 を送る場合

要素	コマンド文字列	文字数	概要
便利 ASCII コマンドコード	TX	2	TX コマンドを表します。
区切り	␣	1	
引数 1	01	2	送信オブジェクトの TAG_ID を指定します。 00～FF の 16 進数 2 桁の文字列で表記します。 ユーザーのシステム設計に合わせて任意に設定できます。 ここでは ID を 0x01 としています。
区切り	␣	1	
引数 2	04	2	送信オブジェクトの型指定 ID を指定します。 UINT32 の ID、0x04 を指定します。
区切り	␣	1	
引数 3	02468ACE	8	送信する UINT32 の値を指定します。 38177486 を 16 進数文字列で表した 02468ACE を指定します。

便利 ASCII コマンド TX 文字列サンプル

$$TX␣01␣04␣02468ACE↵

TX コマンド例：

符号なし 32bit 整数値 38177486 と、utf8 文字列「012abc 漢字」 の同時送信の場合

要素	コマンド文字列	文字数	概要
便利 ASCII コマンドコード	TX	2	TX コマンドを表します。
区切り	␣	1	
引数 1	01	2	送信オブジェクトの TAG_ID を指定します。 00～FF の 16 進数 2 桁の文字列で表記します。 ここでは ID を 0x01 としています。
区切り	␣	1	
引数 2	04	2	送信オブジェクトの型指定 ID を指定します。 UINT32 の ID、0x04 を指定します。
区切り	␣	1	
引数 3	02468ACE	8	送信する UINT32 の値を指定します。 38177486 を 16 進数文字列で表した 02468ACE を指定します。
区切り	␣	1	
引数 4	02	2	送信オブジェクトの TAG_ID を指定します。 00～FF の 16 進数 2 桁の文字列で表記します。 ここでは ID を 0x02 としています。
区切り	␣	1	
引数 5	20	2	送信オブジェクトの型指定 ID を指定します。 utf8 文字列の ID、0x20 を指定します。
区切り	␣	1	
引数 6	303132616263E6BCA2E5AD97	24	送信する utf8 文字列を表す文字列を指定します。 「012abc 漢字」の 16 進数文字列、303132616263E6BCA2E5AD97 を指定します。

便利 ASCII コマンド TX 文字列サンプル

```
$$TX␣01␣04␣02468ACE␣02␣20␣303132616263E6BCA2E5AD97↵
```

便利 ASCII レスポンスペイロード構成

デバイスアダプターへのオブジェクト送信成功時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	オブジェクト転送単位 ID(16 進表記 32 桁)

オブジェクトが正常にデバイスアダプターへ送信されると、便利 ASCII レスポンスペイロードには、便利 ASCII レスポンスデータが返され、その中で「オブジェクト転送単位 ID」が応答されます。

この ID は 16 進表記 32 桁表記の値で表現された符号なし 128bit の数値です。

便利 *ASCII* レスポンスデータ例

64A100C091E041CA85A81BC60CDF2371

TX コマンドとして認識できるが構文異常の場合

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII コマンド Result	失敗内容に応じたコマンド固有メッセージ

何らかの問題があった際には、一部の問題ではその要因を便利 ASCII コマンド Result として応答します。問題によっては無い場合もあります。

メッセージ	内容
ILLEGAL␣PARAMETER	サポートされてない構文で指定された場合

便利 *ASCII* コマンド *Result* 例

ILLEGAL␣PARAMETER

便利 ASCII レスポンス Result 構成

正常にコマンドを受信し、デバイスアダプターに送信できた場合のみ OK が返ります。

それ以外はすべて NG となります。

条件	応答文字列
成功時	「OK」(0x4f4b)
失敗時	「NG」(0x4e47)

便利 *ASCII* レスポンス *Result* 例

```
OK
```

便利 *ASCII* レスポンス例

```
64A100C091E041CA85A81BC60CDF2371 ↵
OK ↵
```

TX コマンド～レスポンスまで例 1

```
$$TX␣01␣04␣02468ACE ↵
64A100C091E041CA85A81BC60CDF2371↵
OK ↵
```

TX コマンド～レスポンスまで例 2

```
$$TX␣01␣04␣FFFFFFFFFFFFFFFF ↵
ILLEGAL␣PARAMETER ↵
NG↵
```

RX コマンド

デバイスアダプターから 1 オブジェクト転送単位をこのデバイスに取得して、その中に含まれる一つ～複数のオブジェクトを UART 経由で取得できるコマンドです。

このコマンドでは、まずこのデバイス向けのモノプラットフォーム上の受信(オブジェクト転送単位をデバイス向けに送る方向)キューを確認し、このデバイス向けのオブジェクト転送単位が存在する場合はキューの先頭から 1 オブジェクト転送単位を SIPF Firmware の受信バッファに受信します。(1 オブジェクトでないことに注意してください。サービスアダプター経由でこのデバイス向けにオブジェクトを送る際に、1 オブジェクト転送単位内に詰めたオブジェクトのセットが一度に受信されます。1 オブジェクトずつの受信が必要な場合はオブジェクト転送単位内に一つだけオブジェクトを入れて送るとこの RX コマンドで一つずつ受け取ることが可能です。)この際、このデバイス向けのモノプラットフォーム上の受信キューからこのオブジェクト転送単位は消え、次のオブジェクト転送単位が先頭になり、次に発行される RX コマンドで受信可能となります。

その後レスポンスペイロードとして、受信したオブジェクト転送単位のオブジェクト転送単位 ID と時刻情報、そしてオブジェクト転送単位内の先頭のオブジェクトから順に便利 ASCII レスポンスデータとして UART 上で応答します。

便利 ASCII レスポンス Result の OK/NG はモノプラットフォームから受信して、オブジェクト転送単位内のオブジェクトを正常に応答するまでを正常に完了したかどうかを示します。ただし例外として、このデバイス向けのモノプラットフォーム上の受信キューが空であることを正常に確認できた時は OK の便利 ASCII レスポンス Result を返します。

NG の場合はレスポンスペイロードとして失敗となった原因を表現します。

モノプラットフォームからオブジェクト転送単位の取得から UART での応答 までの複数の送受処理を連続で自動的に行うため、応答が完了するまでに、環境や状況次第で数十～数千 ms 程度の時間がかかります。

便利 ASCII コマンドペイロード構成

基本構成

要素	コマンド文字列	文字数	概要
便利 ASCII コマンドコード	RX	2	RX コマンドを表します。

- 便利 ASCII コマンドコード「RX」
- 必要引数 0

便利 ASCII コマンド文字列サンプル

$$RX↵

便利 ASCII レスポンスペイロード構成

SIPF からのオブジェクト受信成功時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	オブジェクト転送単位 ID(16 進表記 32 桁)
	USER SEND DATATIME(16 進表記 16 桁)
	SIPF RECEIVE DATATIME(16 進表記 16 桁)
	REMAIN(16 進表記 2 桁)
	OBJECT_QTY(16 進表記 2 桁)
	オブジェクト 1
	オブジェクト 2
	～
	オブジェクト n

SIPF から受信できるオブジェクト転送単位が無かった場合

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	なし

いずれの場合も便利 ASCII コマンド Result は OK となります

便利 ASCII レスポンスデータ詳細

要素			コマンド文字列	文字数	概要
要素 1		OTID	16 進数 32 桁	32	オブジェクト転送単位 ID が指定されます。 00～FF の 16 進数 32 桁で表記します。
区切り			↵	2	
要素 2		user send time	16 進数 16 桁	16	ユーザーがこのオブジェクト転送単位に対してセットした時刻が ms の unixtime で指定されます。省略可能なパラメータで、省略されている場合は 0 が入っています。 ユーザーが任意に設定できる時刻情報です。 16 進数 16 桁で表記します。
区切り			↵	2	
要素 3		SIPF Recive time	16 進数 16 桁	16	モノプラットフォームがユーザーサーバーからこのオブジェクト転送単位を受け取った時刻が ms の unixtime で指定されます。モノプラットフォーム上で自動的に付加されます。 16 進数 16 桁で表記します。
区切り			↵	2	
		Remains	16 進数 2 桁		このオブジェクト転送単位を取り出した後の、モノプラットフォームに残っている受信可能なオブジェクト転送単位数が 00～FF の 16 進数 2 桁で表記されます。
区切り			↵	2	
		Object Qty.	16 進数 2 桁		オブジェクト転送単位に含まれるオブジェクト数が 00～FF の 16 進数 2 桁で表記されます。
区切り			↵	2	
オブジェクト 1					オブジェクト内の構成詳細は下の表を参照してください
区切り			↵	2	
オブジェクト 2					オブジェクト内の構成詳細は下の表を参照してください
⋮			⋮	⋮	⋮
区切り			↵	2	
オブジェクト n					オブジェクト内の構成詳細は下の表を参照してください

各オブジェクトのフォーマットは下記のフォーマットで供給されます

要素	コマンド文字列	文字数	概要
要素 1	16 進数 2 桁	2	受信オブジェクトの TAG_ID が表記されます。 00～FF の 16 進数 2 桁で表記します。
区切り	␣	1	
要素 2	16 進数 2 桁	2	受信オブジェクトの型(Type)指定 ID が指定されます。 00～FF の 16 進数 2 桁で表記します。
区切り	␣	1	
要素 3	16 進数 2 桁	2	TYPE 毎に定められた VALUE のバイナリ時のサイズが表記されます。 00～FF の 16 進数 2 桁で表記します。 Value の 16 進表記文字列のサイズではなくバイナリ表現時のサイズであることに注意してください。このレスポンスでの VALUE の表記文字数は SIZE の 2 倍の値になります。
区切り	␣	1	
要素 4	16 進数文字列	m	受信したオブジェクトの値(Value)が指定されます。 16 進数文字列で表されます。桁数は Type によって変わり、対応した桁数で表記します。

オブジェクト転送単位 ID は下記の 16 進表記 32 桁の値で供給されます。

便利 *ASCII* レスポンスデータ例

TAG=1, uint32 38177486

TAG=2, string.utf8 012abc 漢字

の 2 つのオブジェクトが含まれるオブジェクト転送単位を受信した場合

```
6EEC0C78C7D34F60A568AC6A7F9AE0F7↵
0000000000000000↵
0000017EFCB71D96↵
00↵
02↵
01␣04␣04␣02468ACE↵
02␣20␣0C␣303132616263E6BCA2E5AD97
```

RX コマンドとして認識できるが構文異常の場合

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII コマンド Result	失敗内容に応じたコマンド固有メッセージ

何らかの問題があった際には、一部の問題ではその要因を便利 ASCII コマンド Result として応答します。問題によっては無い場合もあります。

メッセージ	内容
ILLEGAL␣PARAMETER	サポートされてない構文で指定された場合

便利 *ASCII* コマンド *Result* 例

```
LLIGAL␣PARAMETER
```

便利 ASCII レスポンス Result 構成

正常にコマンドを受信し、SIPF から受信できた場合のみ OK が返ります。

それ以外はすべて NG となります。

条件	応答文字列
成功時	「OK」(0x4f4b)
失敗時	「NG」(0x4e47)

便利 ASCII レスポンス Result 例

```
OK
```

RX コマンド～レスポンスまで例 1

```
$$RX↵┘
6EEC0C78C7D34F60A568AC6A7F9AE0F7↵┘
0000000000000000↵┘
0000017EFCB71D96↵┘
00↵┘
02↵┘
01␣04␣04␣02468ACE↵┘
02␣20␣0C␣303132616263E6BCA2E5AD97↵┘
OK↵┘
```

RX コマンド～レスポンスまで例 2

受信できるオブジェクト転送単位が無かった場合

```
$$RX↵┘
OK↵┘
```

FGET コマンド

モノプラットフォームのファイル送受信機能を使用して、モノプラットフォームからファイルを取得するコマンドです。モノプラットフォームで FILE_ID で登録したファイルを、FILE_ID を引数として指定して取得することができます。SIPF Firmware の UART インターフェイス上ではユーザーMCU に対して XMODEM プロトコルを用いたファイル送出機能を提供します。

このコマンドによりファイルの取得が開始されると、SIPF Firmware は UART インターフェイス側で XMODEM プロトコルの送信側として、ユーザーMCU からの送信要求を待ち受けるようになります。ユーザーMCU は XMODEM プロトコルに基づいて受信処理を行うことによりファイルを取得できます。XMODEM によるユーザーMCU への転送は、SIPF Firmware がモノプラットフォームから一部データを受信した段階でスタートします。

転送終了、もしくはエラーやキャンセルにより XMODEM プロトコルは終了し、便利 ASCII レスポンスペイロードと便利 ASCII レスポンス Result を応答してこのコマンドは終了します。

該当するファイルが無い、途中で回線切断/リトライ上限/タイムアウトなどのエラーになった、ユーザーMCU 側からキャンセルした、などファイルの受信成功以外はすべて失敗扱いとなり全て NG を返して終了します。

モノプラットフォームへのファイルの登録は、モノプラットフォームのプロジェクトの UI や、ユーザーデバイスからの FPUT コマンドで行うことができます。

便利 ASCII コマンドペイロード構成

要素	コマンド文字列	文字数	概要
便利 ASCII コマンドコード	FGET	4	ファイル受信コマンドを表します。
区切り	␣	1	
引数 1	FILE_ID	32 以下	モノプラットフォームでファイルを登録した時の FILE_ID を指定します。 FILE_ID 使用可能文字は「0-9」「a-z」「A-Z」「._」の ASCII 文字のみ使用可能で、「0-9」「a-z」「A-Z」で始まり、「.」は最大一つまで、最大 32 文字です。

- ASCII コマンドコード 「FGET」
- 必要引数　1
- 引数 1　FILE_ID

便利 ASCII コマンド文字列サンプル

```
$$FGET␣EXAMPLE.BIN↵
```

XMODEM プロトコル

- 対応モード：XMODEM/SUM 128bit_Data、8bit_Checksum
- タイムアウト ： 転送開始待ち時間 30sec
  FGET コマンドが発行され XMODEM の転送処理の開始後、ユーザーMCU が送信開始要求としての NAK を送信するまでの許容時間です。この間に NAK を受け取れなかった場合、SIPF Firmware は CAN を 2 つ発行して XMODEM の転送をキャンセルします。コマンドは失敗となります。
- タイムアウト/リトライ：データブロック 500ms、3 回
  SIPF Firmware からデータブロックを送出後、ユーザーMCU からの応答(ACK/NAK/CAN)する必要がある許容時間です。この間に応答が確認できなかった場合は SIPF Firmware はリトライとして同じデータブロックを再送しタイムアウトまで応答を待ちます。3 回のリトライでも応答が無い場合、SIPF Firmware は CAN を 2 つ発行して XMODEM の転送をキャンセルします。コマンドは失敗となります。
- キャンセル - SIPF Firmware 側からの場合：CAN 文字を 2 つ送信
  何らかの理由で SIPF Firmware が、XMODEM の転送をキャンセルしてコマンドを失敗させる必要が生じた時に行われます。文字化け等での取り損ないに備えて念のため 2 つ送出します。余分な CAN については破棄してください。CAN 送出後 SIPF Firmware は XMODEM の転送状態を抜け、NG を返したのちにコマンド待ち受けに戻ります。
- キャンセル - ユーザーMCU 側からの場合：CAN 文字を送信
  何らかの理由でユーザーMCU 側から、XMODEM の転送をキャンセルする必要が生じた際は SIPF Firmware に対して CAN を送出してください。CAN 受領後 SIPF Firmware は XMODEM の転送状態を抜け、NG を返したのちにコマンド待ち受けに戻ります。
- XMODEM の仕様として、受信ファイルが 16byte で割り切れない場合、最後の 16byte に満たないデータには EOF 文字でパディングし 16byte にそろえてユーザ MCU へ送出されます。ユーザ MCU 側では XMODEM 終了後のレスポンスデータの受信ファイルサイズ以降の、パディングされた EOF 文字を除去する必要があります。

便利 ASCII レスポンスペイロード構成

SIPF からのファイル受信成功時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	受信ファイルサイズのバイト数を 16 進 8 桁で表記

SIPF からのファイル受信失敗時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	なし

便利 ASCII レスポンスデータ例

受信したファイルサイズが 1024byte の場合

00000400

FGET コマンドとして認識できるが構文異常の場合

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII コマンド Result	失敗内容に応じた、コマンド共通メッセージ、もしくは、コマンド固有メッセージのいずれか

何らかの問題があった際には、一部の問題ではその要因を便利 ASCII コマンド Result として応答します。問題によっては無い場合もあります。

メッセージ	内容
ILLEGAL␣PARAMETER	サポートされてない構文で指定された場合

便利 *ASCII* コマンド *Result* 例

LLIGAL␣PARAMETER

便利 ASCII レスポンス Result 構成

正常にコマンドを受信し、モノプラットフォームからファイルを受信しユーザーMCU に対して XMODEM プロトコルでファイルを受け渡せた場合のみ OK が返ります。

それ以外はすべて NG となります。

条件	応答文字列
成功時	「OK」(0x4f4b)
失敗時	「NG」(0x4e47)

便利 ASCII レスポンス Result 例

```
OK
```

FGET コマンド～レスポンスまで例 1

正常にファイルを取得できた場合

```
$$FGET␣EXAMPLE.BIN ↵
 XMODEM プロトコルでの受信処理(成功)
00000400↵
OK↵
```

FGET コマンド～レスポンスまで例 2

ファイルの受信に何らかの原因で失敗した場合

```
$$FGET␣EXAMPLE.BIN ↵
 XMODEM プロトコルでの受信処理(失敗)
NG↵
```

FPUT コマンド

モノプラットフォームのファイル送受信機能を使用して、モノプラットフォームに対してファイルを送信するコマンドです。引数として指定した FILE_ID でモノプラットフォームにファイルを登録することができます。SIPF Firmware の UART インターフェイス上ではユーザーMCU に対して XMODEM プロトコルを用いたファイル受信機能を提供します。

このコマンドの開始により SIPF Firmware は UART インターフェイス側で XMODEM プロトコルの受信側として、ユーザーMCU からのファイルの送信を待ち受けるようになります。 ユーザーMCU は XMODEM プロトコルに基づいて送信処理を行うことによりファイルを送信できます。ユーザーMCU からファイルの送信が開始されると、SIPF Firmware はモノプラットフォームに対してファイルの送信を開始します。

転送終了、もしくはエラーやキャンセルにより XMODEM プロトコルは終了し、便利 ASCII レスポンスペイロードと便利 ASCII レスポンス Result を応答してこのコマンドは終了します。

不正な FILE_ID、途中で回線切断/リトライ上限/タイムアウトなどのエラーになった、ユーザーMCU 側からキャンセルした、などファイルの送信成功以外はすべて失敗扱いとなり全て NG を返して終了します。

この送信によりモノプラットフォームへ登録されたファイルは、モノプラットフォームのプロジェクトの UI や、ユーザーデバイスからの FGET コマンドで利用することができます。

便利 ASCII コマンドペイロード構成

要素	コマンド文字列	文字数	概要
便利 ASCII コマンドコード	FPUT	4	ファイル受信コマンドを表します。
区切り	␣	1	
引数 1	FILE_ID	32 以下	モノプラットフォームにファイルを登録する為の FILE_ID を指定します。 FILE_ID 使用可能文字は「0-9」「a-z」「A-Z」「._」の ASCII 文字のみ使用可能で、「0-9」「a-z」「A-Z」で始まり、「.」は最大一つまで、最大 32 文字です。
区切り	␣	1	
引数 2	FILE_SIZE	8	送信するファイルサイズを 16 進 8 桁で指定します。実際に送信されるデータ量と一致している必要があります。小さい値の場合も頭を 0 で埋めて 8 桁にする必要があります。

- ASCII コマンドコード「FPUT」
- 必要引数　2
- 引数１　FILE_ID
- 引数２　FILE_SIZE

便利 ASCII コマンド文字列サンプル

```
$$FPUT␣EXAMPLE.BIN␣000220C4↵
```

XMODEM プロトコル実装固有情報

- 対応モード：XMODEM/SUM 128bit_Data、8bit_Checksum
- タイムアウト/リトライ ： 転送開始要求後 3sec、10 回

  FPUT コマンドが発行される SIPF Firmware はユーザーMCU に対して送信開始要求としての NAK を送信します。この NAK に対してユーザーMCU がデータブロックで応答するまでの許容時間です。この間にデータブロックを受け取れなかった場合、SIPF Firmware は送信開始要求としての NAK の送信からリトライします。10 回のリトライでもデータブロックが確認できない場合、SIPF Firmware は CAN を２つ発行して XMODEM の転送をキャンセルします。コマンドは失敗となります。
- タイムアウト/リトライ：データブロック 1sec、 10 回

  SIPF Firmware から応答(ACK/NAK)を送出後、ユーザーMCU からの次のデータブロックの送出までの許容時間です。この間にデータブロックを受け取れなかった場合、SIPF Firmware はリトライとして NAK を送信し、同じデータブロックの再送をユーザーMCU に対して要求しデータブロックを待ちます。10 回のリトライでも応答が無い場合、SIPF Firmware は CAN を２つ発行して XMODEM の転送をキャンセルします。コマンドは失敗となります。
- キャンセル - SIPF Firmware 側からの場合：CAN 文字を２つ送信

  何らかの理由で SIPF Firmware が、XMODEM の転送をキャンセルしてコマンドを失敗させる必要が生じた時に行われます。文字化け等での取り損ないに備えて念のため２つ送出します。余分な CAN については破棄してください。CAN 送出後 SIPF Firmware は XMODEM の転送状態を抜け、NG を返したのちにコマンド待ち受けに戻ります。
- キャンセル - ユーザーMCU 側からの場合：CAN 文字を送信

  何らかの理由でユーザーMCU 側から、XMODEM の転送をキャンセルする必要が生じた際は SIPF Firmware に対して CAN を送出してください。CAN 受領後 SIPF Firmware は XMODEM の転送状態を抜け、NG を返したのちにコマンド待ち受けに戻ります。

- XMODEM の仕様として、送信ファイルが 16byte で割り切れない場合、最後の 16byte に満たないデータには EOF 文字でパディングし 16byte にそろえることが必要です。SIPF Firmware 側ではコマンド実行時に受け取ったファイルサイズを基準にこのパディングを除去してモノプラットフォームへ送信します。

便利 ASCII レスポンスペイロード構成

SIPF からのファイル受信成功時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	RSVD この通りの挙動ではない場合があります。(実際にモノプラットフォームに対して送信されたファイルサイズのバイト数を 16 進 8 桁で表記)

SIPF からのファイル受信失敗時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	なし

便利 ASCII レスポンスデータ例

RSVD この通りの挙動ではない場合があります。(モノプラットフォームに対して送信したファイルサイズが 1024byte の場合)

00000400

FPUT コマンドとして認識できるが構文異常の場合

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII コマンド Result	失敗内容に応じた、コマンド共通メッセージ、もしくは、コマンド固有メッセージのいずれか

何らかの問題があった際には、一部の問題ではその要因を便利 ASCII コマンド Result として応答します。問題によっては無い場合もあります。

メッセージ	内容
ILLEGAL␣PARAMETER	サポートされてない構文で指定された場合

便利 *ASCII* コマンド *Result* 例

```
LLIGAL␣PARAMETER
```

便利 ASCII レスポンス Result 構成

正常にコマンドを受信し、ユーザーMCU から XMODEM プロトコルでファイルを受けとりモノプラットフォームへファイルを送信できた場合のみ OK が返ります。

それ以外はすべて NG となります。

条件	応答文字列
成功時	「OK」(0x4f4b)
失敗時	「NG」(0x4e47)

便利 ASCII レスポンス Result 例

```
OK
```

FPUT コマンド～レスポンスまで例 1

正常にファイルを送信できた場合

```
$$FPUT␣EXAMPLE.BIN␣000220C4↵
 XMODEM プロトコルでの送信処理(成功)
000220C4↵
OK↵
```

FPUT コマンド～レスポンスまで例 2

ファイルの送信に何らかの原因で失敗した場合

```
$$FPUT␣EXAMPLE.BIN␣000220C4↵
 XMODEM プロトコルでの送信処理(失敗)
NG↵
```

UNLOCK コマンド

SIPF Firmware に対して致命的な影響を与える可能性のあるコマンドは、間違って実行されて障害を起こすことがないように、安全の為、通常では実行にロックがかかっており、常にエラーが返り実行することができません。これらの致命的な影響を与える可能性のあるコマンドは、実行直前に本 UNLOCK コマンドを実行することで実行することが可能になります。このコマンドを直前に実行しない限り SIPF Firmware に対して致命的な影響を与える可能性のあるコマンドはエラーのみ返る状態になっています。この手順により、致命的な影響を与える可能性のあるコマンドの実行がユーザーによる明示的なコマンドの発行であることを SIPF Firmware に対して通知することで事故の発生の低減を狙っています。

UNLOCK コマンド実行後、一定の条件を満たすとロック状態に戻ります。

- 種類やエラーの有無を問わず何らかのコマンドが実行された場合(解釈できずエラーとなる場合を含みます)
  ただし UNLOCK コマンドを実行された場合は引き続きロック解除状態が維持されます。

SIPF Firmware に対して致命的な影響を与える可能性のあるコマンドは、各コマンドの項にて UNLOCK してから実行するように記載がありますので、各コマンドの項を参照してください。

便利 ASCII コマンドペイロード構成

UNLOCK 設定時

要素	ASCII 文字列	サイズ	概要
便利 ASCII コマンドコード	UNLOCK	6	UNLOCK コマンドを表します。
区切り	␣	1	
引数 1	UNLOCK	6	UART 上でのデータ化けなどの影響で誤って SIPF Firmware 他のコマンドのペイロードの一部の UNLOCK という文字列をコマンドと解釈してしまうなどの問題を軽減するために正しい引数が渡されたときのみコマンドが成立するようになっています。 この引数が正しく、UNLOCK の文字列ではなかった場合エラーとなります。

- ASCII コマンドコード「UNLOCK」
- 必要引数　1
- 引数 1　文字列「UNLOCK」

便利 ASCII コマンド文字列サンプル

$$UNLOCK ␣UNLOCK↵

便利 ASCII レスポンスペイロード構成

UNLOCK 成功時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII レスポンスデータ	なし

正常にコマンドを受信し、UNLOCK が行えた場合は便利 ASCII レスポンスデータはありません。

コマンド成功時は便利 ASCII レスポンスデータを返しますが、本コマンドの成功時はこれが空なので、便利 ASCII レスポンスペイロードも空となります。

UNLOCK 失敗時

便利 ASCII レスポンスペイロード内要素	文字列
便利 ASCII コマンド Result	失敗内容に応じた、コマンド共通メッセージ、もしくは、コマンド固有メッセージのいずれか

何らかの問題があった際には、便利 ASCII コマンド Result を便利 ASCII レスポンスペイロードとして応答し、コマンド共通メッセージ、もしくは、コマンド固有メッセージのいずれかを返します。このコマンドでは共通メッセージとは別に下記のコマンド固有メッセージを返します。

メッセージ	内容
ILLEGAL PARAMETER	サポートされてない構文で指定された場合

便利 *ASCII* コマンド *Result* 例

ILLEGAL PARAMETER

便利 ASCII レスポンス Result 構成

正常にコマンドを受信し、UNLOCK できた場合のみ OK が返ります。

それ以外はすべて NG となります。

条件	応答文字列
成功時	「OK」(0x4f4b)
失敗時	「NG」(0x4e47)

便利 ASCII レスポンス Result 例

```
OK
```

UNLOCK コマンド～レスポンスまで例 1

```
$$UNLOCK ␣UNLOCK↵
OK↵
```

UNLOCK コマンド～レスポンスまで例 2

```
$$UNLOCK ␣LOCK↵
ILLEGAL PARAMETER ↵
NG↵
```

UPDATE コマンド

SIPF Firmware の更新確認、取得、更新を自動的に行うコマンドです。

このコマンドは SIPF Firmware に対して致命的な影響を与える可能性があるため、直前に UNLOCK コマンドの実行が必要となります。詳細は UNLOCK コマンドの項を参照してください。

SIPF から更新情報の取得、更新ファイルの取得、SIPF Firmware の書き換え処理を連続で自動的に行うため、応答が完了するまでに、環境や状況次第で数十 sec～数 min 程度の時間がかかります。この間は一切の操作を受け付けません。完全に更新が完了するか、途中でのエラーで停止するまで待つ必要があります。

UPDATE コマンドを実行中の間は、十分に安定した電源を供給した状態で、ハードウェアリセットや電源断などを行わない/発生しないよう十分に注意してください。更新作業に失敗し、復旧不能な状態になる可能性があります。

このコマンド UPDATE コマンドは更新ファイルの取得のためにファイル送受信機能を利用します。更新対象のデバイスが含まれるモノプラットフォームのプロジェクトに対して事前にファームウェアの更新用ファイルを登録しておく必要があります。

UPDATE コマンドの更新ファイル取得方法には 2 種類あり、規定の FILE_ID(app_update.bin)で登録された更新ファイルを取得して更新する「UPDATE」と、ユーザーMCU 側で任意の FILE_ID で更新ファイルを指定して取得して更新する「VERSION」とがあります。「UPDATE」はプロジェクト内の全デバイスで共通の SIPF Firmware を使用する際などで最適です。「VERSION」の場合では FILE_ID 違いで複数の異なる更新ファイルをプロジェクト内に用意でき更新時に選択することができるので、プロジェクト内のデバイスにより異なるバージョンや仕様の SIPF Firmware を使い分ける必要があるときに最適です。どちらの場合でも各 FILE_ID に割り当てる更新ファイルはユーザー自身で明示的に事前にアップロードする必要があります。

該当するファイルが無い、途中で回線切断/リトライ上限/タイムアウトなどのエラーになった、適切なファームウェアファイルではない、などファームウェアの更新成功以外はすべて失敗扱いとなり全て NG を返して終了します。

モノプラットフォームへのファームウェアファイルの登録は、モノプラットフォームのプロジェクトにセットされているサービスアダプターや、ユーザーデバイスからの FPUT コマンドで行うことができます。

動作と書き込みに支障のない最低限の要件さえ満たしていれば、ユーザーカスタムしたファームウェアや SIPF Firmware と互換の無い任意のユーザーFW に書き換えることも可能ですが、ユーザーカスタム FW やユーザー独自 FW への書き換えは十分にユーザー自身にて評価/確認したうえでユーザーの判断にてご利用ください。書き込みに必要な要件は SIPF Firmware の github (https://github.com/sakura-internet/sipf-std-client_nrf9160)を参考にしてください。ユーザーカスタム FW やユーザー独自 FW でのモノプラットフォームの利

用自体は大変歓迎いたしますが、この場合の動作についても SIPF Firmware 自体と同様にサポートについてはできかねますのでご了承ください。

またこのコマンドのみ便利 ASCII レスポンスが他の便利 ASCII コマンドと異なる独自形式となります。詳細は UPDATE 独自 便利 ASCII レスポンスの項を参照してください。

便利 ASCII コマンドペイロード構成

基本構成：「UPDATE」の場合

要素	ASCII 文字列	サイズ	概要
便利 ASCII コマンドコード	UPDATE	6	UPDATE コマンドを表します。
区切り	␣	1	
引数 1	UPDATE	6	既定の FILE_ID(app_update.bin)を指定して更新ファイルを取得/適用するモードです。

基本構成：「VERSION」の場合

要素	ASCII 文字列	サイズ	概要
便利 ASCII コマンドコード	UPDATE	6	UPDATE コマンドを表します。
区切り	␣	1	
引数 1	VERSION	7	引数 2 の文字列で FILE_ID を指定して更新ファイルを取得/適用するモードです。
区切り	␣	1	
引数 2	FILE_ID	32 以下	モノプラットフォームでファイルを登録した時の FILE_ID を指定します。 FILE_ID 使用可能文字は「0-9」「a-z」「A-Z」「._」の ASCII 文字のみ使用可能で、「0-9」「a-z」「A-Z」で始まり、「.」は最大一つまで、最大 32 文字です。

- ASCII コマンドコード 「UPDATE」
- 必要引数　1 or 2

引数 1　文字列「UPDATE」 or「VERSION」

引数 2 「VERSION」時のみ：FILE_ID 文字列

便利 ASCII コマンド文字列サンプル

```
$$UPDATE␣UPDATE↵
```

```
$$UPDATE␣VERSION␣FIRMWARE_V987.BIN↵
```

UPDATE 独自 便利 ASCII レスポンス

このコマンドの便利 ASCII レスポンスは他の便利 ASCII コマンドのレスポンスと異なり、独自形式となります。

UPDATE コマンド成功時

便利 ASCII レスポンス内要素	文字列
UPDATE 更新進捗表示	UPDATE コマンドの進捗を表すメッセージ。 処理状況に合わせて現在の状況を順次表示する。 文字数は不定です。
起動完了 ASCII レスポンス	SIPF Firmware の起動メッセージ。 詳細は、起動中～起動完了メッセージの項を参照してください。

UPDATE コマンド成功時サンプル便利 *ASCII* レスポンス *1*

「UPDATE」の場合

```
DOWNLOAD FILE: app_update.bin
FOTA DOWNLOAD START
  0% DOWNLOADED
  0% DOWNLOADED
  1% DOWNLOADED
  1% DOWNLOADED
  2% DOWNLOADED
  2% DOWNLOADED
  3% DOWNLOADED
  3% DOWNLOADED
  ～～～～～～～～～～～～～～～～
  ～～～～～～～～～～～～～～～～
 90% DOWNLOADED
 90% DOWNLOADED
 91% DOWNLOADED
 91% DOWNLOADED
 92% DOWNLOADED
 92% DOWNLOADED
 93% DOWNLOADED
 93% DOWNLOADED
 94% DOWNLOADED
 94% DOWNLOADED
 95% DOWNLOADED
 95% DOWNLOADED
 96% DOWNLOADED
 96% DOWNLOADED
 97% DOWNLOADED
 97% DOWNLOADED
 98% DOWNLOADED
 98% DOWNLOADED
 99% DOWNLOADED
 99% DOWNLOADED
 99% DOWNLOADED
FOTA DOWNLOAD FINISHED
*** Booting Zephyr OS build v2.6.99-ncs1-1  ***
I: Starting bootloader
I: Primary image: magic=good, swap_type=0x2, copy_done=0x1, image_ok=0x1
I: Secondary image: magic=good, swap_type=0x2, copy_done=0x3, image_ok=0x3
I: Boot source: none
I: Swap type: test
I: Bootloader chainload address offset: 0x10000
*** Booting Zephyr OS build v2.6.99-ncs1-1  ***
Flash regions       Domain      Permissions
00 03 0x00000 0x20000   Secure      rwxl
04 31 0x20000 0x100000  Non-Secure  rwxl

Non-secure callable region 0 placed in flash region 3 with size 32.

SRAM region         Domain      Permissions
00 07 0x00000 0x10000   Secure      rwxl
08 31 0x10000 0x40000   Non-Secure  rwxl

Peripheral          Domain      Status
00 NRF_P0           Non-Secure  OK
01 NRF_CLOCK        Non-Secure  OK
02 NRF_RTC0         Non-Secure  OK
03 NRF_RTC1         Non-Secure  OK
04 NRF_NVMC         Non-Secure  OK
05 NRF_UARTE1       Non-Secure  OK
06 NRF_UARTE2       Secure      SKIP
07 NRF_TWIM2        Non-Secure  OK
08 NRF_SPIM3        Non-Secure  OK
09 NRF_TIMER0       Non-Secure  OK
10 NRF_TIMER1       Non-Secure  OK
11 NRF_TIMER2       Non-Secure  OK
12 NRF_SAADC        Non-Secure  OK
13 NRF_PWM0         Non-Secure  OK
14 NRF_PWM1         Non-Secure  OK
15 NRF_PWM2         Non-Secure  OK
16 NRF_PWM3         Non-Secure  OK
17 NRF_WDT          Non-Secure  OK
18 NRF_IPC          Non-Secure  OK
19 NRF_VMC          Non-Secure  OK
20 NRF_FPU          Non-Secure  OK
21 NRF_EGU1         Non-Secure  OK
22 NRF_EGU2         Non-Secure  OK
23 NRF_DPPIC        Non-Secure  OK
24 NRF_REGULATORS   Non-Secure  OK
25 NRF_PDM          Non-Secure  OK
26 NRF_I2S          Non-Secure  OK
27 NRF_GPIOTE1      Non-Secure  OK

SPM: NS image at 0x20200
SPM: NS MSP at 0x20020310
SPM: NS reset vector at 0x26da1
SPM: prepare to jump to Non-Secure image.
*** SIPF Client(Type00) v.0.4.0 ***
* PLMN: 44020
Trying to attach to LTE network (TIMEOUT: 120000 ms)
SEARCHING
REGISTERD
ICCID: 8981040000000260815
 +++ Ready +++
```

UPDATE コマンド成功時サンプル便利 *ASCII* レスポンス 2

「VERSION」の場合

```
DOWNLOAD FILE: FIRMWARE_V987.BIN
FOTA DOWNLOAD START
  0% DOWNLOADED
  0% DOWNLOADED
  1% DOWNLOADED
  1% DOWNLOADED
  2% DOWNLOADED
  2% DOWNLOADED
  3% DOWNLOADED
 ～～～～～～～～～～～～～～
 ～～～～～～～～～～～～～～
 93% DOWNLOADED
 93% DOWNLOADED
 94% DOWNLOADED
 94% DOWNLOADED
 95% DOWNLOADED
 95% DOWNLOADED
 96% DOWNLOADED
 96% DOWNLOADED
 97% DOWNLOADED
 97% DOWNLOADED
 98% DOWNLOADED
 98% DOWNLOADED
 99% DOWNLOADED
 99% DOWNLOADED
 99% DOWNLOADED
FOTA DOWNLOAD FINISHED
□*** Booting Zephyr OS build v2.6.99-ncs1-1  ***
I: Starting bootloader
I: Primary image: magic=good, swap_type=0x2, copy_done=0x1, image_ok=0x1
I: Secondary image: magic=good, swap_type=0x2, copy_done=0x3, image_ok=0x3
I: Boot source: none
I: Swap type: test
I: Bootloader chainload address offset: 0x10000
*** Booting Zephyr OS build v2.6.99-ncs1-1  ***
Flash regions          Domain         Permissions
00 03 0x00000 0x20000   Secure         rwxl
04 31 0x20000 0x100000  Non-Secure      rwxl

Non-secure callable region 0 placed in flash region 3 with size 32.

SRAM region            Domain         Permissions
00 07 0x00000 0x10000   Secure         rwxl
08 31 0x10000 0x40000   Non-Secure      rwxl

Peripheral             Domain         Status
00 NRF_P0              Non-Secure      OK
01 NRF_CLOCK           Non-Secure      OK
02 NRF_RTC0            Non-Secure      OK
03 NRF_RTC1            Non-Secure      OK
04 NRF_NVMC            Non-Secure      OK
05 NRF_UARTE1          Non-Secure      OK
06 NRF_UARTE2          Secure         SKIP
07 NRF_TWIM2           Non-Secure      OK
08 NRF_SPIM3           Non-Secure      OK
09 NRF_TIMER0          Non-Secure      OK
10 NRF_TIMER1          Non-Secure      OK
11 NRF_TIMER2          Non-Secure      OK
12 NRF_SAADC           Non-Secure      OK
13 NRF_PWM0            Non-Secure      OK
14 NRF_PWM1            Non-Secure      OK
15 NRF_PWM2            Non-Secure      OK
16 NRF_PWM3            Non-Secure      OK
17 NRF_WDT             Non-Secure      OK
18 NRF_IPC             Non-Secure      OK
19 NRF_VMC             Non-Secure      OK
20 NRF_FPU             Non-Secure      OK
21 NRF_EGU1            Non-Secure      OK
22 NRF_EGU2            Non-Secure      OK
23 NRF_DPPIC           Non-Secure      OK
24 NRF_REGULATORS      Non-Secure      OK
25 NRF_PDM             Non-Secure      OK
26 NRF_I2S             Non-Secure      OK
27 NRF_GPIOTE1         Non-Secure      OK

SPM: NS image at 0x20200
SPM: NS MSP at 0x200203f0
SPM: NS reset vector at 0x26da1
SPM: prepare to jump to Non-Secure image.
*** SIPF Client(Type00) v.0.4.0 ***
* PLMN: 44020
Trying to attach to LTE network (TIMEOUT: 120000 ms)
SEARCHING
REGISTERD
ICCID: 8981040000000260815
+++ Ready +++
```

UPDATE コマンド失敗時

便利 ASCII レスポンス内要素	文字列
ILLEGAL PARAMETER	サポートされてない構文で指定された場合
UPDATE 更新進捗表示	UPDATE コマンドの進捗を表すメッセージ。 失敗時点までの進捗を表示する。 失敗タイミング次第では何もメッセージが出ない場合があります。

UPDATE* コマンド失敗時サンプル便利 *ASCII* レスポンス *1

```
ILLEGAL PARAMETER
```

UPDATE* コマンド失敗時サンプル便利 *ASCII* レスポンス *2

SIPF Firmware と無関係なファイルがアップロードされていた場合

```
$$UNLOCK UNLOCK
OK
$$UPDATE VERSION FIRMWARE_V987.BIN
DOWNLOAD FILE: FIRMWARE_V987.BIN
FOTA DOWNLOAD START
FOTA DOWNLOAD FAILED
NG
```

便利 ASCII レスポンス Result 構成

更新に失敗した時に NG が返ります。

成功した時には NG の表示が無く、自動的に再起動され bootloder による書きかえと SIPF Firmware の起動メッセージが表示されます。

基本的には bootloder に処理が渡る前に書き込めないファイルは NG になりますが、ごくまれに bootloder でのチェックにて破損を検出し書き換えが行われないことがあります。確実な判定が必要な際には bootloder のメッセージでの判定を行ってください。

条件	応答文字列
失敗時	「NG」(0x4e47)

UPDATE コマンド～レスポンスまで例 1

「UPDATE」の場合

```
$$UNLOCK UNLOCK
OK
$$UPDATE UPDATE
DOWNLOAD FILE: app_update.bin
FOTA DOWNLOAD START
  0% DOWNLOADED
  0% DOWNLOADED
  1% DOWNLOADED
  1% DOWNLOADED
  2% DOWNLOADED
  2% DOWNLOADED
  3% DOWNLOADED
  3% DOWNLOADED
 ～～～～～～～～～～～～～～～～
 ～～～～～～～～～～～～～～～～
 90% DOWNLOADED
 90% DOWNLOADED
 91% DOWNLOADED
 91% DOWNLOADED
 92% DOWNLOADED
 92% DOWNLOADED
 93% DOWNLOADED
 93% DOWNLOADED
 94% DOWNLOADED
 94% DOWNLOADED
 95% DOWNLOADED
 95% DOWNLOADED
 96% DOWNLOADED
 96% DOWNLOADED
 97% DOWNLOADED
 97% DOWNLOADED
 98% DOWNLOADED
 98% DOWNLOADED
 99% DOWNLOADED
 99% DOWNLOADED
 99% DOWNLOADED
FOTA DOWNLOAD FINISHED
 *** Booting Zephyr OS build v2.6.99-ncs1-1  ***
I: Starting bootloader
I: Primary image: magic=good, swap_type=0x2, copy_done=0x1, image_ok=0x1
I: Secondary image: magic=good, swap_type=0x2, copy_done=0x3, image_ok=0x3
I: Boot source: none
I: Swap type: test
I: Bootloader chainload address offset: 0x10000
*** Booting Zephyr OS build v2.6.99-ncs1-1  ***
Flash regions         Domain          Permissions
00 03 0x00000 0x20000   Secure          rwxl
04 31 0x20000 0x100000  Non-Secure      rwxl

Non-secure callable region 0 placed in flash region 3 with size 32.

SRAM region           Domain          Permissions
00 07 0x00000 0x10000   Secure          rwxl
08 31 0x10000 0x40000   Non-Secure      rwxl

Peripheral            Domain          Status
00 NRF_P0             Non-Secure      OK
01 NRF_CLOCK          Non-Secure      OK
02 NRF_RTC0           Non-Secure      OK
03 NRF_RTC1           Non-Secure      OK
04 NRF_NVMC           Non-Secure      OK
05 NRF_UARTE1         Non-Secure      OK
06 NRF_UARTE2         Secure          SKIP
07 NRF_TWIM2          Non-Secure      OK
08 NRF_SPIM3          Non-Secure      OK
09 NRF_TIMER0         Non-Secure      OK
10 NRF_TIMER1         Non-Secure      OK
11 NRF_TIMER2         Non-Secure      OK
12 NRF_SAADC          Non-Secure      OK
13 NRF_PWM0           Non-Secure      OK
14 NRF_PWM1           Non-Secure      OK
15 NRF_PWM2           Non-Secure      OK
16 NRF_PWM3           Non-Secure      OK
17 NRF_WDT            Non-Secure      OK
18 NRF_IPC            Non-Secure      OK
19 NRF_VMC            Non-Secure      OK
20 NRF_FPU            Non-Secure      OK
21 NRF_EGU1           Non-Secure      OK
22 NRF_EGU2           Non-Secure      OK
23 NRF_DPPIC          Non-Secure      OK
24 NRF_REGULATORS     Non-Secure      OK
25 NRF_PDM            Non-Secure      OK
26 NRF_I2S            Non-Secure      OK
27 NRF_GPIOTE1        Non-Secure      OK

SPM: NS image at 0x20200
SPM: NS MSP at 0x200203f0
SPM: NS reset vector at 0x26da1
SPM: prepare to jump to Non-Secure image.
*** SIPF Client(Type00) v.0.4.0 ***
* PLMN: 44020
Trying to attach to LTE network (TIMEOUT: 120000 ms)
SEARCHING
REGISTERD
ICCID: 89810400000000260815
+++ Ready +++
```

UPDATE コマンド～レスポンスまで例 2

```
$$UPDATE␣UPDATE␣FIRMWARE␣V987.BIN ↵
ILLEGAL PARAMETER ↵
```

オブジェクト送受で使用する型指定 ID と表記例一覧

型	型 ID	型 ID 表記	型名	バイナリサイズ	文字数	型表記例
8bit 符号なし整数	0x00	00	uint8	1	2	16 進 2 桁で表記 例：10 進で 10 の場合：0A
8bit 符号付き整数	0x01	01	int8	1	2	16 進 2 桁で表記 例：10 進で-10 の場合：F6
16bit 符号なし整数	0x02	02	uint16	2	4	16 進 4 桁で表記 例：10 進で 10 の場合：000A
16bit 符号付き整数	0x03	03	int16	2	4	16 進 4 桁で表記 例：10 進で-10 の場合：FFF6
32bit 符号なし整数	0x04	04	uint32	4	8	16 進 8 桁で表記 例：10 進で 10 の場合：0000000A
32bit 符号付き整数	0x05	05	int32	4	8	16 進 8 桁で表記 例：10 進で-10 の場合：FFFFFFF6
64bit 符号なし整数	0x06	06	uint64	8	16	16 進 16 桁で表記 例：10 進で 10 の場合： 000000000000000A
64bit 符号付き整数	0x07	07	int64	8	16	16 進 16 桁で表記 例：10 進で-10 の場合：FFFFFFFFFFFFFFF6
IEEE754 32bit 浮動小数点数	0x08	08	float32	4	8	IEEE754 単精度浮動小数点形式 を 16 進 8 桁で表記 例：10 進で-1.1 の場合：BFF19999
IEEE754 64bit 浮動小数点数	0x09	09	float64	8	16	IEEE754 倍精度浮動小数点形式 を 16 進 16 桁で表記 例：10 進で-1.1 の場合： FF199999999999A
バイナリ	0x10	10	binary	n	2n	任意長(※)バイナリ列を 16 進表記で表記 例：0x303132616263 のバイナリ列の場合： 303132616263
utf8 文字列	0x20	20	string.utf8	m	2m	任意長(※)UTF8 文字列形式を 16 進表記で表記 例：012abc 漢字 の場合： 303132616263E6BCA2E5AD97

※最大 510 文字以下にする必要があります。

改訂履歴

日付	版	改訂内容
2022 年 03 月 24 日	1.0.0	初版公開