

DS ダウンロードプレイ プロトコル仕様解説

Ver 1.0.0

任天堂株式会社発行

このドキュメントの内容は、機密情報であるため、
厳重な取り扱い、管理を行ってください。

目次

1	はじめに	5
1.1	概要	5
2	DSダウンロードプレイの動作	6
2.1	ロビー情報の制御	6
2.1.1	親機ロビー情報配信	7
2.1.2	子機ロビー情報列挙	8
2.2	セッションへの参加とダウンロード	9
2.2.1	子機ロビー接続	10
2.2.2	子機セッション参加要求	10
2.2.3	親機セッション参加許可	10
2.2.4	プログラムダウンロード	10
2.2.5	子機プログラム再起動	10
3	WDPデータフォーマット定義	11
3.1	基本型	11
3.2	WDPIconDataFormat	12
3.3	WDPPlayerDataFormat	13
3.4	WDPBeaconAttributeFormat	14
3.5	WDPSessionDataFixedFormat	15
3.6	WDPBeaconFormat	16
3.7	WDPEntryRequestFormat	18
3.8	WDPSegmentTableFormat	19
3.9	WDPSegmentFormat	20
3.10	WDPPacketFormat	21
3.11	WDPパケットコマンド	23
4	プロトコル仕様および制限事項	24
4.1	基本的な仕様	24
4.1.1	ダウンロードプログラムの配置可能領域	24
4.1.2	セッション情報の一意性の保証	24
4.1.3	ダウンロード要求コマンドのAID	25
4.2	IPLダウンロードプログラムの制限	25
4.2.1	子機ダウンロードプログラム再起動アドレス	25
4.2.2	MP通信における送受信サイズの制限	26



図 1-1	DSダウンロードプレイ概念図	5
図 2.1-1	ロビーとセッション、プレイヤーの関係	6
図 2.1-2	親機ビーコン送信	7

図 2.1-3	ビーコンストリーム内のビーコン順序	7
図 2.1-4	ビーコン取得とセッション情報の再構築	8
図 2.1-5	可変情報の更新	8
図 2.2-1	WDPのMP通信プロトコルシーケンス	9
図 3.11-1	WDPパケットコマンド一覧	23

改訂履歴

版	改訂日	改 訂 内 容	担当者
1.0.0	2006-03-16	初版	吉崎

1 はじめに

本ドキュメントは DS ダウンロードプレイ用ライブラリの作成・保守・拡張・移植の際に必要な内部開発者向け資料です。ゲームアプリケーション開発者向けの一般的な情報については別途 [AboutMultiboot.pdf](#) の資料を参照してください。

1.1 概要

DS ダウンロードプレイは、本体 IPL をクライアントとしてゲームプログラムを配信し、あるいは単に簡易的なローカルゲーム通信用セッションの構築手段としても使用可能な通信プロトコルです。処理の大まかな流れを以下に示します。

- (1) 親機ロビー情報配信
- (2) 子機ロビー情報列挙
- (3) 子機ロビー接続
子機は得られた情報をもとにロビー親機へ MP 通信接続します。
- (4) 子機セッション参加要求
子機は MP 通信パケットとしてセッション参加コマンドを親機へ送出します。
- (5) 親機セッション参加許可・ゲームダウンロード
親機はセッション参加コマンド内のプレイヤー情報をもとに参加許可し、ゲームバイナリを分割送信します。
- (6) 子機プログラム再起動
ゲームバイナリを全て受信した子機は親機の指示を受けて切断し、プログラムを再起動します。

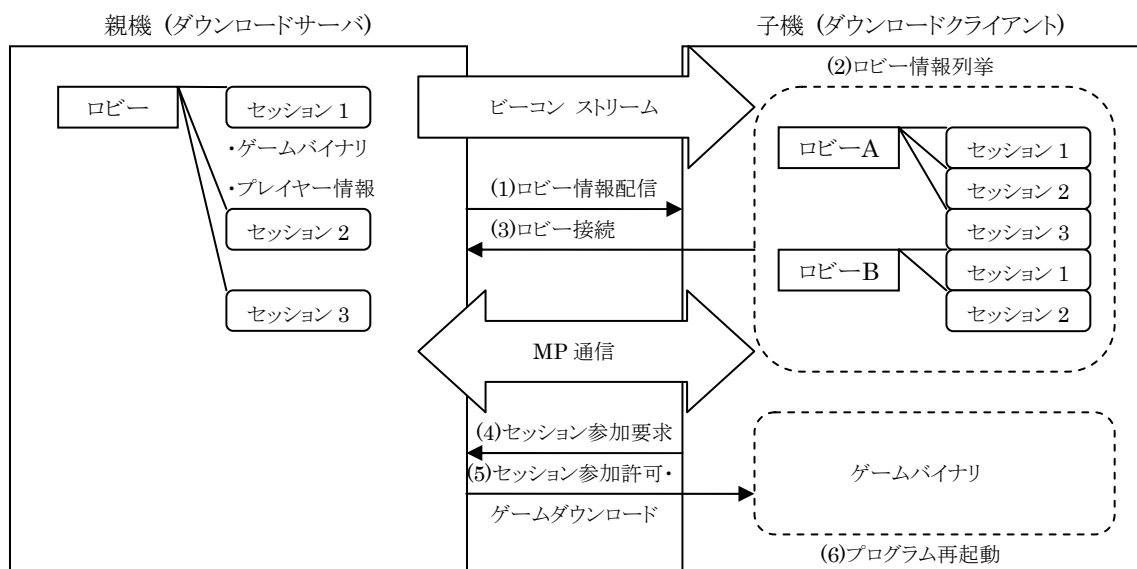


図 1-1 DS ダウンロードプレイ概念図

2 DS ダウンロードプレイの動作

本章では「[1.1 概要](#)」で述べた内容をさらに詳細に解説し、ビーコン送受信とMP通信を使用してDSダウンロードプレイを実現するWDPライブラリの一連の動作の流れを示します。ワイヤレスライブラリ自体の具体的な制御方法についてはここでは触れません。

2.1 ロビー情報の制御

親機は単一の【ロビー】として1個以上の【セッション】を保持します。

セッションにはダウンロードプログラムの詳細や参加済みプレイヤー状況などの情報が含まれます。

子機は特定のロビーに接続し、特定のセッション1個にプレイヤーとして参加することができます。

親機は自身のロビーに含まれる全てのセッションに常に参加しているものとみなされます。

子機からは個々のロビーの一意性を MAC アドレスと TGID の対で判定し、ロビー内の個々のセッションの一意性は 6bit の ID 値で判定します。このため、単一のロビーが同時に保持可能なセッションは最大 64(2 の 6 乗)個に制限されます。

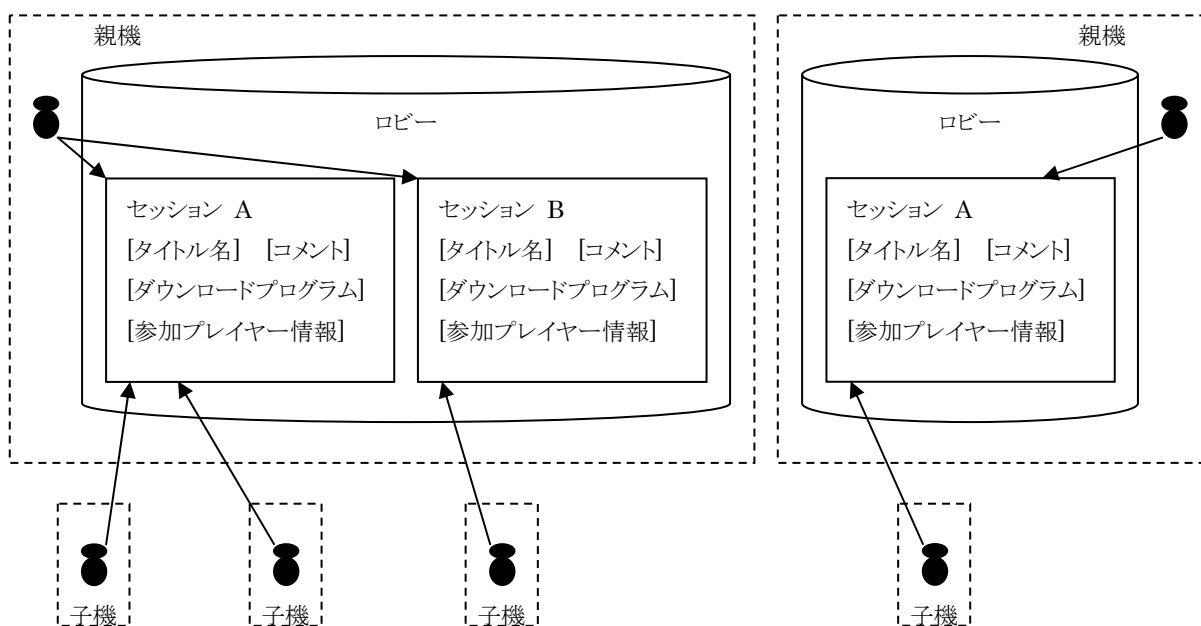


図 2.1-1 ロビーとセッション、プレイヤーの関係

2.1.1 親機ロビー情報配信

親子の通信確立にあたっては、まず子機がロビー親機が存在を認識してセッションに参加する必要があります。
ロビーの情報は親機から子機へビーコンによって一方方向に配信されます。

親機はロビーに含まれるおのおののセッション情報をビーコンデータに分割して常時配信します。

例えば A、B、C 3 個のセッションを含むロビーならば $A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow C \rightarrow A \rightarrow \dots$ と巡回的にセッションが選択され、この選択されたセッションの情報をひとつづきの分割ビーコンの束として送信します。以降、この分割ビーコンの束を【ビーコンストリーム】と呼びます。

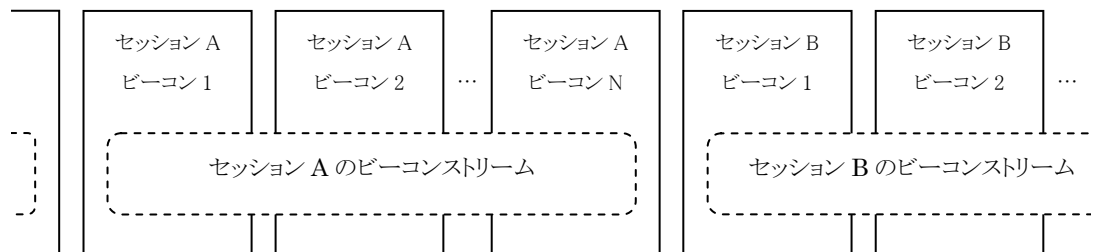


図 2.1-2 親機ビーコン送信

ビーコンストリーム内のフォーマットは決まっており、前半にはセッション内で変更されない固定的な情報が送信され、後半には最新の参加プレイヤー情報が可変長で送信されます。以降、この前半部分の情報を【固定情報】、後半部分の情報を【可変情報】と呼びます。

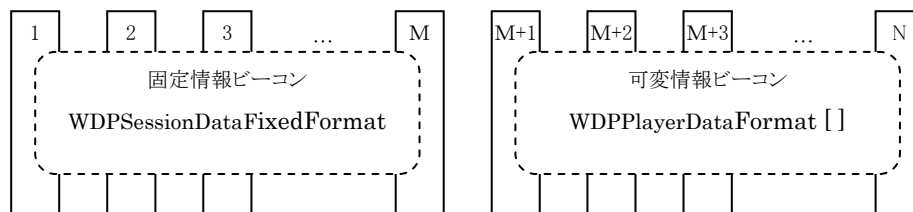


図 2.1-3 ビーコンストリーム内のビーコン順序

2.1.2 子機ロビー情報列举

子機は分割されたビーコンの断片を収集してロビーとセッションの情報を再構築します。

ビーコンは WDP 親機のものでない場合もあり、同一チャンネル上に複数の WDP 親機が存在する場合があります。

また、子機側では同一内容の断片を重複して受信したり、一部を取りこぼしたりする場合があります。

これらのビーコン内容判定と断片情報の蓄積は WDP ライブラリ内部で行われます。

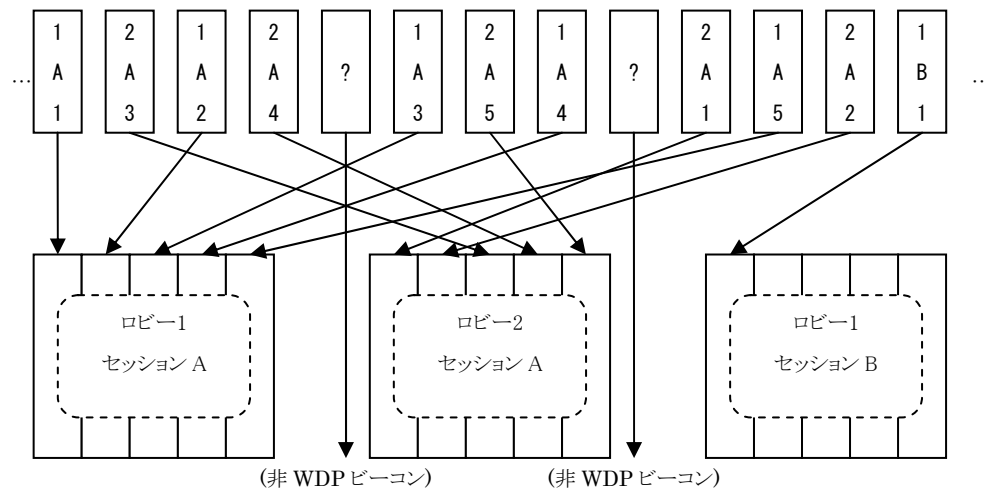


図 2.1-4 ビーコン取得とセッション情報の再構築

セッション情報のうちの固定情報部分については1回すべての断片を取得すれば再び収集する必要はありませんが、可変情報はプレイヤーの参加や退出に応じて動的に更新されるためその都度再取得する必要があります。

新旧の可変情報断片の混在収集を避けるため、可変情報ビーコンにはインクリメンタルなカウンタ値が含まれます。

同じカウンタ値である全ての断片を取得した時点ではじめて、有効な可変情報全体を取得したとみなされます。

その後はカウンタ値が同じである可変情報ビーコンは無視し、カウンタ値の変化を認識すれば再取得を開始しますが、新しいカウンタ値が前回の値+1 の場合には差分情報に相当する部分のみ取得すればよいので、再び情報が有効になるまでに要する更新時間は初回取得時よりも短くなります。この差分更新処理が追いつかない頻度でセッション情報が更新された場合、すなわちカウンタ値が前回の値+1 とならない場合には、取得済みの有効な可変情報を破棄して無効な状態とみなし、全ての断片を再取得します。

可変情報の有効化と無効化のタイミングで、WDP ライブラリからアプリケーションへ通知が発生します。

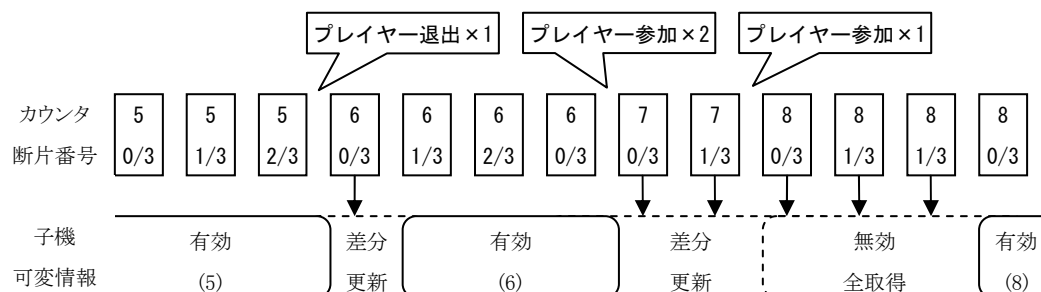


図 2.1-5 可変情報の更新

2.2 セッションへの参加とダウンロード

前項までの手順で、子機側には親機のロビーとセッションの情報が配信された状態になります。

この情報をもとに子機は親機へ接続し、以降の手順で MP 通信を行い WDP のプロトコルを進めます。

アプリケーションからの指定を待機する数箇所を除き、基本的にプロトコルシーケンスは MP 通信パケットの交換のみで自動的に進行します。

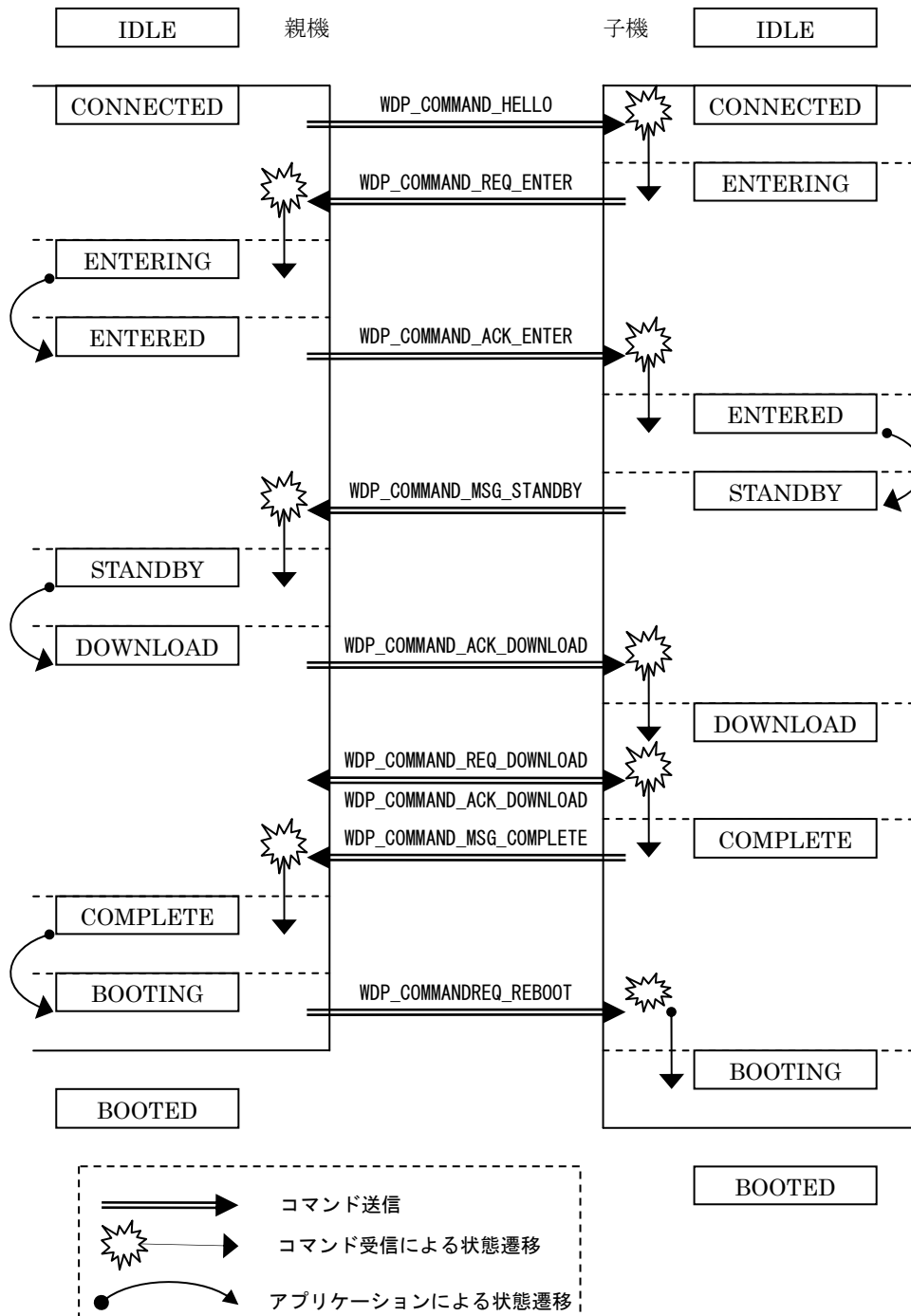


図 2.2-1 WDP の MP 通信プロトコルシーケンス

2.2.1 子機ロビー接続

子機はローカル MP 通信としてロビー親機へ接続する必要があります。

ローカル MP 通信接続に必要な WMBssDesc 情報は、WDP がビーコン収集時に保持したものを使用できます。

接続後は親子とも MP 通信パケット上でコマンドを発行し続けますが、この処理は WDP ライブラリ内部で自動的に行われるため、アプリケーションが直接このフォーマットを理解する必要はありません。

2.2.2 子機セッション参加要求

通信確立後、子機はセッション参加要求コマンド `WDP_COMMAND_REQ_ENTER` を発行します。

このコマンドには子機自身のプレイヤー情報とセッション ID が引数として指定されますが、これらの情報は複数のパケットに分割して親機へ送信されます。親機は分割された断片を全て取得した時点で子機のコマンド発行を認識します。

2.2.3 親機セッション参加許可

前項で受信したセッション参加要求を許可するか拒否するかは、最終的にはアプリケーションで判断します。

まず、参加を要求されたセッション ID に合致するセッションが存在しない場合は内部で自動的にセッション参加拒否メッセージ `WDP_COMMAND_MSG_REFUSE` を返信発行します。また、コマンド受信時にすでにセッション参加人数が上限である場合には内部で自動的にセッション参加人数上限メッセージ `WDP_COMMAND_MSG_MAXENTRY` を返信発行します。

上記いずれにも該当しない場合、すなわち指定されたセッションが存在し参加人数に余裕がある場合には、WDP ライブラリからアプリケーションへ通知が発生します。以降、アプリケーションが `WDP-AllowChildStatus` 関数を呼び出して許可か拒否を指定するまで(あるいは何らかの理由で通信が切断されるまで)、親機と対象子機との間の通信プロトコルはこの「セッション参加待ち」状態を維持します。

セッション参加を許可すると、親機は対象子機へセッション参加応答 `WDP_COMMAND_ACK_ENTER` を発行します。

2.2.4 プログラムダウンロード

前項のとおり子機がセッション参加応答を受信すると子機側アプリケーションでは WDP ライブラリから通知が発生します。この段階で、アプリケーションは WDP プロトコルの次のシーケンスであるプログラムダウンロードの形式を以下のいずれかから選択する必要があります。

(1) プログラムの全てのセグメントをダウンロードする

WDP 子機としての通常の形式です。この形式を選択する子機はプログラム全てのセグメント(ARM9 常駐セグメント、ARM7 常駐セグメント、ROM 内登録データ)が連結された巨大なデータブロックを受信するための連続したメモリ領域を確保しておく必要があり、プログラム構成は大きな制約を受けますことになります。

(2) プログラムの全てのセグメントのダウンロード処理をスキップする

WDP のセッション構築機能のみを利用し、プログラムダウンロード機能が不要な場合に選択する形式です。上記(1)で必要とされた受信用メモリは必要ありません。この形式を、特に【擬似ダウンロード子機(fake-child)】と呼ぶ場合があります。

なお、ひとつのセッションに上記(1)と(2)が混在することも可能です。

2.2.5 子機プログラム再起動

プログラムダウンロードが完了すると、親機側アプリケーションが `WDP-AllowChildStatus` 関数を呼び出して再起動を要求するまで子機は待機状態になります。子機が再起動要求 `WDP_COMMAND_REQ_REBOOT` を受信するとアプリケーションに通知が発生し、ここで WDP プロトコルは終了状態となります。

3 WDP データフォーマット定義

本章では WDP の通信プロトコルを構成するデータフォーマットを解説します。

ここでは WDP プロトコル実装の際にプラットフォームに依存せず必須となる変更不可能な仕様情報のみを扱っており、「ニンテンドーDS 版 WDP ライブラリ」固有のインタフェースに依存した変更可能な実装情報は除外しています。異なるプラットフォーム間で WDP プロトコルの互換性を維持する際はこの点に注意する必要があります。

3.1 基本型

ここでは、以降の解説に使用する WDP の基本的なデータ型を定義します。

【定義】

```
typedef unsigned char  u8;  
typedef unsigned short u16;  
typedef u16 WDPChar;  
typedef struct WDPu16LE { u8 byte[2]; } WDPu16LE;  
typedef struct WDPu32LE { u8 byte[4]; } WDPu32LE;
```

【説明】

メンバ	意味
u8	プラットフォームに依存した、符号無し 8bit の組み込み型です。
u16	プラットフォームに依存した、符号無し 16bit の組み込み型です。
WDPChar	WDP プロトコルで使用する文字型です。 UTF-16LE の 1 文字です。
WDPu16LE	リトルエンディアンの符号無し 16bit 値をあらわすデータです。
WDPu32LE	リトルエンディアンの符号無し 32bit 値をあらわすデータです。

3.2 WDPIconDataFormat

セッションのアイコン画像を表現するデータフォーマットです。
親機からの送信ビーコンで使用されます。

【定義】

```
typedef struct WDPIconDataFormat
{
    u16    palette[WDP_ICON_COLOR_MAX];
    u8     image[WDP_ICON_IMAGE_SIZE];
}
WDPIconDataFormat;
```

【説明】

メンバ	意味
palette	アイコン画像の 16 色パレットをあらわす配列です。 GXRgb フォーマットで、インデックス 0 は透明色として無視されます。
image	アイコン画像のピクセルイメージをあらわす配列です。 16 色 32×32dot の OBJ フォーマットです。

3.3 WDPPlayerDataFormat

セッションに参加する個々のプレイヤー情報を表現するデータフォーマットです。
親機からの送信ビーコンと子機からの WDP_COMMAND_REQ_ENTRY コマンドで使用されます。

【定義】

```
typedef struct WDPPlayerDataFormat
{
    #if (WDP_BITFIELDS_ENDIAN == WDP_ENDIAN_LITTLE)
        u8      color:4;
        u8      aid:4;
    #else
        u8      aid:4;
        u8      color:4;
    #endif
    u8      name;
    WDPChar name[WDP_PLAYER_NAME_MAX];
}
WDPPlayerDataFormat;
```

【説明】

メンバ	意味
color	本体設定『おきにいりの色』を想定した 0-15 の値です。 IPL ダウンロードプログラムはこの値を表示色に反映します。
aid	プレイヤーの接続局 AID をあらわす 0-15 の値です。
name_length	プレイヤー名 name の文字数です。終端は含みません。
name	プレイヤー名をあらわす文字列です。 name_length が WDP_PLAYER_NAME_MAX の場合、終端は付与しません。

3.4 WDPBeaconAttributeFormat

親機から送信する個々のビーコンの基本情報を表現するデータフォーマットです。
親機からの送信ビーコンで使用されます。

【定義】

```
typedef struct WDPBeaconAttributeFormat
{
    WDPu32LE id1;
#ifdef (WDP_BITFIELDS_ENDIAN == WDP_ENDIAN_LITTLE)
    u8      type : 2;
    u8      id2 : 6;
#else
    u8      id2 : 6;
    u8      type : 2;
#endif
    u8      serial;
    u8      volat_count;
    u8      index;
}
WDPBeaconAttributeFormat;
```

【説明】

メンバ	意味	
id1 id2 serial	ビーコンの分割元となったセッションを示す一意な ID です。(前述参照)	
type	ビーコンの種別を示す値です。以下のいずれかになります。 IPL が誤動作するため、これらの値以外を指定することはできません。	
	値	種別
	0	通常のセッションの固定情報ビーコンであることを示します。
	1	アイコン画像が無いセッションの固定情報ビーコンであることを示します。
	2	可変情報ビーコンであることを示します。
volat_count	可変情報ビーコンの内容が更新されるたびに 1 ずつ増加するカウンタ値です。 子機側がビーコンストリームの取りこぼしを検出するために使用されます。	
index	ビーコンストリーム全体のシーケンスを示すインデックス値です。 0 から 8 は固定情報ビーコンで、それ以降に可変情報ビーコンが続きます。 アイコン画像がない場合には 0 から 3 までは固定情報ビーコンになります。	

3.5 WDPSessionDataFixedFormat

セッション情報のうち変更されない固定情報を表現するデータフォーマットです。
親機からの固定情報ビーコン送信時に使用されます。

【定義】

```
typedef struct WDPSessionDataFixedFormat
{
    WDPIconDataFormat icon;
    WDPPlayerDataFormat parent;
    u8    player_max;
    u8    padding[1];
    WDPChar title[WDP_SESSION_TITLE_MAX];
    WDPChar comment[WDP_SESSION_COMMENT_MAX];
}
WDPSessionDataFixedFormat;
```

【説明】

メンバ	意味
icon	セッションのアイコン画像情報です。
parent	セッションのオーナーであるロビー親機のプレイヤー情報です。
player_max	このセッションが許可する最大の参加プレイヤー数を示す値です。
title	セッションのタイトル名をあらわす文字列です。 文字列は NUL で終端するかまたは WDP_SESSION_TITLE_MAX 文字すべて使用する必要があります。また、改行記号 L'¥n'を 1 個だけ含むことができます。ただし、各行の文字列は IPL のプロポーショナルフォントにおいて 185pixel 以内である必要があります。
comment	セッションの詳細説明文をあらわす文字列です。 文字列は NUL で終端するかまたは WDP_SESSION_COMMENT_MAX 文字すべて使用する必要があります。また、改行記号 L'¥n'を 1 個だけ含むことができます。ただし、各行の文字列は IPL のプロポーショナルフォントにおいて 199pixel 以内である必要があります。

3.6 WDPBeaconFormat

親機からの単体の送信ビーコンそのものを表現するデータフォーマットです。
親機からのビーコン送信時に使用されます。

【定義】

```
typedef struct WDPBeaconFormat
{
    WDPBeaconAttributeFormat attr;
    WDPu16LE checksum;
    union
    {
        u8      argument[102];
        struct
        {
            u8      index;
            u8      total;
            u8      length;
            u8      padding[1];
            u8      buffer[WDP_BEACON_FIXED_DATA_SIZE];
        } fixed;
        struct
        {
            u8      player_count;
            u8      padding[1];
            WDPu16LE player_bitmap;
            WDPu16LE update_bitmap;
            WDPPlayerDataFormat member[WDP_BEACON_VOLAT_PLAYER_MAX];
            u8      option[WDP_BEACON_OPTION_SIZE];
        } volat;
    } /* unnamed */ ;
}
WDPBeaconFormat;
```

【説明】

メンバ	意味
attr	分割ビーコンとしての属性情報です。
checksum	以降の後続データ 102 バイトのチェックサムをあらわす値です。 計算は一般にインターネットチェックサムと呼ばれる方法と同一です。
argument	後続データのサイズを表明するブレースホルダです。 fixed メンバおよび volat メンバとの共用体です。

fixed	attr メンバの type 値が 0 または 1 の場合のみ有効な情報です。 WDPSessionDataFixedFormat 構造体が 102 バイト単位で分割された個々の断片データを保持し、以下のメンバを持ちます。	
	メンバ	意味
	index	断片データのインデックス値です。 0 から total-1 の範囲を取ります。
	total	分割された断片の総数です。 この値は実際には attr メンバの type 値から一意に定まり、type 値が 0 の場合は 9 で 1 の場合は 4 となります。
	length	断片データのサイズです。 分割された終端の断片以外はすべて 102 バイトになります。
	buffer	断片データです。 total 個すべての断片データを index の順に連結すると WDPSessionDataFixedFormat 構造体のセッション情報になります。
volat	attr メンバの type 値が 2 の場合のみ有効な情報です。 動的に更新される参加プレイヤー情報を保持し、以下のメンバを持ちます。	
	メンバ	意味
	player_count	現在参加中のプレイヤー総数をあらわす 1-16 の値です。 親機自身は常にプレイヤーに含まれます。
	player_bitmap	現在参加中のプレイヤーをあらわすビットマップ値です。
	update_bitmap	前回の可変情報ストリームから今回の可変情報ストリームの間に更新されたプレイヤー情報の差分をあらわすビットマップ値です。
	member	現在参加中のプレイヤー情報 4 個分をあらわす配列です。 aid 値が 0 の要素の内容は無効です。
	option	ユーザ定義の任意のデータです。 親機側アプリケーション固有のルールで使用されます。

3.7 WDPEntryRequestFormat

セッション参加要求コマンドの情報を表現するデータフォーマットです。

子機からの WDP_COMMAND_REQ_ENTRY コマンドで使用されます。

【定義】

```
typedef struct WDPEntryRequestFormat
{
    WDPu32LE id1;
    WDPPlayerDataFormat player;
    WDPu16LE version;
    u8      id2;
    u8      padding[3];
}
WDPEntryRequestFormat;
```

【説明】

メンバ	意味
id1 id2	参加を要求するセッションを示す一意な ID です。(前述参照)
player	参加を要求する子機自身のプレイヤー情報です。
version	バージョン番号を想定した値です。 通常 1 を指定します。この値の運用に関する取り決めは不明です。

3.8 WDPSegmentTableFormat

個々のセグメント情報を表現するデータフォーマットです。
後述する WDPSegmentFormat 構造体のメンバとして使用されます。

【定義】

```
typedef struct WDPSegmentTableFormat
{
    WDPu32LE recv_addr;
    WDPu32LE load_addr;
    WDPu32LE length;
#ifdef (WDP_BITFIELDS_ENDIAN == WDP_ENDIAN_LITTLE)
    u8      target : 1;
    u8      reserved1 : 7;
#else
    u8      reserved1 : 7;
    u8      target : 1;
#endif
    u8      reserved2[3];
}
WDPSegmentTableFormat;
```

【説明】

メンバ	意味	
recv_addr	子機側がこのセグメントを一時的に退避すべき受信先メモリアドレスです。 IPLと同じメモリ配置で構成されるアプリケーションにのみ意味があります。	
load_addr	セグメントの配置アドレスです。	
length	セグメントのサイズです。	
target	セグメントの種別を示す値です。以下の値のいずれかになります。	
	値	種別
	0	ROM ヘッド情報かまたは ARM9 常駐セグメントを示します。
	1	ARM7 常駐セグメントを示します。
reserved1 reserved2	将来の拡張のための予約メンバです。 常に 0 を指定します。	

3.9 WDPSegmentFormat

ダウンロードプログラムのセグメント情報全体を表現するデータフォーマットです。
親機からの WDP_COMMAND_ACK_ENTRY コマンドで使用されます。

【定義】

```
typedef struct WDPSegmentFormat
{
    WDPu32LE entry_arm9;
    WDPu32LE entry_arm7;
    u8        reserved[4];
    WDPSegmentTableFormat table[WDP_SEGMENT_TYPE_MAX];
    u8        auth_code[WDP_AUTHCODE_SIZE];
    u8        user_param[WDP_USER_PARAM_SIZE];
}
WDPSegmentFormat;
```

【説明】

メンバ	意味	
entry_arm9	ARM9 プロセッサの起動アドレスです。 IPL は再起動時、ROM ヘッダ情報の内容ではなくこの値を採用します。	
entry_arm7	ARM7 プロセッサの起動アドレスです。 IPL は再起動時、ROM ヘッダ情報の内容ではなくこの値を採用します。	
table	ダウンロードプログラムに配置すべき 3 個のセグメント情報の配列です。 各要素の内容は以下のとおりです。	
	要素	種別
	0	ROM ヘッダ情報を示します。
	1	ARM9 常駐セグメントの情報を示します。
	2	ARM7 常駐セグメントの情報を示します。
auth_code	ダウンロードプログラムに付与する署名コードです。 署名コードは弊社の専用サーバから発行いたします。 詳細は弊社技術担当窓口までお問い合わせください。	
user_param	ダウンロードプログラムの再起動後へ持ち越されるユーザ定義データです。 アプリケーション固有のルールで使用されます。	
reserved	将来の拡張のための予約メンバです。 常に 0 を指定します。	

3.10 WDPPacketFormat

親子共通の MP 通信パケットそのものを表現するデータフォーマットです。
MP 通信で使用されます。

【定義】

```
typedef struct WDPPacketFormat
{
    u8      command;
    union
    {
        u8      argument[228];
        struct
        {
            u8      index;
            u8      buffer[6];
        } req_entry;
        struct
        {
            WDPSegmentFormat segment;
        } ack_entry;
        struct
        {
            WDPu16LE index;
            WDPu16LE count;
        } req_download;
        struct
        {
            WDPu16LE id2;
            WDPu16LE index;
            u8      buffer[1];
        } ack_download;
        u8      for_compiler[228];
    } /* unnamed */ ;
}
WDPPacketFormat;
```

【説明】

メンバ	意味
command	WDP_COMMAND_*で指定されるいずれかのコマンド ID です。
argument	後続データのサイズを表明するブレースホルダです。 req_entry, ack_entry, req_download, ack_download との共用体です。

req_entry	command 値が WDP_COMMAND_REQ_ENTRY の場合のみ有効な情報です。 WDPEntryRequestFormat 構造体が 6 バイト単位で分割された個々の断片データを保持し、以下のメンバを持ちます。	
	メンバ	意味
	index	断片データのインデックス値です。 0 から 4 の範囲を取ります。
	buffer	断片データです。 5 個すべての断片データを index の順に連結すると WDPEntryRequestFormat 構造体のセッション参加要求情報になります。
ack_entry	command 値が WDP_COMMAND_ACK_ENTRY の場合のみ有効な情報です。 以下のメンバを持ちます。	
	メンバ	意味
	segment	ダウンロードプログラムのセグメント情報です。
req_download	command 値が WDP_COMMAND_REQ_DOWNLOAD の場合のみ有効な情報です。 以下のメンバを持ちます。	
	メンバ	意味
	index	現在要求するセグメントパケットのインデックス値です。 親機側ではこの値を次回以降の応答処理の参考にしします。
	count	現在受信済みのセグメントパケット数です。 親機側ではこの値を子機側進捗状況の参考にすることができます。
ack_download	command 値が WDP_COMMAND_ACK_DOWNLOAD の場合のみ有効な情報です。 以下のメンバを持ちます。	
	メンバ	意味
	id2	セグメントパケットの分割元セッションを示す id2 の値です。 子機群はこのメンバをもとに自身あての応答パケットを判断するため、id2 が重複した複数のセッションを同一ロビー内で同時に生成することはできません。
	index	セグメントパケットのインデックス値です。
	buffer	セグメントパケットデータです。 セグメントパケットデータのサイズは動的に決定され、(親機最大送信サイズ-6)の値になります。

3.11 WDP パケットコマンド

WDPacketFormat 構造体で使用する command の値と後続引数の一覧を以下に示します。

コマンド	値	親	子	後続引数							
				0	1	2	3	4	5	6	...
WDP_COMMAND_MSG_IDLE	0	○	○								
WDP_COMMAND_MSG_HELLO	1	○									
WDP_COMMAND_MSG_REFUSE	2	○									
WDP_COMMAND_ACK_ENTRY	3	○		segment (WDPSegmentFormat)							
WDP_COMMAND_ACK_DOWNLOAD	4	○		id2		index			buffer[index][]		
WDP_COMMAND_REQ_REBOOT	5	○									
WDP_COMMAND_MSG_MAXENTRY	6	○									
WDP_COMMAND_REQ_ENTRY	7		○	index	buffer[index][6]						
WDP_COMMAND_MSG_STANDBY	8		○								
WDP_COMMAND_REQ_DOWNLOAD	9		○	index		count					
WDP_COMMAND_MSG_COMPLETE	10		○								
WDP_COMMAND_ACK_REBOOT	11		○								

図 3.11-1 WDP パケットコマンド一覧

4 プロトコル仕様および制限事項

WDP ライブラリにはいくつかの通信上の仕様、旧バージョンとの互換性による仕様、ならびにそれらから生じる使用上の制限が存在します。また、本体内蔵プログラムである IPL ダウンロードプログラム固有の実装に配慮した制限も存在します。本章では、これらの仕様および制限事項を記述します。

4.1 基本的な仕様

以下は、WDP ライブラリのバージョンや特定の实装に依存しない基本的な仕様です。

4.1.1 ダウンロードプログラムの配置可能領域

ダウンロードプログラムを構成する 3 個のセグメントに対する制限は、通常のカードブート用プログラムのものと全く同様で、以下の通りです。

実際のダウンロード処理にあたっては、各セグメントサイズを〔親機 MP 最大送信サイズ-6〕の単位で切り上げて連結したデータブロックを格納できる十分なメモリ領域を必要とします。

セグメント	最大サイズ	配置範囲	備考
ARM9 常駐	2.5 MB	[02000000, 022C0000)	compstatic 使用時、最大サイズは圧縮状態を対象。
ARM7 常駐	256 kB	[02000000, 02300000) [02300000, 023FE000) [037F8000, 0380F000)	compstatic 使用時、最大サイズは圧縮状態を対象。
ROM 内登録情報	352 B	[027FFE00, 027FFF60)	サイズ、配置先、ともに固定。

4.1.2 セッション情報の一意性の保証

ロビーを保持する親機側アプリケーションは、セッション情報に含まれる各種の ID について以下の条件を満たした値を正しく設定する必要があります。

- ・{ MAC アドレス、TGID、id1、id2 } の組でセッションの一意性を保証する必要があります。
具体的には、なるべく時間的に一意な値を TGID に採用し、id1 と id2 をセッション生成ごとに一意にする必要があります。そうでない場合、親機側の本体が WDP ライブラリやアプリケーション自体を再起動すると子機側アプリケーションで保持している古いセッション情報が新しいセッション情報と混同される症状が発生します。
- ・セッションの固定情報を変更する場合、serial を異なる値に更新する必要があります。
そうでない場合、セッションの固定情報の更新を子機側が認識せず再取得を実行しません。
※ただしこの用途に関しては IPL ダウンロードプログラム側で検証されておらず、
現在の WDP ライブラリにも対応する機能は実装されていません。

WDP ライブラリの上位層である MB ライブラリでは、各メンバを以下のように使用しています。

メンバ	値
id1	任意の 4 バイト値としてアプリケーションが自由に設定します。また、再起動後の子機がこの値を取得することができ、親機へ再接続するための GGID として用いられます。
id2	単に親機側で管理しているセッション情報配列のインデックスと同義です。
serial	プログラム起動直後を 0 として、セッションを新規生成するたびに 1 増加します。

このため、以下のような制約が生じます。

- MB_RegisterFile 関数と MB_UnregisterFile 関数を繰り返すと id2 が容易に重複することに加え、id1 の指定をアプリケーションに開放しているため、MB_UnregisterFile 関数を 1 回でも呼び出した後は MB_RegisterFile 関数の呼び出しを使用することができません。

4.1.3 ダウンロード要求コマンドの AID

子機から親機へ WDP_COMMAND_REQ_ENTRY コマンドで送信される WDPEntryRequestFormat 構造体の player.aid は実際の AID にかかわらず常に 0 です。WDP ライブラリでは親機側が受信時にこれを加工して AID を格納します。

4.2 IPL ダウンロードプログラムの制限

以下は、IPL ダウンロードプログラムの実装に起因する事項です。

4.2.1 子機ダウンロードプログラム再起動アドレス

IPL ダウンロードプログラムが再起動する際 ARM9 および ARM7 プロセッサの再起動アドレスは、親機からの WDP_COMMAND_ACK_ENTRY コマンドで受信した WDPSegmentFormat 構造体に含まれている entry_arm9、entry_arm7 の値を常に参照します。

なお、これらの値は実際には、親機から別途受信している ROM 内登録情報の値と常に同じです。

entry_arm9、entry_arm7 に対する正当性判定は行われていないため、悪意ある親機から改竄されることによって、IPL ダウンロードプログラムを介して任意のプログラムを実行させることが可能です。

具体例を以下の手順に示します。

- (1)書き込み可能な AGB FLASH カートリッジを用意して任意の DS プログラムを書き込み、前もって子機本体のカートリッジスロットに挿入しておく。
- (2)ARM7 起動アドレスをカートリッジ ROM 領域の先頭に指定する。
ARM9 起動アドレスを WDPSegmentFormat 構造体の user_param メンバ格納先(0x023FE8C4)に指定する。
- (3)WDPSegmentFormat 構造体の user_param メンバには、上記(1)で用意した ARM7 プログラムと同期を取るための 32 BYTE 以内のコードを設定しておく。
- (4)パケットキャプチャ等で取得した署名付きの正当なダウンロードプログラムを使用しつつ上記設定でダウンロードプレイ親機を実行する。

4.2.2 MP 通信における送受信サイズの制限

WDP を使用するアプリケーションの MP 通信には以下の制限が生じます。

- (1) 親機の最小送信サイズは 229 BYTE 以上である必要があります。

これは、最大のコマンドパケットである WDP_COMMAND_ACK_ENTRY のサイズが
 コマンド ID(1BYTE) + WDPSegmentFormat 構造体(228BYTE)であることに起因します。
 少なくともこのサイズは子機側でも受信可能である必要があります。

- (2) 親機が WMParentParam 構造体の childMaxSize に設定する送受信サイズは 8BYTE である必要があります。
 これは IPL ダウンロードプログラム公開時の MB ライブラリ不具合に起因します。

(詳細)

子機最大送信サイズは親機が設定する WMParentParam 構造体から取得され、IPL ダウンロードプログラムはこの最大送信サイズの範囲内でコマンドパケットを生成します。WDPEntireRequestFormat 構造体を複数のパケットに分割して送信する際、最大送信サイズを P、コマンドヘッダサイズを H とすると

$$N = (\text{sizeof}(\text{WDPEntireRequestFormat}) + ((P - H) - 1)) / (P - H)$$

として十分な分割数 N を求めることができますが、IPL ダウンロードプログラムでは

$$N = (\text{sizeof}(\text{WDPEntireRequestFormat})) / (P - H)$$

として計算しているため、サイズによっては終端の分割パケットが切り捨てられ含まなくなります。

実際には常に $\text{sizeof}(\text{WDPEntireRequestFormat}) = 30$ 、 $H = 2$ 、として扱われるため終端の分割パケットが含まれるか否かは下表のとおり送信サイズ P に依存します。

送信サイズ P	分割数 N	最大送信サイズ (P - H) * N	生じる問題
≤ 4			WDP_COMMAND_REQ_DOWNLOAD 送信不可
6	7	28	id2 欠落
8	5	30	(なし)
10	3	24	player.name,version,id2 欠落
12	3	30	(なし)
14	2	24	player.name,version,id2 欠落
16	2	28	id2 欠落
18	1	16	player.name,version,id2 欠落
20	1	18	player.name,version,id2 欠落
...			
28	1	26	player.name,version,id2 欠落
30	1	28	id2 欠落
32	1	30	(なし)
≥34	0	0	id1,player,version,id2 欠落

このよう

に、正しく分割可能なパケットサイズは 8、12、30 BYTE のみに限定されます。

またこの問題とは別に子機側では生成したパケット内容にかかわらず常に 8BYTE しか MP 送信していないという不具合も確認されており、これから親機は `childMaxSize` メンバに 8 BYTE 以外を指定することができなくなっています。

© 2006 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複写・転写・頒布・貸与することを禁じます。