AvaFit
Avatar solution + WebFit

システム解説書

AVAFIT Professional 4.40

2020.3.30

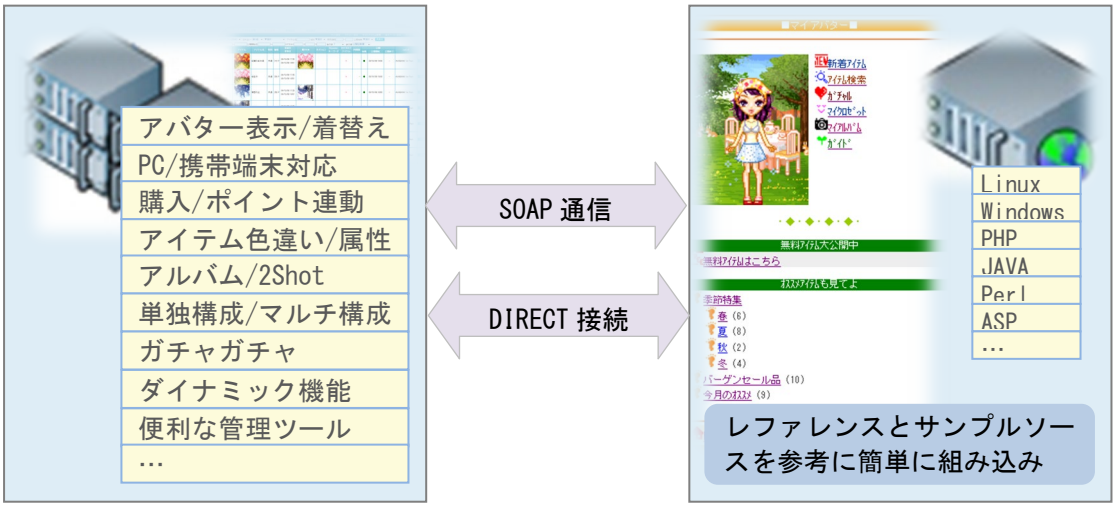
株式会社ウェブフィット

目次

Ⅰ　はじめに

ＡＶＡＦＩＴは、あらゆるコミュニティーサイトで強力なアバター機能を簡単に導入できるように作られたアバター専門システムです。サイトにアバター機能を導入するために必要な豊富な機能を搭載しており、ＡＰＩ関数を利用してユーザーインターフェースを作成することや管理機能を使って容易な管理ができるよう支援します。

アバターに関するすべての機能をＡＰＩ関数として提供することで、ユーザーインターフェース作成に制限がなく、自由な表現が可能になります。また、ユーザーインターフェース作成においては、サンプルソースコードが付属されているので、より簡単に作ることができます。



AVAFIT Professional　　お客様サイト

本システムは、単一サーバー構成／マルチサーバー構成を支援しています。サイトの規模に合わせて適切な構成が選べます。

本解説書をご覧いただいてもご不明な点がある場合は、下記までご連絡いただきますようお願いいたします。

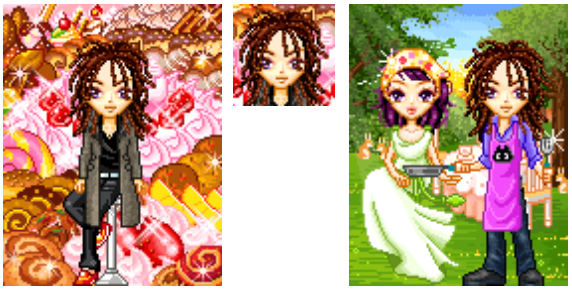
お問い合わせ先
support@webfit.co.jp

Ⅱ AVAFIT でできること

ＡＶＡＦＩＴはアバター機能に特化した専門システムです。本システムを利用することでお持ちのサイトにおいて下記のような機能がサービス可能になります。

１．豊富なアバター機能

- アバター作成
- アバターの全身表示・サムネール表示
- GIF アバターアニメーション生成機能
- アイテム一覧表示
- 豊富なアイテム検索機能
- 着替え処理
- アイテム購入処理／ポイント連動
- アバターアルバム
- ２ショット
- ＰＣ、携帯、スマートフォン対応
- 複数アイテムの同時着用・同時購入
- ランダムアイテム（ガチャガチャ）
- 親子関係設定（色違いアイテムなど）
- アイテム別属性設定・属性によるアイテム検索
- ダイナミック機能（条件によって異なる画像を使ってアバター合成）



＜アバター全身／サムネール／２ショット＞

＜天気情報と連動してダイナミック機能を使った一例＞

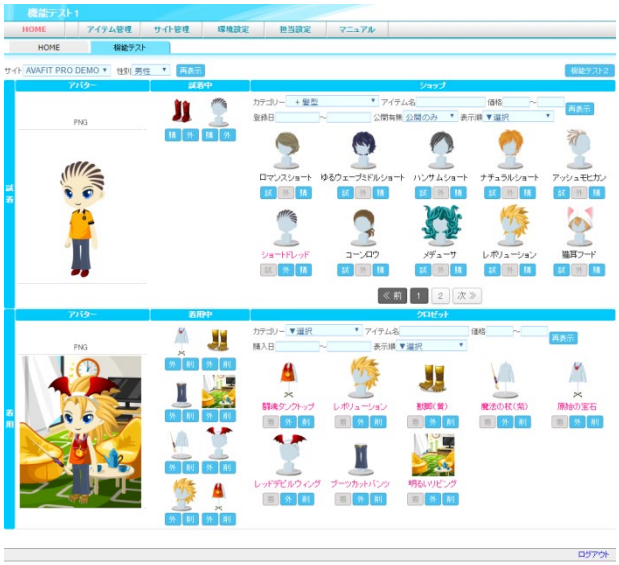
- GIF／PNG／JPEG 画像の自動生成
- 複数のサイト/複数のカテゴリー設定
- 男性／女性の区分だけではなく、複数のタイプを設定可能

２．強力なプラットフォーム

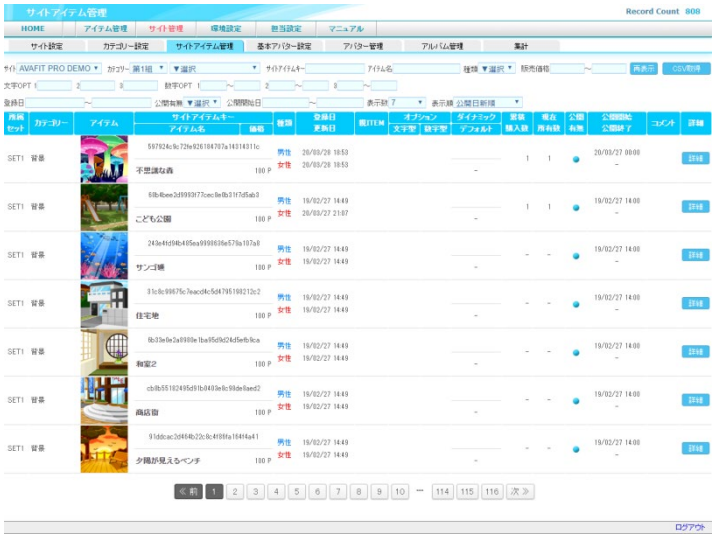
- 大規模に対応できる完全なマルチサーバー環境
- マルチ接続環境（SOAP／DIRECT）

３．進化した管理機能

- 見やすくて使いやすい管理インターフェース
- アイテム一括登録
- アイテム一括割り当て
- アイテムの一括カテゴリー移動
- 機能別アクセス・編集権限設定
- マニュアルのオンサイト
- 機能テスト
- アバター及びアイテム利用集計



＜機能テスト画面＞

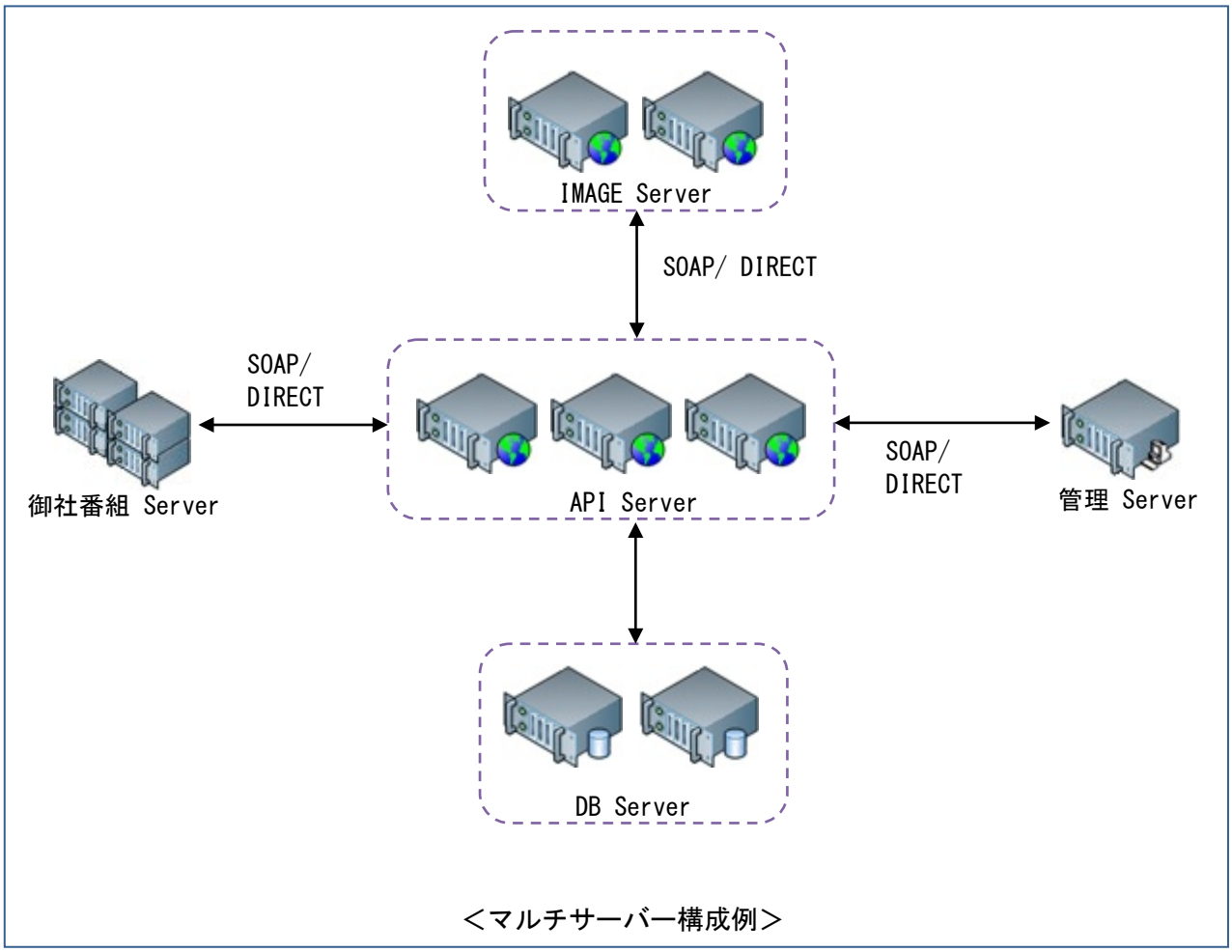


＜サイトアイテム一覧画面＞

Ⅲ システム構成

ＡＶＡＦＩＴは、アバターに関する豊富な機能をＡＰＩとして提供しており、お持ちのサイトのプラットフォーム／開発言語を問わず、すべてのサイトに導入することができます。また、ＡＰＩ接続においても複数の接続方式を支援しているので状況に合わせて接続方法を選ぶことができます。

小規模サイトから大規模サイトまですべてのサイトにおいてリーズナブルな環境を提供します。大規模サイトでは機能ごとに制限なくサーバーを増やすことができ、完璧なマルチサーバー環境をサポートすることで強力なアバター基盤を提供します。また、小規模サイトにおいては 1 台にすべての機能が収納でき、ＡＰＩ接続を使わないダイレクト接続を行うことで、無駄のない軽い基盤を提供します。



＜マルチサーバー構成例＞

IMAGE サーバー／DB サーバー／管理サーバー／UI 搭載サーバーのすべてのサーバー群は API サーバーだけに接続する独立基盤設計になっており、優れた拡張性を持ちます。例えば、管理サーバーを IMAGE サーバーと統合することも、管理サーバーをマルチ化することもできます。また、既にサービスを稼働している場合でもサービスを止めることなく API サーバーを増設したり、IMAGE サーバーを増設したり、管理サーバーを増設したりすることができます。

１．API サーバー

大規模環境において複数の API サーバーを使う場合、すべての API サーバーは同じ処理を担当し、同じ内容の静的プログラムが搭載されます。そのために、万が一 API サーバーの１台が壊れた場合は、API サーバーを参照する管理システム及び UI の設定ファイルから対象の API サーバーを無くすだけで障害を解消できます。また、サービス中においても、サービスを停止することなく増設することも可能です。API サーバーは下記のような特徴を持ちます。

- AVAFIT システムにおいてのエンジン役割
- 他のサーバー群はすべて API サーバーのみを経由するので拡張性が優れている。
- 複数の API サーバーを導入したシステムにおいて、UI から API サーバーを参照する際に、API サーバーが同じドメインである必要はないので L4 Switch などの装置を使わなくてもマルチ構成が可能
- API サーバーの台数には制限がない

２．画像サーバー

画像サーバーはアイテム画像ファイルを保管し、アバター及びアルバムなどの生成、発信を担当するサーバーです。管理画面よりアバターのアイテムを登録すると、管理サーバーはアイテムの画像ファイルを画像サーバーに送ります。画像サーバーを複数導入したシステムであれば、アイテムを登録する度に自動的に全画像サーバーにアイテムを送ることになります。そのため、複数の画像フーバーがある場合は、１台が壊れても他の画像サーバー上にあるアイテム画像ファイルをコピーすることで復旧が可能になります。

反面、アイテムの合成で作られたアバター画像及びアルバム画像は、アイテムそのものとは異なり、ユーザー毎に配置される画像サーバーは一つだけになります。アバター作成・

着替えなどの処理を行う際に、複数の画像サーバーで同時に同じ処理を行うのは負荷がかかるためです。もし、画像サーバーが壊れた場合、アバター画像及びアルバム画像は完全に無くなってしまうのではといった心配をなさるかもしれませんが、アバター画像及びアルバム画像は、元のアイテム画像さえあれば再生することができるので心配は要りません。API 関数の中にはアバターを再作成する関数が用意されていますのでユーザー分実行するだけでアバターが再作成されます。

- アイテム保管（全画像サーバー共通）
- アバター及びアルバムの作成・保管（ユーザー毎に一つの画像サーバーに保存）
- サービスを停止せずに増設可能
- アバターまたはアルバム画像がなくなった場合は、再作成可能
- 複数の画像サーバーのどのサーバーにアバターを作るかは自動的に決まる。ただ、指定することも可能
- 複数の画像サーバーがある場合、同一ドメインを使う必要がないので L4 Switch などの装置を使わなくてもマルチ構成が可能
- 画像サーバーの台数には制限がない

３．データベースサーバー

本システムは、MySQL データベースを利用しています。複数のデータベースを利用する場合は、マスタ・スレーブといったレプリケーション構造を利用することになります。データベースにデータが追加される際には、まずマスタサーバーに情報が書き込まれ、ほぼリアルタイムでその情報がスレーブサーバーにコピーされる仕組みです。マスタサーバーは新規データの追加や変更の処理を担当し、スレーブサーバーはその内容の参照を担当することになります。

- MySQL データベースを利用し、画像以外の全てのデータを保存・管理するサーバー
- マルチ構成の場合は、レプリケーション構造を利用する
- 複数のスレーブサーバーを利用する場合、各スレーブサーバーの処理割合を指定することができる
- マスタサーバーもスレーブの役割を担当させることができる
- データベースの増設・故障などの時は、サービスを停止する必要がある

４．管理サーバー

管理サーバーは API サーバーのみに接続します。データベースサーバーや画像サーバーに対する情報を保管する必要がなく、すべての処理を API 経由で行いますので、UI と同様に設置に制限がありません。

アバターに関する情報以外の管理ツール独自の情報（担当者情報・権限情報など）は独自のデータベースで管理しています。そのデータベースは、アバター用データベースと統合することも独自に持たせることも可能になります。

- API サーバーだけ接続するので設置に制限がない（UI と同様）
- 管理サーバーが壊れてもサービスには影響を与えない

Ⅳ サービス準備の手順

お持ちのサイトでアバター機能をサービスするためには、いくつかの準備が必要になります。ここではＡＶＡＦＩＴを使ってユーザーにアバターサービスを提供するまでの流れについて説明しますので、その手順に従って準備してください。

１．アイテム準備

アバターの素体及びアイテムは、新たに作成する方法と既存パッケージを導入する方法があります。新規で作成する場合は、お客様のオリジナル性を確保することができますが、アバターの設計（レイヤー構成・種類・テイスト・サイズなど）が必要になり、アバターの作成にも時間を要します。反面、既存パッケージを導入する場合は、安価で簡単に導入できるメリットがあります。

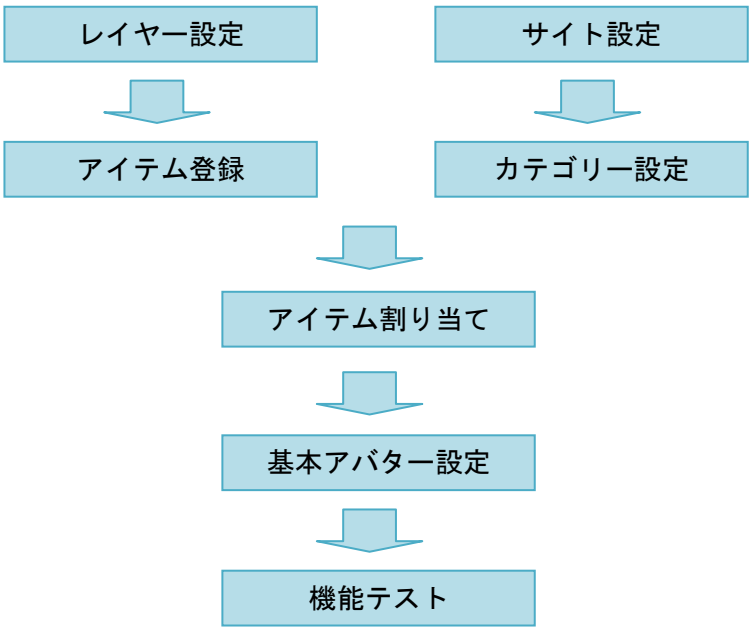
ＷＥＢＦＩＴでは、アバター素体及びアイテムの新規作成及び既存パッケージ導入についてもサポートしておりますので、お気軽にお問い合わせください。

２．管理ツールでの準備

管理ツールの使い方に関する説明は管理ツール内に付属しています。ここでは管理ツールを使っての準備手順について説明します。

準備する必要のある作業としては、レイヤー設定、アイテム登録、サイト設定、カテゴリー設定、アイテム割り当て、基本アバター設定などの項目がありますが、各項目の作業には順番があります。例えば、アイテムを登録するためには、そのアイテムをどのレイヤーに位置させるかを決めないといけませんので、アイテムの登録よりレイヤー設定を先に行う必要があります。また、カテゴリーを設定するのもどのサイトのカテゴリーなのかが決まらないといけませんので、カテゴリーの登録をする前にサイトを登録する必要があります。

＜管理ツールでの準備手順＞



管理ツールのログインした直後の画面には現在各サイトの準備状況が表示されます。ユーザーにアバター機能をサービスするためには、すべての項目が「●」になっている必要があります。

SITE_KEY	サイト名	利用有無	登録/更新	準備状況			コメント	移動	編集
				カテゴリー登録	アイテム登録	基本アバター設定			
46c5003268baeee8393e504f6f5d2599	AVAFIT PRO DEMO	●	13/09/25 17:14 20/03/27 21:11	●	●	●		↓	編集
d0789cfca42737fa2182da6058836ba2	まるまる	●	20/03/30 17:01 -	●	●	●		↑	編集

上記例では「AVAFIT Pro DEMO」サイトはサービス可能な状況を示しています。また、「まるまる」サイトはカテゴリー、アイテム登録、基本アバター設定などが進んでいないのでサービスすることはできません。

３．UI の作成

アイテムの準備及び管理ツールでの設定が終わったらユーザーインターフェースの作成を行います。本システム解説書の「UI 作成」、「API レファレンス」を参考にユーザーインターフェース画面を作ることができますが、本システムに付属しているサンプルソースコード（PHP 版）を活用することも可能です。UI 作成の詳細については、「V UI 作成」をご覧ください。

Ⅴ UI 作成

アバター機能付きのユーザーインターフェース（UI）を作るためには、本システムのＡＰＩ関数の使い方を把握していただく必要があります。下記は PHP を使ったＡＰＩ関数利用の簡単な例になります。

```
1    $option = array('encoding' => 'UTF-8');
2    $AVAFIT = new SoapClient("http://hogehote.jp/api/avafit.wsdl", $option);
3
4    $return = $AVAFIT->avatar_call($site_key, $user_code);
5    if (!$return["result"]) {
6        $ERROR[] = $return["error_shot"];
7    }
8    else {
9        $avatar = $return["value"];
10       $nickname  = $avatar["NICKNAME"];
11       $sex       = $avatar["SEX"];
12       $image_url = $avatar["URL"];
13       …
14   }
```

上記の利用例では、

① 選んだ接続方法に合わせてＡＰＩサーバーに接続（１～２行）
② ＡＰＩ関数の呼び出し（４行）
③ ＡＰＩ関数から戻された結果の判断（５行）
④ ＡＰＩ関数より得た情報の利用（９行～）

のような手順で利用しています。簡単な例ではありますが、すべてのＡＰＩ関数利用において上記の形式で利用することになります。各項目において下記に詳しく説明します。

本マニュアルでは PHP の利用例を説明します。その他の開発言語においても同様な使い方でＡＰＩを利用することができます。

１．接続方法

ＡＰＩサーバーへの接続手段として３つの方法が用意されています。適切な接続環境を選んでご利用してください。

１）ダイレクト接続（Library Include）

UI を含むアバターに関するすべての機能を１台サーバーに収まる場合に使える接続手段です。接続というよりもライブラリーをそのままインクルードすることなので、他のライブラリーの利用と変わりはありません。

```
include_once ("/home/avafit/lib/avatar.lib");
$AVAFIT = new WFavatar();
```

２）SOAP 通信による接続

UI を含め、２台以上のサーバー構成では上記のようなインクルード形式は使えません。そのために別途用意した通信手段を利用してＡＶＡＦＩＴの API サーバーに接続する必要がありますが、その一つとして最も一般的な SOAP 通信があります。

```
$option = array('encoding' => 'UTF-8');
$AVAFIT = new SoapClient("http://hogehote.jp/api/avafit.wsdl", $option);
```

SOAP 通信を利用するためには、呼び出す側のサーバーに SOAP ライブラリーを設置する必要があります。PHP、JAVA、ASP、Perl などほとんどの開発言語では SOAP を支援していますので UI を作成する開発言語に合わせて SOAP ライブラリーを設置してください。

※SOAP 通信を利用してＡＶＡＦＩＴの API サーバーに接続を行う場合、API 関数を呼び出したユーザーインターフェース（UI）搭載サーバー上に「avafit.wsdl」のキャッシュが残るようになります。ＡＶＡＦＩＴのバージョンアップなどで「avafit.wsdl」が変更された場合は、UI 搭載サーバー上のキャッシュを削除する必要があります。キャッシュを削除しない場合は、正しく機能しない場合があります。

２．ＡＰＩ関数の呼び出し

「１．接続方法」で説明した方法に従ってＡＰＩサーバーに接続してからは、利用に応じて各ＡＰＩ関数を呼び出します。

```
$return = $AVAFIT->avatar_call($site_key, $user_code);
```

すべてのＡＰＩ関数は、上記のような形式で呼び出すことになります。どのＡＰＩ関数がどのような役割を担当するか、また呼び出したＡＰＩ関数から得られる情報には何があるかなど、ＡＰＩの詳しい情報については「VI API レファレンス」をご確認ください。

３．ＡＰＩ関数から戻された結果の判断

ＡＰＩ関数を呼び出した場合は、まずその結果が正しいかどうかを確認する必要があります。パラメーターの入力ミス、サイトコード（$site_key）の未発行など、様々な理由で正しい結果が得られない場合もありますので、ＡＰＩ関数を呼び出した場合は必ずその結果を確認します。

```
$return = $AVAFIT->avatar_call($site_key, $user_code);
```

上記のようにＡＰＩ関数を呼び出した場合、「$return」にセットされる値は下記のような構造になります。すべてのＡＰＩ関数で共通です。

$return["result"]	成功の場合「true」、失敗の場合「false」
$return["value"]	成功の場合、それぞれの情報（関数ごとに異なります。）
$return["error"]	失敗の場合、エラーの詳細内容
$return["error_shot"]	失敗の場合、エラーの簡略な内容
$return["error_no"]	失敗の場合、エラー番号

すべてのＡＰＩ関数において上記のような形式で戻り値が得られますので、ＡＰＩ関数を呼び出した場合は下記のように結果を確認します。

```
$return = $AVAFIT->avatar_call($site_key, $user_code);
if (!$return["result"]) {
```

```
    //API 関数の呼び出し結果が「false」の場合はエラー処理
    $ERROR[] = $return["error_shot"];
}
else {
   //API 関数の呼び出し結果が「true」の場合は、その値の利用
   …
}
```


４．ＡＰＩ関数より得た情報の利用

どのＡＰＩ関数かによって得られる情報は異なります。関数によっては単独の値がもらえる場合もありますが、ほとんどの関数は情報を配列として渡します。その内容は、関数毎にことなり、渡す情報については「VI API レファレンス」で詳しく説明します。

```
4    $return = $AVAFIT->avatar_call($site_key, $user_code);
5    if (!$return["result"]) {
6        $ERROR[] = $return["error_shot"];
7    }
8    else {
9        $avatar = $return["value"];
10       $nickname  = $avatar["NICKNAME"];
11       $sex       = $avatar["SEX"];
12       $image_url = $avatar["URL"];
13       …
14   }
```

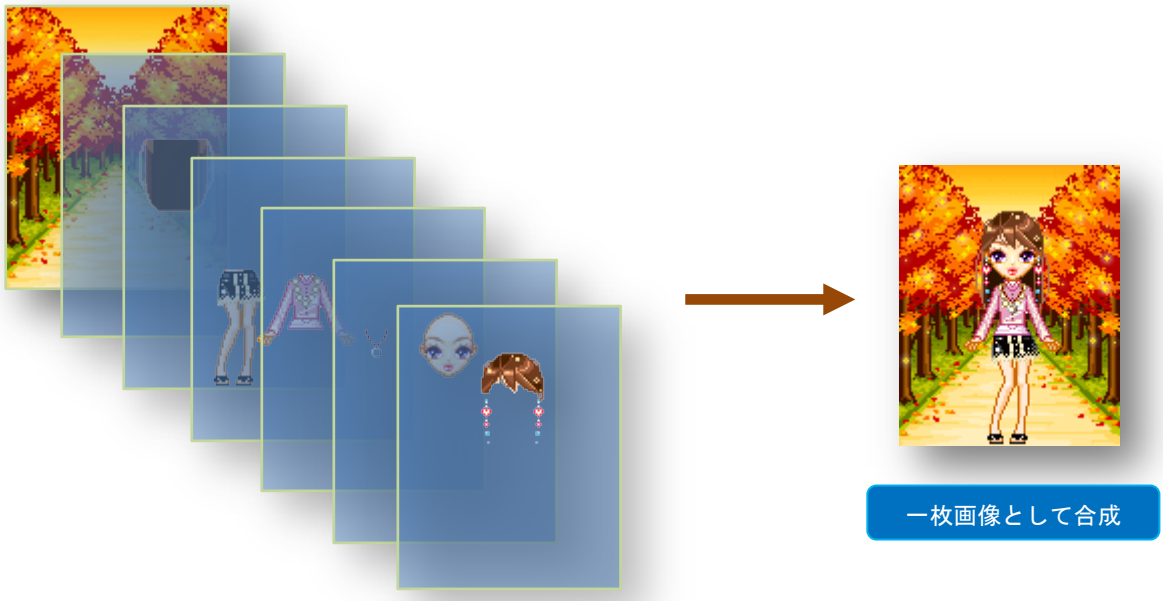
「avatar_call」といったＡＰＩ関数は **$return["value"]** として下記のような情報を渡します。

$avatar["KEY"]	アバターの判別キー
$avatar["USER_CODE"]	アバターを作成する際に渡したユーザーコード
$avatar["NICKNAME"]	アバターを作成する際に渡したニックネーム
$avatar["SEX"]	アバター種類（例）男性「m」、女性「f」、犬「d」）
$avatar["REGIST_DATE"]	登録日付（2009/10/09 20:35:36）

$avatar["UPDATE_DATE"]	着替えなどで更新された日付
$avatar["URL"]	アバター画像の URL（拡張子はついていない）
$avatar["TAGSET"]	PC でアバター表示をする場合の HTML タグ
$avatar["COMMENT"]	管理ツールより記入したコメント
$avatar["ITEM_TOTAL_COUNT"]	着用中アイテム数（Default アイテムは除く）
$avatar["ITEMS"] (Item Array)	着用中アイテムの詳細（配列）

上記例で、**$avatar["ITEMS"]** は着用中のアイテムに関する情報を配列構造で渡すことに注意してください。

１）携帯端末での表示




```
$return = $AVAFIT->avatar_call($site_key, $user_key);
if ($return["result"]) {
    $avatar = $return["value"];
    //アバター画像の表示
    <img src="<?=$avatar["URL"]?>.gif?<?=uniqid("_")?>">
}
```

携帯端末のキャッシュが効かないようにするため付ける

http://xxx.com/data/avatar/afc/67a/2b8/acd...238.gif

２）PC 端末での表示

```
$return = $AVAFIT->avatar_call($site_key, $user_key);
if ($return["result"]) {
    $avatar = $return["value"];
    //アバター画像の表示
    <?=$avatar["TAGSET"]?>
}
```

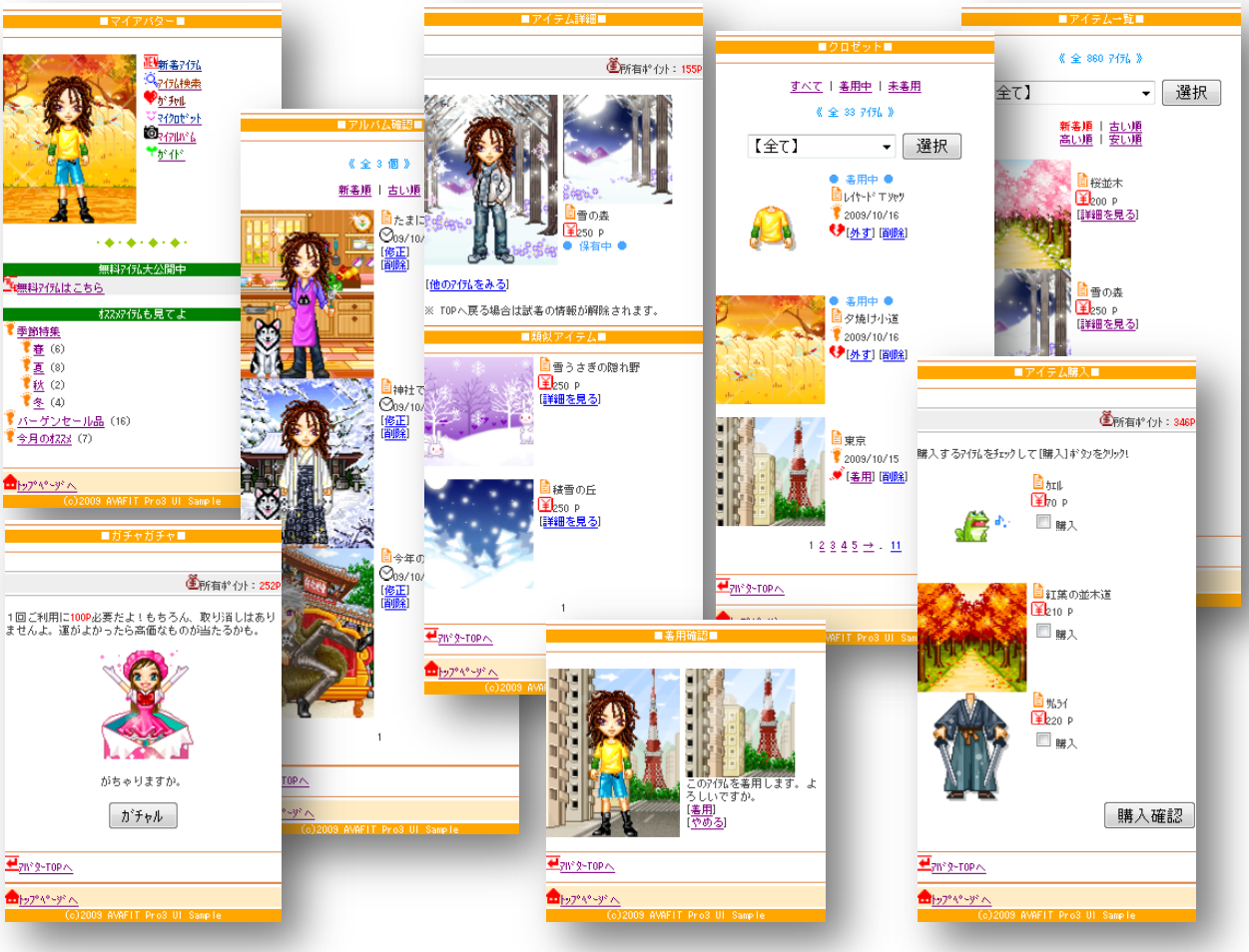
PC ではアバターではなくアイテムを重ね合わせて表示している。アイテムは随時変わるものではないのでキャッシュ対策は不要

```
<DIV id='avatar' style='LEFT: 0px; VISIBILITY: visible; WIDTH: 110px; POSITION: relative; HEIGHT: 140px'>
  <IMG style='LEFT: 0px; TOP: 0px POSITION: absolute;' src='http://xxx.com/items/62/284...865.gif' border='0'>
  <IMG style='LEFT: 0px; TOP: 0px POSITION: absolute;' src='http://xxx.com/items/1a/fe5...925.gif' border='0'>
  <IMG style='LEFT: 0px; TOP: 0px POSITION: absolute;' src='http://xxx.com/items/10/33c...901.gif' border='0'>
  <IMG style='LEFT: 0px; TOP: 0px POSITION: absolute;' src='http://xxx.com/items/76/793...d74.gif' border='0'>
  <IMG style='LEFT: 0px; TOP: 0px POSITION: absolute;' src='http://xxx.com/items/8e/bef...753.gif' border='0'>
  <IMG style='LEFT: 0px; TOP: 0px POSITION: absolute;' src='http://xxx.com/items/b7/f9b...82a.gif' border='0'>
  <IMG style='LEFT: 0px; TOP: 0px POSITION: absolute;' src='http://xxx.com/items/7f/c4f...fae.gif' border='0'>
</DIV>
```

PC でアバターを表示する際には、「avatar_call」などで返された値から「TAGSET」を「URL」の代わりに利用します。「TAGSET」には各アイテムの URL 及びそれを囲む HTML タグが含まれております。（もちろん、「TAGSET」を利用せずに「URL」を利用することも可能です。）

５．サンプルソース

ＡＶＡＦＩＴにはサンプルソースコード（PHP 版）が付属されています。一からコーディングするのが煩わしいと思ったらサンプルソースコードを活用すべきです。サンプルではありますが、ほとんどの機能を網羅していますので是非ご活用してください。



また、サンプルソースコードは更新される場合がありますのでＷＥＢＦＩＴホームページより最新の情報をご確認ください。

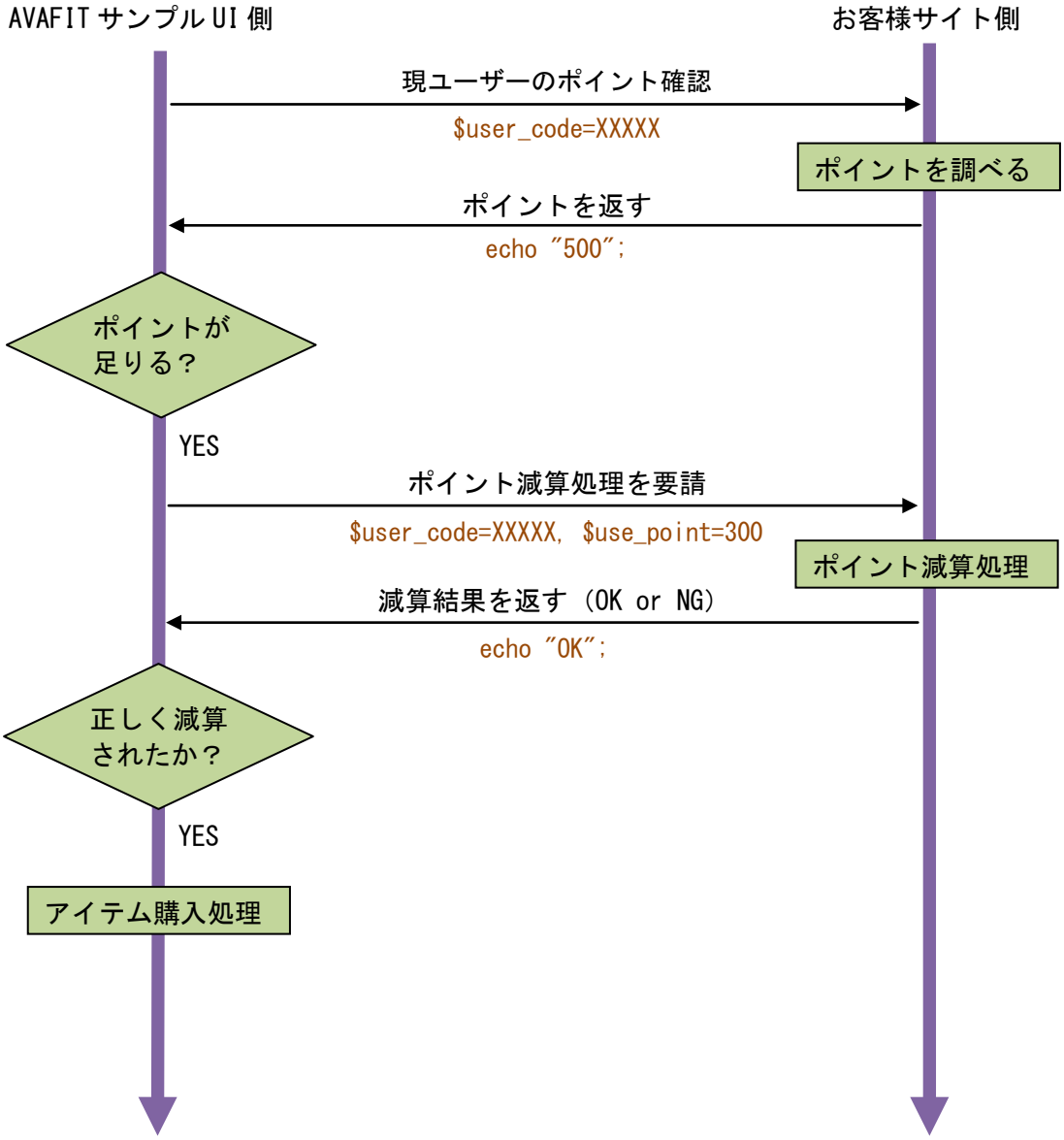
１）ポイント連動処理

アバターを導入するサイトの中にはポイント制度を実施しているサイトも多いと思います。本サンプルソースはポイント処理に対応していますのでご活用してください。

アバター処理側（AVAFIT サンプル UI）とサイト側（お客様のお持ちのサイト）でポイントを連動するというのは、アイテムを購入するタイミングにおいて、アバター処理側から

サイト側に現在のユーザーがどれほどのポイントを所持しているのか問い合わせを行います。また、アイテムを購入した場合は、購入に使った利用ポイントがいくらなのかをサイト側に報告し、サイト側で減算処理を行う必要があります。

＜ポイント連動処理の流れ＞



サンプルソースコードの中には、「point_get.php」、「point_use.php」というファイルが含まれています。その 2 つのファイルは、ポイントの確認（point_get.php）、ポイント減算処理（point_use.php）を行うファイルで、実際にはサイト側に置いておく必要のあるフ

ファイルですが、例としてサンプルソースコードの中に含めました。お持ちのサイト上で、「point_get.php」、「point_use.php」と同様の機能を実装していただき、サンプルソースの中にある「config.ini」ファイルの_WU_POINT_GET_URL、_WU_POINT_USE_URL にそのファイル URL を指定すれば、ポイント処理が連動されます。

「point_get.php」、「point_use.php」ファイルの詳細については、そのファイルの中身をご確認してください。

２）サンプルソースの設定ファイル

位置 ⇒ UI SAMPLE Source/config/config.ini

● _WU_RETURN_URL

サイトへ戻るリンクをクリックした際の戻り先 URL を記入してください。

● _WU_SITE_NAME

コピーライト表示用としてサイト名を指定します。

● _WU_POINT_GET_URL

現在ユーザーのポイントが確認できるサイト側の URL を記入してください。

● _WU_POINT_USE_URL

アイテムを購入する際に、ポイント減算処理を行うサイト側の URL を記入してください。

● _WU_AVAFIT_SITE_KEY

ＡＶＡＦＩＴのご利用サイトキー（32Byte）を記入してください。

● _WU_ITEM_CNT_LIST_PAGE

アイテムのリストを表示する際に１ページに表示するアイテム数を記入します。携帯端末においては、多く表示できない場合がありますので少なめに設定してください。

● _WU_GACHA_POINT

ガチャガチャ機能で減算するポイントを記入してください。

Ⅵ API レファレンス

１．戻り値

すべてのＡＰＩ関数は下記の共通的な戻り値構造を持っています。また、呼び出された各関数はその用途によって $return["value"] にそれぞれ異なる情報を渡します。

$return["result"]	成功の場合「true」、失敗の場合「false」
$return["value"]	成功の場合、それぞれの情報（関数ごとに異なります。）
$return["error"]	失敗の場合、エラーの詳細内容
$return["error_shot"]	失敗の場合、エラーの簡略な内容
$return["error_no"]	失敗の場合、エラー番号

実際、$return["value"] で渡される情報はある程度定型化されています。ここでは、$return["value"] で渡される値の中で定型化されている構造を説明します。どの関数がどのような定型パターンを返してもらえるかについてはＡＰＩ関数の詳細をご覧ください。

$return["value"]の値が Item Array の配列 のように「… の配列」と表記されている場合がありますが、その場合は Item Array がさらに配列化されたことを意味します。

1）Avatar Array

「avatar_create」「avatar_call」などアバターに関する情報取得や操作を行う際に返される配列です。

値	内容
$avatar["KEY"]	アバターの判別キー
$avatar["USER_CODE"]	アバターを作成する際に渡したユーザーコード
$avatar["NICKNAME"]	アバターを作成する際に渡したニックネーム
$avatar["SEX"]	アバター種類の文字
$avatar["REGIST_DATE"]	登録日付（2009/10/09 20:35:36）
$avatar["UPDATE_DATE"]	着替えなどで更新された日付
$avatar["URL"]	アバター画像の URL（拡張子はついていない）
$avatar["TAGSET"]	PC でアバター表示をする場合の HTML タグ
$avatar["USE_POINT"]	アイテム購入に使ったポイント合計
$avatar["LAST_DYNAMIC_CONDITION"]	最後の着替えで使ったダイナミック条件
$avatar["COMMENT"]	管理ツールより記入したコメント
$avatar["ITEM_TOTAL_COUNT"]	着用中アイテム数（Default アイテムは除く）
$avatar["ITEMS"]	着用中アイテムの詳細（ Item Array の配列 ）

２）Category Array

アイテムのカテゴリー情報を取得する関数で返される配列です。

値	内容
$category["INDEX"]	カテゴリーのインデックス番号
$category["NAME"]	カテゴリー名
$category["DEPTH"]	カテゴリーの階層
$category["ITEM_COUNT"]	対象カテゴリーに含まれるアイテム数（対象カテゴリーのサブカテゴリーに属するアイテムや親子関係で子になっているアイテムは含まない）

３）Item Array

アイテム情報を取得する際に返される配列です。

値	内容
$item["KEY"]	サイトアイテムキー
$item["NAME"]	アイテム名
$item["SEX"]	アイテム利用のアバター種類文字
$item["COMMENT"]	管理ツールより記入したコメント
$item["PRICE"]	販売価格
$item["REGIST_DATE"]	システムへの登録日付
$item["UPDATE_DATE"]	内容が更新された日付
$item["SELL_START_DATE"]	販売開始日付
$item["SELL_END_DATE"]	販売終了日付
$item["URL"]	アイテム画像[トルソー]ファイルの URL (拡張子含む)
$item["URL2"]	アイテム画像[トルソー]ファイル・2 枚目の URL (拡張子含む)
$item["CATEGORY_KEY"]	カテゴリーインデックス番号
$item["CATEGORY_NAME"]	カテゴリー名
$item["IS_CLOSET"]	対象ユーザーの所有有無（所有 true、未所有 false） ユーザーコードを渡す関数のみセットされます。
$item["IS_PUTON"]	着用状態（着用中 true、未着用中 false）※
$item["PARENT_KEY"]	親サイトアイテムキー（親アイテムがある場合のみ）
$item["CHILD_COUNT"]	子アイテム数
$item["OPTION1"]	文字型オプション１の値
～	～
$item["OPTIONXX"]	文字型オプション XX の値（XX は設定による）
$item["VALUE1"]	数字型オプション１の値
～	～
$item["VALUEXX"]	数字型オプション XX の値（XX は設定による）
$item["DYNAMIC"]	ダイナミックキーワード

※クロゼットに同一アイテムが複数あって、そのアイテムのうち１つを着用している場合は、「IS_PUTON」は「true」を返します。

４）Closet Array

特定のアバターのクロゼット情報を返却します。

値	内容
$closet["KEY"]	サイトアイテムキー
$closet["CLOSET_ITEM_KEY"]	クロゼットアイテムキー※
$closet["NAME"]	サイトアイテム名
$closet["REGIST_DATE"]	購入日付
$closet["POINT"]	購入に使ったポイント
$closet["IS_PUTON"]	着用状態（着用中 true、未着用中 false）
$closet["URL"]	アイテム画像[トルソー]ファイルの URL（拡張子含む）
$closet["URL2"]	アイテム画像[トルソー]ファイル・2 枚目の URL（拡張子含む）
$closet["CATEGORY_KEY"]	対象アイテムのカテゴリーインデックス番号
$closet["CATEGORY_NAME"]	対象アイテムのカテゴリー名

※「CLOSET_ITEM_KEY」は、一人のアバターが同一アイテムを複数保有できる仕様（avatar_item_add_duplicate を利用してアイテムをクロゼットに追加する場合）で使われます。一人のアバターのクロゼットに同じアイテムが複数存在する場合、複数の同じアイテムはすべて同じ「サイトアイテムキー（KEY）」を持つため、そのキーだけではそのうちの一つを判別することができません。それに対して、「クロゼットアイテムキー（CLOSET_ITEM_KEY）」は、同じアイテムであってもすべて異なるキーになりますので、複数の同一アイテムから一つを特定することができます。

クロゼットのアイテムを着用する動作に使われる「avatar_item_puton」、「avatar_item_putoff」、クロゼットアイテムを削除する「avatar_item_delete」などの関数では「サイトアイテムキー」をパラメーターに渡しますが、その代わりに「クロゼットアイテムキー（CLOSET_ITEM_KEY）」を渡すこともできます。「avatar_item_delete」の場合、パラメーターに「サイトアイテムキー（KEY）」を渡せば、そのキーを持つ複数の同一アイテム全てが削除されますが、「クロゼットアイテムキー（CLOSET_ITEM_KEY）」を渡せば、そのキーを持つ一つのアイテムのみが削除されます。

また、「avatar_item_puton」を使う際の例として、昨日「A」というアイテムを購入して今日も「A」というアイテムを購入して同じ「A」を２つ持っている場合、昨日買った「A」を着用させたい場合は「クロゼットアイテムキー（CLOSET_ITEM_KEY）」をパラメーターに渡すことで対応が可能です。

５）Album Array

アルバム情報を返却します。

値	内容
$album["KEY"]	アルバムキー
$album["USER_CODE"]	所有ユーザーのユーザーコード
$album["NICKNAME"]	所有ユーザーのニックネーム
$album["URL"]	写真 URL（拡張子はついていない）
$album["TAGSET"]	PC でアルバム表示をする場合の HTML タグ
$album["TEXT"]	アルバム登録の際に入力したテキスト
$album["REGIST_DATE"]	アルバム登録日付
$album["UPDATE_DATE"]	アルバム修正日付

６）Try Array

試着情報及び試着に使われたアイテム情報を返します。

値	内容
$try["TRYKEY"]	試着状態維持のための KEY
$try["URL"]	試着状態のアバター画像 URL
$try["TAGSET"]	PC でアバター表示をする場合の HTML タグ
$try["LAST_DYNAMIC_CONDITION"]	最後の着替えで使ったダイナミックキーワード条件
$try["ITEM_TOTAL_COUNT"]	試着中アイテム数（Default アイテムは除く）
$try["ITEMS"]	試着中アイテムの詳細（ Item Array の配列 ）

２．検索条件の指定方法

AVAFIT の検索型関数（avatar_get、category_info、category_get, category_item_get, category_closet_get など）は条件指定による絞り込みができます。各関数の指定可能な検索条件はそれぞれ異なりますが、指定方法は統一されています。

API 関数の検索パラメーターに複数の条件を同時に指定する場合は、下記のルールを参考に指定してください。

[条件指定方法]

[条件名]:[条件]

API 関数毎に決まった条件を記入

コロン（:）は必須

条件が文字列系であればあいまい検索（%）が可能、比較系の場合は >, >=, <, <=が指定可能
例）name:%空%、selldate:>= 2011/3/12

[複数条件の指定方法]

条件 1 , 条件 2 | 条件 3 | 条件 4 , 条件 5

コンマ（,）で条件を繋ぐとアンド（AND）条件になる。

「|」記号で条件を繋ぐと OR 条件になる。

(条件 1) AND (条件 2 OR 条件 3 OR 条件 4) AND (条件 5)

例）

```
$condition = "name:あ% | name:%い | option1: う, selldate: >= 2011/3/1"
```

上記のように指定した場合は、下記のように解析されます。

((名前 LIKE "あ%") OR (名前 LIKE "%い") OR (オプション 1 = "う")) AND (販売日付 >= 2011/3/1)

３．API 関数一覧

関数名	機能
avatar_create	アバター作成
avatar_delete	アバター削除
avatar_call	アバター情報を呼び出す
avatar_call_temp	アバターの試着情報を呼び出す
avatar_call_2shot	アバターの２ショット情報を呼び出す
avatar_rebuild	アバター画像およびアルバム画像の再作成
avatar_reset	アバターの初期化（クロゼットクリア、デフォルト状態）
avatar_is_valid	ユーザーコードに対するアバターの有効性確認
avatar_get_usercode	アバターキーに対するユーザーコードを取り出す
avatar_info_edit	アバター情報変更(ニックネームや性別の変更)
avatar_get	作成された複数のアバター情報を取得
avatar_item_add	クロゼットにアイテムを追加（ある場合はエラー）
avatar_item_add_force	クロゼットにアイテムを追加（ある場合は無視）
avatar_item_add_duplicate	クロゼットにアイテムを追加（同一アイテム重複可）
avatar_item_delete	クロゼットからアイテムを削除
avatar_item_puton	クロゼットのアイテムを着用（既に着用中はエラー）
avatar_item_puton_force	クロゼットのアイテムを着用（既に着用中は無視）
avatar_item_putoff	着用中のアイテムを外す（既に非着用中はエラー）
avatar_item_putoff_force	着用中のアイテムを外す（既に非着用中は無視）
avatar_item_puton_temp	アイテムを試着する
avatar_item_putoff_temp	試着中アイテムを外す
avatar_2shot	２ショットを撮る
avatar_album_add	アルバムに現在のアバター情報を保存
avatar_album_2shot_add	アルバムに２ショット情報を記録
avatar_album_delete	アルバムから写真情報を削除
avatar_album_clear	アルバムの全削除
avatar_album_edit	アルバムテキスト変更
avatar_album_info	アルバムから１枚の写真情報を取得
avatar_album_get	アルバムから複数の写真情報を取得
category_info	指定カテゴリー情報を取得

category_get	指定カテゴリーの子カテゴリー情報を取得
category_item_get	カテゴリーに属するアイテム情報を取得
category_closet_get	クロゼットのカテゴリー情報を取得
category_closet_item_get	クロゼットカテゴリーに属するアイテム情報を取得
category_rank_item_get	カテゴリーに属するアイテム情報をランキング順で取得
item_info	アイテム情報を取得
item_info_child	アイテムの子アイテム情報を取得
item_info_parent	アイテムの親アイテム情報を取得
item_get_random	アイテム情報をランダムで取得
item_serch_other_client	異なるクライアントでの同一アイテムを探す

４．API 関数詳細

省略可　引数の記入を省略することができます。ただ、次の引数を記入する場合は省略できませんので「""」などのように指定してください。

OPTION　必須項目ではないのでデータタイプに合わせて「""」か「0」の入力が可能です。ただ、引数の省略はできません。

数　数字型
字　文字列型
配　文字列の配列型
TF　[true]または[false]（「0」、「1」としても指定可）

avatar_create

新しいアバターを作成します。既に同一「site_key」・「user_code」に対してアバターが作成されている場合はエラーが返されます。

引数

字 site_key

サイトの 32Byte 識別キー（ＡＶＡＦＩＴ管理ツールより確認）

字 user_code

番組側のユーザー識別キー（**[半角英数字][-][_][.]のみ有効、最大 32Byte**）

字 sex

アバター種類の区分

※アバター種類は男・女だけに限らず、複数の種類の定義が可能です。アバター種類の定義はＡＰＩサーバーの「config/api_config.ini」ファイルを利用します。

アバター種類定義の一例）

```
$WF_AVATAR_TYPE[] = array("KEY" => "m", "NAME" => "男性", "COLOR"=>"0080FF");
$WF_AVATAR_TYPE[] = array("KEY" => "f", "NAME" => "女性", "COLOR"=>"FF0000");
$WF_AVATAR_TYPE[] = array("KEY" => "d", "NAME" => "犬",   "COLOR"=>"008000");
$WF_AVATAR_TYPE[] = array("KEY" => "c", "NAME" => "猫",   "COLOR"=>"FF00FF");
```

上記のように設定されている場合は

[m] 男性

[f] 女性

[d] 犬

[c] 猫

字 nickname 省略可

ユーザーニックネーム

字 comment 省略可

コメント

字 def_server_key 省略可

アバターを作成する画像サーバーのキー

※本システムで複数の画像サーバーを利用する場合、あるアバターの画像及びそのアバターのアルバム画像はその複数のサーバーのうち、一つのサーバーに保存されます。その保存先はシステムのほうからランダム選択で決めますが、このパラメーターで保存先を強制的に指定することもできます。画像サーバーキーの定義や確認は API サーバーの設定ファイル（api_config.ini）の「**$WF_IMG_SERVER**」をご確認ください。

return["value"]の値

Avatar Array

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_create("e13579ae01173ae199aea5e8e902ce51",
                                 "hogehoge",
                                 "f",
                                 "",
                                 "",
                                 "");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    //Avatar Array
    $avatar = $return["value"];
    …
}
```

avatar_delete

作成済みのアバターを削除します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_delete("e13579ae01173ae199aea5e8e902ce51", "hogehoge");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

avatar_call

作成済みのアバターの情報を取得します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

return["value"]の値

Avatar Array

備考

携帯端末及び PC の設定によってはアバターを表示する際に、アバター画像ファイルがキャッシュされ、着替えても更新されない場合があります。<IMAGE>タグの利用の際に画像 URL の後ろの「uniqid("_")」をつけることで URL がその都度変更されるのでその問題を解決することができます。

```
$gif_avatar = "<IMAGE src='" . $avatar["URL"] . ".gif?" . uniqid("_") . "'>";
```

また、掲示板の投稿一覧などに複数のアバターサムネールを表示する際に、各人のアバター画像 URL を求めるために人数分 API 呼び出し（avatar_call）をするとその分 API サーバーの負荷が増加されます。**アバター画像ファイル**は着替え操作があるたびに変更されますが、**ファイル名は変わりません**。そのため、アバターを新規作成する時に**画像ファイルの URL をサイト側の会員データベースなどに保存**しておき、掲示板の投稿一覧などでのアバター表示の際には API 呼び出しを行わず、保存しておいた画像 URL を利用することをお勧め致します。

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_call("e13579ae01173ae199aea5e8e902ce51", "hogehoge");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    //Avatar Array
    $avatar = $return["value"];

    //着用中アイテム
    $puton_items = $avatar["ITEMS"];

    for ($i = 0; $i < sizeof($puton_items); $i++) {
        if ($itm_exp) {
            $itm_exp .= ", ";
        }
        $itm_exp .= $puton_items[$i]["NAME"];
    }

    //アバター画像
    $gif_avatar = "<IMAGE src='".$avatar["URL"].".gif?".uniqid("_")."'>";
    $png_avatar = "<IMAGE src='".$avatar["URL"].".png?".uniqid("_")."'>";
    $jpg_avatar = "<IMAGE src='".$avatar["URL"].".jpg?".uniqid("_")."'>";

    //PC 用 HTML タグ
    $avata_tag = $avatar["TAGSET"];

    //サムネール画像
    $gif_thumbnail = "<IMAGE src='".$avatar["URL"]."_s.gif?".uniqid("_")."'>";
    $png_thumbnail = "<IMAGE src='".$avatar["URL"]."_s.png?".uniqid("_")."'>";
    $jpg_thumbnail = "<IMAGE src='".$avatar["URL"]."_s.jpg?".uniqid("_")."'>";
}
```

avatar_call_temp

アバターの試着情報を取得します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code OPTION

番組側のユーザー識別キー

※非会員からもアバター試着を試すことができるようにするため、必須ではありません。ただし、入力しない場合は「sex」を必ず指定する必要があります。

字 sex OPTION

アバター種類の区分

※「user_code」が指定されている場合は無視

※「avatar_create」を参照してください。

字 try_key

試着状態を維持するためのキー

※「avatar_item_puton_temp」の戻り値($array["TRYKEY"])を渡すことで前の状態が維持されます。

return["value"]の値

Try Array

try["TRYKEY"]

試着キー

try["URL"]

アバターの試着画像 URL（拡張子はついていない）

※「api_config.ini」の「_WF_SERVICE_MOBILE」が「true」の場合のみ返されます。

try["TAGSET"]

アバター試着 HTML タグ
※「api_config.ini」の「_WF_SERVICE_PC」が「true」の場合のみ返されます。

try["LAST_DYNAMIC_CONDITION"]
最後の着替えで利用したダイナミックキーワード条件

try["ITEM_TOTAL_COUNT"]
試着中アイテム数

try["ITEMS"]
(Item Array)
試着中アイテム詳細情報

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_call_temp("e13579ae01173ae199aea5e8e902ce51",
                                    "hogehoge",
                                    "",
                                    $try_key);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    $try = $return["value"];
    $try_key = $try["TRYKEY"];
    $url = $try["URL"] . ".gif";
    $tag = $try["TAGSET"];
    $item_count = $try["ITEM_TOTAL_COUNT"];
    $try_items = $try["ITEMS"];
}
```

avatar_call_2shot

アバターの最後に撮った２ショット情報を呼び出します。２ショットを一度も撮ったことがない場合は、エラーが返されます。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

return["value"]の値

twoshot["URL"]
アバターの２ショット画像 URL（拡張子はついていない）
※「api_config.ini」の「_WF_SERVICE_MOBILE」が「true」の場合のみ返されます。

twoshot["TAGSET"]
アバターの２ショット HTML タグ
※「api_config.ini」の「_WF_SERVICE_PC」が「true」の場合のみ返されます。

twoshot["LAST_DYNAMIC_CONDITION"]
２ショットを撮る際に指定したダイナミックキーワード条件

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_call_2shot("e13579ae01173ae199aea5e8e902ce51",
                                     "hogehoge");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    $twoshot = $return["value"];
    $url = $twoshot["URL"] . ".gif";
    $tag = $twoshot["TAGSET"];
}
```

avatar_rebuild

アバターの画像およびアルバム画像を再作成します。
※ 普段はアバターの画像を再作成する必要はありませんが、イメージサーバーの増設、統合・廃止などでアバター保存サーバーを変更する必要がある場合やイメージサーバーの故障でアバター画像やアルバム画像を再作成する必要がある場合、作成する画像のフォーマット（gif、png、jpg に対応）を増やす時、ダイナミック機能を使う時などにこの関数を利用します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 dyna_cond 省略可
ダイナミック機能を利用する場合、ダイナミックキーワード
例）「FACE:3」⇒「FACE」といったダイナミックキーワードを持つアイテムを着用中の場合、
3 番目の画像を使ってアバターを作成

字 def_server_key 省略可
アバター画像を保存するイメージサーバーのキー（指定しない場合は、自動的に決められます。また、「svr_reset」が「true」になっている必要があります。）

TF svr_reset 省略可
イメージサーバーを再決定するかどうかのフラグ
[true] 再決定する（DEFAULT）
[false] 既存サーバーをそのまま維持する

TF album_rebuild 省略可
アルバムデータも再作成の対象にするかどうかのフラグ
[true] 再作成
[false] 作成しない（DEFAULT）

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_rebuild("e13579ae01173ae199aea5e8e902ce51",
                                  "hogehoge",
                                  "",
                                  "IMG1",
                                  true,
                                  true);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

備考

本システムではサーバー故障などに備えてアバター画像及びアルバム画像を再作成するツールを別途用意しています。API サーバーの「batch」ディレクトリにある「avatar_rebuild.php」ファイルをコマンドラインから実行することにより、各画像サーバーの全員分のアバター及びアルバムデータを再作成することができます。故障だけではなく、画像サーバーの統合や分散などの際にもご利用できます。「avatar_rebuild.php」ファイルには、詳しい利用方法が含まれています。

avatar_reset

アバターを新規で作成した状態に戻します。クロゼットにアイテムがある場合はそのアイテムを削除します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

TF include_puton 省略可

着用中のアイテムも削除するかどうかのフラグ

[true] 着用中アイテムも削除

[false] 着用中アイテムを除いて未着用アイテムだけを削除 (DEFAULT)

※着用中アイテムも削除する場合は、アバター画像を変更する必要があるのでアバターを再作成します。

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_reset("e13579ae01173ae199aea5e8e902ce51",
                                "hogehoge",
                                true);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

avatar_is_valid

ユーザーコードに対するアバターの有効性（作成有無）を確認します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

return["value"]の値

is_exist
アバターが存在する場合は「true」、存在しない場合は「false」が返されます。

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_is_valid("e13579ae01173ae199aea5e8e902ce51",
                                   "hogehoge");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    $is_exist = $return["value"];

    if (!$is_exist) {
        $return = $AVAFIT->avatar_create($site_key, $user_code, $sex, "");
        …
    }
}
```

avatar_get_usercode

アバターキーに対するサイトのユーザーコード（番組側のユーザー識別キー）を取り出します。

引数

字 site_key
サイトの 32Byte 識別キー

字 avatar_key
アバターの 32Byte キー

return["value"]の値

usercode
アバターが存在する場合は usercode、存在しない場合は「false」が返されます。

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_get_usercode("e13579ae01173ae199aea5e8e902ce51",
                                       "cbf70bcad7ec4584fc07522fe6be8bb4");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    $user_code = $return["value"];

    if (!$user_code) {
        …
    }
}
```

avatar_info_edit

アバターのニックネームまたはコメント、性別情報を変更します。
※性別情報を変更する場合、着用中の旧性用アイテムはすべて外されます。（削除はしません。）

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 sex
アバター種類の区分
[m] 男性
[f] 女性
※アバター種類は男・女の区分に限らず、複数の種類を設定ファイル（api_config.ini）に定義定義して利用可能です。「avatar_create」関数を参照してください。

字 nickname 省略可
ユーザーニックネーム
※「""」を記入する場合は、ニックネームを削除しませんのでご注意してください。

字 comment 省略可
コメント

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_info_edit("e13579ae01173ae199aea5e8e902ce51",
                                    "hogehoge",
                                    "f",
                                    "さくらんぼ",
                                    "");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

avatar_get

作成された複数のアバター情報を指定の条件に合わせて取得します。

引数

字 site_key

サイトの 32Byte 識別キー

数 page 省略可

取得するアイテム情報のページ番号（1 ページから指定、省略する場合は 1 ページ）

数 pagePerRecord 省略可

１ページのアイテム数（省略する場合は、システム規定値の 10 個）

字 condition 省略可

取得条件

[key:%アバターキー%]

アバターキー条件

例 ⇒ "key:286fa32e108438ef823900a2fd8b70cb"
　　アバターキーが「286fa32e108438ef823900a2fd8b70cb」と一致するアバター

例 ⇒ "key:ef3b%"　アバターキーが「ef3b」で始まるすべてのアバター

例 ⇒ "key:%ef3b%"　アバターキーに「ef3b」が含まれるすべてのアバター

例 ⇒ "key:%ef3b"　アバターキーが「ef3b」で終わるすべてのアバター

[sex:アバター種類判別文字の組み合わせ]

※「avatar_create」を参照してください。

[name:%ニックネーム%]

ニックネーム条件

例 ⇒ "name:咲子%"　ニックネームが「咲子」で始まるすべてのアバター

例 ⇒ "name:%咲子%"　ニックネームに「咲子」の文字列が含まれるすべてのアバター

例 ⇒ "name:%咲子"　ニックネームが「咲子」で終わるすべてのアバター

[usercode:%ユーザーコード%]

ユーザーコード条件

例 ⇒ "usercode:tomato%"　ユーザーコードが「tomato」で始まるすべてのアバター

例 ⇒ "usercode:%tomato%"　ユーザーコードに「tomato」の文字列が含まれるすべてのアバター

例 ⇒ "usercode:%tomato"　ユーザーコードが「tomato」で終わるすべてのアバター

[registdate:[記号][yyyy/mm/dd hh:mm:ss]]

システムへのアバター登録日条件

例 ⇒ "registdate:<2009/12/16"　2009/12/15 以前登録アバター

例 ⇒ "registdate:<=2009/12/15 23:59:59"　2009/12/15 以前登録アバター

字 orderBy 省略可

アバター取得順（例 ⇒ "registdate desc, name asc, usercode asc"）

[name asc]　ニックネーム昇順

[name desc]　ニックネーム降順

[usercode asc]　ユーザーコード昇順

[usercode desc]　ユーザーコード降順

[registdate asc]　登録日付古い順

[registdate desc]　登録日付最新順

return["value"]の値

avatar["AVATAR_TOTAL_COUNT"]

取得条件に対する総アバター数

avatar["AVATARS"]

Avatar Array の配列

アバター詳細情報

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_get("e13579ae01173ae199aea5e8e902ce51",
                              $page,
                              10,
                              "registdate:>=2012/3/15",
                              "registdate desc");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    $avatar_info = $return["value"];
    $avatars = $avatar_info["AVATARS"];

    //表示用情報を追加
    for ($i = 0; $i < sizeof($avatars); $i++) {
        if ($avatars[$i]["SEX"] == "m") {
            $avatar_sex[$i] = "<FONT COLOR='blue'>男性</FONT>";
        }
        else if ($avatars[$i]["SEX"] == "f") {
            $avatar_sex[$i] = "<FONT COLOR='red'>女性</FONT>";
        }

        //サムネール画像
        $avatar_url[$i] = "<img src='".$avatars[$i]["URL"]."_s.gif'>";
}
```

avatar_item_add

クロゼットにアイテムを追加します。同じアイテムがクロゼットに存在する場合はエラーが返されます。異性のアイテムでも追加できます。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 配 item_keys
追加するサイトアイテムキー（複数の場合は、配列として渡します。）

数 buy_point 省略可
アイテム購入価格

[-1]	アイテムの販売価格で購入（DEFAULT）
[0]	無料で購入
[それ以外の数字]	指定価格で購入（複数のアイテムを同時に追加する場合は、追加する１番めのアイテムに指定価格が記録され、２番目以降のアイテムは「0」として記録）

return["value"]の値

なし

サンプル

PHP コード例

```
for ($i = 0; $i < sizeof($try_items); $i++) {
    if (!$try_items[$i]["IS_CLOSET"]) {
        $buy_item_keys[] = $try_items[$i]["KEY"];
        $sum_price += $try_items[$i]["PRICE"];
    }
}

$return = $AVAFIT->avatar_item_add("e13579ae01173ae199aea5e8e902ce51",
                                   "hogehoge",
                                   $buy_item_keys,
                                   $sum_price);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

関連関数

avatar_item_add_force

avatar_item_add_duplicate

avatar_item_add_force

クロゼットにアイテムを追加します。同じアイテムがクロゼットに存在する場合はエラーを返さずに無視します。異性のアイテムでも追加できます。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 配 item_keys
追加するサイトアイテムキー（複数の場合は、配列として渡します。）

数 buy_point 省略可
アイテム購入価格
[-1] アイテムの販売価格で購入（DEFAULT）
[0] 無料で購入
[それ以外の数字] 指定価格で購入（複数のアイテムを同時に追加する場合は、追加できた１番めのアイテムに指定価格が記録され、２番目以降のアイテムは「0」として記録）

return["value"]の値

なし

サンプル

PHP コード例

```
//ランダムでアイテムを 3 つ取得
$return = $AVAFIT->item_get_random($site_key, 1, $cate_index, 3, "sex:fc");
if ($return["result"]) {
    $rand_item = $return["value"];
    $item_array = $rand_item["ITEMS"];
    for ($i = 0; $i < sizeof($item_array); $i++) {
        $item_keys[] = $itm_array[$i]["KEY"];
    }
}

//アイテムをクロゼットに追加（該当アイテムが既に存在する場合は無視して、存在しないアイテムのみ追加）
$return = $AVAFIT->avatar_item_add_force($site_key,
                                         $user_code,
                                         $item_keys,
                                         -1);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

関連関数

avatar_item_add

avatar_item_add_duplicate

avatar_item_add_duplicate

クロゼットにアイテムを追加します。同じアイテムがクロゼットに存在する場合でも追加します（重複許可）。異性のアイテムでも追加できます。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 配 item_keys
追加するサイトアイテムキー（複数の場合は、配列として渡します。）

数 buy_point 省略可
アイテム購入価格
[-1] アイテムの販売価格で購入（DEFAULT）
[0] 無料で購入
[それ以外の数字] 指定価格で購入（複数のアイテムを同時に追加する場合は、追加できた１番めのアイテムに指定価格が記録され、２番目以降のアイテムは「0」として記録）

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->item_get_random($site_key, 1, $cate_index, 3, "sex:fc");
if ($return["result"]) {
    $rand_item = $return["value"];
    $item_array = $rand_item["ITEMS"];
    for ($i = 0; $i < sizeof($item_array); $i++) {
        $item_keys[] = $itm_array[$i]["KEY"];
    }
}

//追加するアイテムが既にクロゼットに存在する場合でも追加します。
$return = $AVAFIT->avatar_item_add_duplicate($site_key,
                                             $user_code,
                                             $item_keys,
                                             -1);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

関連関数

avatar_item_add_force

avatar_item_add

avatar_item_delete

クロゼットにあるアイテムを削除します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

字 配 item_keys

削除する**サイトアイテムキー**または**クロゼットアイテムキー**（複数の場合は、配列として渡します。）

※同一アイテムを複数所有する仕様（**avatar_item_add_duplicate** を利用してアイテムをクロゼットに登録する場合）では、同じアイテムがクロゼットに複数ある場合、その「サイトアイテムキー」をパラメーターに渡せば同一のアイテム全てが削除されますが、「クロゼットアイテムキー」をパラメーターに渡せばその指定したアイテム１個のみが削除されます。

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_item_delete($site_key,
                                      $user_code,
                                      $item_keys);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

avatar_item_puton

クロゼットにあるアイテムを着用します。対象アイテムを既に着用中の場合はエラーが返されます。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 配 item_keys
着用する**サイトアイテムキー**または**クロゼットアイテムキー**（複数の場合は、配列として渡します。）

字 dyna_cond 省略可
ダイナミック機能を利用する場合、ダイナミックキーワード

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_item_puton($site_key,
                                     $user_code,
                                     $item_keys,
                                     "季節:3");
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

備考

パラメーター「item_keys」に「サイトアイテムキー」を渡す代わりに、「クロゼットアイテムキー（CLOSET_ITEM_KEY)」を渡すこともできます。

同一アイテムを複数持たせることを許可する仕様（アイテムをクロゼットに挿入する際に、avatar_item_add_duplicate を利用して重複関係なくアイテムを挿入する仕様）では、一人のアバターのクロゼットにおいて、同じサイトアイテムキーを持つアイテムが複数存在する場合があります。その場合、サイトアイテムキーを渡して着用させると、そのサイトアイテムキーを持つアイテムが複数あるので、システムが勝手に一つのアイテムを選んで着用します。

もし、同じサイトアイテムキーを持つ複数のアイテムから一つを選んで着用させたい場合は、「item_keys」に「サイトアイテムキー（KEY)」を渡す代わりに「クロゼットアイテムキー（CLOSET_ITEM_KEY)」を渡すことで解決できます。「クロゼットアイテムキー」情報は、「category_closet_item_get」関数の戻り値である「CLOSET」配列から取得できます。

avatar_item_puton_force

クロゼットにあるアイテムを着用します。既に着用中の場合は無視します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 配 item_keys
着用する**サイトアイテムキー**または**クロゼットアイテムキー**（複数の場合は、配列として渡します。）

字 dyna_cond 省略可
ダイナミック機能を利用する場合、ダイナミックキーワード

return["value"]の値

なし

サンプル

PHP コード例

```
$return = $AVAFIT->item_get_random($site_key, 1, $cate_index, 3, "sex:dc");
if ($return["result"]) {
    $rand_item = $return["value"];
    $item_array = $rand_item["ITEMS"];
    for ($i = 0; $i < sizeof($item_array); $i++) {
        $item_keys[] = $itm_array[$i]["KEY"];
    }
}

$return = $AVAFIT->avatar_item_add_force($site_key,
                                         $user_code,
                                         $item_keys,
                                         -1);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
else {
    $return = $AVAFIT->avatar_item_puton_force($site_key,
                                               $user_code,
                                               $item_keys,
                                               "");
}
```

avatar_item_putoff

着用中のアイテムを外します。既に未着用の場合はエラーを返します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

字 配 item_keys

外したい着用中**サイトアイテムキー**または**クロゼットアイテムキー**（複数の場合は、配列として渡します。）

字 dyna_cond 省略可

ダイナミック機能を利用する場合、ダイナミックキーワード

return["value"]の値

なし

avatar_item_putoff_force

着用中のアイテムを外します。既に未着用の場合は無視します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 配 item_keys
外したい着用中**サイトアイテムキー**または**クロゼットアイテムキー**（複数の場合は、配列として渡します。）

字 dyna_cond 省略可
ダイナミック機能を利用する場合、ダイナミックキーワード

return["value"]の値

なし

avatar_item_puton_temp

アイテムを試着します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code OPTION

番組側のユーザー識別キー

※非会員でもアバター試着を試すことができるようにするため、この引数は必須ではありません。ただし、入力しない場合は「sex」を必ず指定する必要があります。

字 sex OPTION

アバター種類の区分

※「avatar_create」を参照してください。

※「user_code」が指定されている場合は無視

字 try_key

試着状態を維持するためのキー

※この関数の戻り値(try["TRYKEY"])を再び渡すことで前の試着状態がそのまま維持されます。また、このキーにヌルが渡された場合は、試着状況を無視して現在アバターの着用状態を基準に処理することになります。

字 配 item_keys

着用したいサイトアイテムキー（複数の場合は、配列として渡します。）

※「クロゼットアイテムキー」は使えません。

字 dyna_cond 省略可

ダイナミック機能を利用する場合、ダイナミックキーワード

return["value"]の値

try["TRYKEY"]

試着キー

try["ITEM_TOTAL_COUNT"]
試着中アイテム数

try["ITEMS"]
Item Array の配列
試着中アイテム詳細情報

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_item_puton_temp($site_key,
                                          $user_code,
                                          "",
                                          $try_key,
                                          $item_key,
                                          "");
if ($return["result"]) {
    $tryset = $return["value"];
    $try_key    = $tryset["TRYKEY"];
    $item_count = $tryset["ITEM_TOTAL_COUNT"];
    $try_items  = $tryset["ITEMS"];

    for ($i = 0; $i < sizeof($try_items); $i++) {
        echo "<img src=" . $try_items[$i]["URL"] . "><br />";
        echo $try_items[$i]["NAME"];
        …
    }
}
```

avatar_item_putoff_temp

試着中のアイテムを外します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code OPTION

番組側のユーザー識別キー

※非会員からもアバター試着を試すことができるようにするため、この引数は必須ではありません。ただし、入力しない場合は「sex」を必ず指定する必要があります。

字 sex OPTION

アバター種類の区分

※「avatar_create」を参照してください。

※「user_code」が指定されている場合は無視

字 try_key

試着状態を維持するためのキー

※この関数の戻り値(try["TRYKEY"])を再び渡すことで前の試着状態がそのまま維持されます。また、このキーにヌルが渡された場合は、試着状況を無視して現在アバターの着用状態を基準に処理することになります。

字 配 item_keys

外したい着用中サイトアイテムキー（複数の場合は、配列として渡します。）

※「クロゼットアイテムキー」は使えません。

字 dyna_cond 省略可

ダイナミック機能を利用する場合、ダイナミックキーワード

return["value"]の値

try["TRYKEY"]

試着キー

try["ITEM_TOTAL_COUNT"]
試着中アイテム数

try["ITEMS"]
Item Array の配列
試着中アイテム詳細情報

avatar_2shot

２ショット写真を撮ります。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

字 friend_code

友達のユーザー識別キー

字 dyna_cond 省略可

ダイナミック機能を利用する場合、ダイナミックキーワード

return["value"]の値

twoshot["URL"]

アバターの２ショット画像 URL（拡張子はついていない）

※「api_config.ini」の「_WF_SERVICE_MOBILE」が「true」の場合のみ返されます。

twoshot["TAGSET"]

アバターの２ショット HTML タグ

※「api_config.ini」の「_WF_SERVICE_PC」が「true」の場合のみ返されます。

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_2shot($site_code, $user_code, $friend_code, $dyna_cond);
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
else {
    $twoshot = $return["value"];
    echo "<img src=" . $twoshot["URL"] . "><br />";
}
```

avatar_album_add

アルバムに現在のアバター情報を写真として保存します。友達との２ショットも保存できます。友達との２ショット保存の場合は、自分が左側に表示され、友達が右側に表示されます。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 text 省略可
テキスト

字 friend_code 省略可
友達のユーザー識別キー
※「friend_code」を指定する場合は、「avatar_album_2shot_add」と同様の処理になります。

return["value"]の値

なし

備考

友達との２ショット保存の場合、アバターを左右にずらす間隔は「api_config.ini」ファイルで定義されています。

avatar_album_2shot_add

アルバムに２ショット写真を保存します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

字 text OPTION

テキスト

字 friend_code

２ショット相手の番組側ユーザー識別キー

return["value"]の値

なし

avatar_album_delete

アルバムから１つの写真情報を削除します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

字 album_key

アルバムキー

return["value"]の値

なし

avatar_album_clear

アルバムのすべての写真情報を削除します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

return["value"]の値

なし

avatar_album_edit

アルバムのテキスト情報を変更します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

字 album_key

アルバムキー

字 text

変更するテキスト文字列

※「""」を指定する場合は、テキスト情報を削除します。

return["value"]の値

なし

avatar_album_info

アルバムから指定のアルバムキーに対する写真（１個）情報を取得します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

字 album_key
アルバムキー

return["value"]の値

Album Array

サンプル

PHP コード例

```
$return = $AVAFIT->avatar_album_info($site_key, $user_code, $album_key);
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
else {
    $album = $return["value"];
    $album_key  = $album["KEY"];
    $album_url  = $album["URL"] . ".gif";
    $album_tag  = $album["TAGSET"];
    $album_text = $album["TEXT"];
}
```

avatar_album_get

アルバムから複数の写真情報を１ページ分取得します。

引数

字 site_key

サイトの 32Byte 識別キー

数 page 省略可

取得する写真情報のページ番号（1 ページから指定、省略する場合は１ページ）

数 pagePerRecord 省略可

１ページのアイテム数（省略する場合は、システム規定値の 10 個）

字 condition 省略可

取得条件

[sex:アバター種類判別文字の組み合わせ]

アバター種類（普段は「男性」・「女性」）はシステムによって設定が異なります。ＡＰＩサーバーの設定ファイル（api_config.ini）に定義された内容が下記の場合

アバター種類文字　m 男性、f 女性、c 猫、d 犬

条件指定は下記のようになります。

例 ⇒ "sex:m"　男性のみ
"sex:cd" 犬と猫

[name:%ニックネーム%]

ニックネーム条件

例 ⇒ "name:咲子%"　ニックネームが「咲子」で始まるすべてのアバター
例 ⇒ "name:%咲子%"　ニックネームに「咲子」の文字列が含まれるすべてのアバター
例 ⇒ "name:%咲子"　ニックネームが「咲子」で終わるすべてのアバター

[usercode:%ユーザーコード%]

ユーザーコード条件
　例 ⇒ "usercode:tomato%"　ユーザーコードが「tomato」で始まるすべてのアバター
　例 ⇒ "usercode:%tomato%"　ユーザーコードに「tomato」の文字列が含まれるすべてのアバター
　例 ⇒ "usercode:%tomato"　ユーザーコードが「tomato」で終わるすべてのアバター

[registdate:[記号][yyyy/mm/dd hh:mm:ss]]
システムへのアバター登録日条件
　例 ⇒ "registdate:<=2009/12/15 23:59:59"　2009/12/15 以前登録アバター

字 orderBy 省略可
写真の取得順
[usercode asc]　ユーザーコード昇順
[usercode desc]　ユーザーコード降順
[registdate asc]　登録日付古い順
[registdate desc] 登録日付最新順
[updatedate asc]　更新日付古い順
[updatedate desc] 更新日付最新順

return["value"]の値

album["ALBUM_TOTAL_COUNT"]
アルバムのすべての写真数

album["ALBUMS"]
Album Array の配列

サンプル

PHP コード例

```
if ($order == "古い順") {
    $order_cond = "registdate asc";
}
else {
    $order_cond = "registdate desc";
}

//女性のアルバムの中で、2012 年 3 月 15 日以降に登録又は更新したアルバムのみ表示
$cond = "registdate: >= 2012/3/15 | updatedate: >= 2012/3/15, sex:f";

$return = $AVAFIT->avatar_album_get($site_key,
                                    $page,          //ページ番号 1～
                                    5,              //1 ページに 5 件表示
                                    $cond,          //表示条件
                                    $order_cond);   //表示順
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
else {
    $album_info = $return["value"];
    $albums = $album_info["ALBUMS"];

    for ($i = 0; $i < sizeof($albums); $i++) {
        echo "<img src=" . $albums[$i]["URL"] . ".gif><br />";
        echo $albums[$i]["TEXT"];
}
```

category_info

指定カテゴリーの情報を取得します。管理ツールから「無効」に設定したカテゴリーは取得しません。

引数

字 site_key
サイトの 32Byte 識別キー

数 set_kubun 省略可
カテゴリー組番号（1～5、省略の場合は「1」）

数 category_index
取得したいカテゴリーのインデックス番号

字 condition 省略可
各カテゴリーに属するアイテム数の取得条件（例 ⇒ "sex:fc, name:水着%, price:>=200"）

[key:%サイトアイテムキー%]
サイトアイテムキー条件
例 ⇒ "key:9fdd%"　サイトアイテムキーが「9fdd」で始まるすべてのアイテム
例 ⇒ "key:%9fdd%"　サイトアイテムキーに「9fdd」が含まれるすべてのアイテム
例 ⇒ "key:%9fdd"　サイトアイテムキーが「9fdd」で終わるすべてのアイテム

[sex: アバター種類判別文字の組み合わせ]
アイテムのアバター種類条件（「avatar_album_get」を参照してください。）

[name:%アイテム名%]
アイテム名条件
例 ⇒ "name:水着%"　アイテム名が「水着」で始まるすべてのアイテム
例 ⇒ "name:%水着%"　アイテム名に「水着」という文字列が含まれるすべてのアイテム
例 ⇒ "name:%水着"　アイテム名が「水着」で終わるすべてのアイテム

[price:[記号][数字]]
販売価格条件

例 ⇒ "price:>=300" 300 以上
例 ⇒ "price:<150" 150 未満

[selldate:[記号][yyyy/mm/dd hh:mm:ss]]
販売開始日条件
例 ⇒ "selldate:>=2009/10/15" 2009/10/15 以降公開アイテム

[registdate:[記号][yyyy/mm/dd hh:mm:ss]]
システムへのアイテム登録日条件
例 ⇒ "registdate:<2009/12/15"
2009/12/14 以前登録アイテム、2009/12/15 は含まない

[updatedate:[記号][yyyy/mm/dd hh:mm:ss]]
システム上のアイテム更新日条件
例 ⇒ "updatedate:>=2009/10/09 00:00:00"
2009/10/09 以降更新アイテム、2009/10/09 は含まれる

[option1:%キーワード%]
[option2:%キーワード%]
[optionXX:%キーワード%]
文字型オプション条件（使い方はアイテム名条件と同様）

[value1:[記号][数字]]
[value2:[記号][数字]]
[valueXX:[記号][数字]]
数字型オプション条件（使い方は販売価格条件と同様）

return["value"]の値

Category Array

category_get

指定カテゴリーに属する子カテゴリー情報を取得します。管理ツールから「無効」に設定したカテゴリーは取得しません。

引数

字 site_key
サイトの 32Byte 識別キー

数 set_kubun 省略可
カテゴリー組番号（1～5、省略の場合は「1」）

数 category_index 省略可
カテゴリーのインデックス番号（指定しない場合はトップレベルからのカテゴリー情報を取得します。また、指定する場合は、指定したカテゴリーに属するサブカテゴリー情報を取得します。）

字 condition 省略可
各カテゴリーに属するアイテム数の取得条件（例 ⇒ "sex:fc, name:水着%, price:>=200"）

[key:%サイトアイテムキー%]
サイトアイテムキー条件
例 ⇒ "key:9fdd%"　サイトアイテムキーが「9fdd」で始まるすべてのアイテム
例 ⇒ "key:%9fdd%"　サイトアイテムキーに「9fdd」が含まれるすべてのアイテム
例 ⇒ "key:%9fdd"　サイトアイテムキーが「9fdd」で終わるすべてのアイテム

[sex: アバター種類判別文字の組み合わせ]
アイテムのアバター種類条件（「avatar_album_get」を参照してください。）

[name:%アイテム名%]
アイテム名条件
例 ⇒ "name:水着%"　アイテム名が「水着」で始まるすべてのアイテム
例 ⇒ "name:%水着%"　アイテム名に「水着」という文字列が含まれるすべてのアイテム
例 ⇒ "name:%水着"　アイテム名が「水着」で終わるすべてのアイテム

[price:[記号][数字]]
販売価格条件
例 ⇒ "price:>=300" 300 以上
例 ⇒ "price:<150" 150 未満

[selldate:[記号][yyyy/mm/dd hh:mm:ss]]
販売開始日条件
例 ⇒ "selldate:>=2009/10/15" 2009/10/15 以降公開アイテム

[registdate:[記号][yyyy/mm/dd hh:mm:ss]]
システムへのアイテム登録日条件
例 ⇒ "registdate:<2009/12/15"
2009/12/14 以前登録アイテム、2009/12/15 は含まない

[updatedate:[記号][yyyy/mm/dd hh:mm:ss]]
システム上のアイテム更新日条件
例 ⇒ "updatedate:>=2009/10/09 00:00:00"
2009/10/09 以降更新アイテム、2009/10/09 は含まれる

[option1:%キーワード%]
[option2:%キーワード%]
[optionXX:%キーワード%]
文字型オプション条件（使い方はアイテム名条件と同様）

[value1:[記号][数字]]
[value2:[記号][数字]]
[valueXX:[記号][数字]]
数字型オプション条件（使い方は販売価格条件と同様）

※オプションの数（option1～optionXX 又は value1～valueXX）はシステムによって異なります。ご利用中のシステムでのオプション数は AVAFIT 管理機能のトップページよりご確認できます。

TF include_sub 省略可
子カテゴリー情報を取得するかどうかのフラグ
[true] 子カテゴリーも含めて情報を取得します。(DEFAULT)

[false] 子カテゴリーは情報に含まれません。

return["value"]の値

Category Array の配列

サンプル

PHP コード例

```
//検索条件としてアバター種類は女性用アイテムのみ、販売価格は無料アイテム
$cond = "sex:f, price:<=0";

//カテゴリー取得
$return = $AVAFIT->category_get($site_key,
                                1,          //第１カテゴリー組
                                "",         //トップレベルからカテゴリー取得
                                $cond,      //カテゴリーのアイテム数
                                true);      //子カテゴリー情報も取得
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
else {
    $category = $return["value"];

    //見栄えのためにカテゴリー配列にインデント挿入
    for ($i = 0; $i < sizeof($category); $i++) {
        $depth = "";
        for ($j = 1; $j <= $category[$i]["DEPTH"]; $j++) {
            $depth .= "   ";
        }
        $exp = $depth."+ ".$category[$i]["NAME"];
        if ($category[$i]["ITEM_COUNT"]) {
            $exp .= "(".$category[$i]["ITEM_COUNT"].")";
        }
        //カテゴリー一覧を配列に保存
        $cate[] = array("KEY"  => $category[$i]["INDEX"],
                        "NAME" => $exp);
    }
}
```

category_item_get

該当カテゴリーに属するアイテム情報を取得します。子アイテムは含まれません。また、管理ツールから「無効」に設定したカテゴリーのアイテム情報は取得しません。

引数

字 site_key
サイトの 32Byte 識別キー

数 set_kubun 省略可
カテゴリー組番号（1～5、省略の場合は「1」）

数 category_index 省略可
カテゴリーのインデックス番号（指定しない場合はすべてのアイテムが対象になります。また、指定する場合は、指定したカテゴリーに属するアイテム情報を取得します。）

数 page 省略可
取得するアイテム情報のページ番号（1 ページから指定、省略する場合は 1 ページ）

数 pagePerRecord 省略可
１ページのアイテム数（省略する場合は、システム規定値の 10 個）

字 condition 省略可
取得条件、「category_get」と同様

字 orderBy 省略可
アイテム取得順（例 ⇒ "price desc, selldate asc"）

[name asc]	アイテム名順
[name desc]	アイテム名逆順
[price asc]	販売価格安順
[price desc]	販売価格高順
[selldate asc]	販売日付古い順
[selldate desc]	販売日付最新順
[registdate asc]	登録日付古い順
[registdate desc]	登録日付最新順

[updatedate asc]　更新日付古い順
[updatedate desc] 登録日付最新順
[value1 asc] 数字型オプション１の昇順
[value1 desc] 数字型オプション１の降順
～
[valueXX asc] 数字型オプション XX の昇順
[valueXX desc] 数字型オプション XX の降順

TF include_sub　省略可
子カテゴリー情報を取得するかどうかのフラグ
[true]　子カテゴリーも含めて情報を取得します。(DEFAULT)
[false] 子カテゴリーは情報に含まれません。

return["value"]の値

category["ITEM_TOTAL_COUNT"]
取得条件に対する総アイテム数

category["ITEMS"]
Item Array の配列
アイテム詳細情報

※　この関数は番組側のユーザー識別キーをパラメーターとして渡さないため、Item Array の「IS_CLOSET」、「IS_PUTON」は「0」がセットされることに注意

サンプル

PHP コード例

```
$cond = "sex:c".$sex.",price:<=0";
$order = "selldate asc, name desc";
$return = $AVAFIT->category_item_get($site_key,
                                     1,          //カテゴリー組
                                     $f_cate,    //カテゴリーINDEX
                                     $page,      //表示ページ番号
                                     5,          //アイテム表示数
                                     $cond,      //表示条件
                                     $order,     //表示順
                                     true);      //サブカテゴリー含む
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
else {
    $item_info = $return["value"];
    $items = $item_info["ITEMS"];
}
```

category_closet_get

クロゼットにあるアイテムのカテゴリー情報を取得します。アイテムのないカテゴリーは取得しません。また、「category_get」とは異なり、管理ツールから「無効」に設定したカテゴリーであってもアイテムが存在する場合は取得します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

数 set_kubun 省略可
カテゴリー組番号（1～5、省略の場合は「1」）

数 category_index 省略可
カテゴリーのインデックス番号（指定しない場合はトップレベルからのカテゴリー情報を取得します。また、指定する場合は、指定したカテゴリーに属するサブカテゴリー情報を取得します。）

字 condition 省略可
各カテゴリーに属するアイテム数の取得条件
（例 ⇒ "name:水着%, price:>=200 | price: < 100"）

[key:%サイトアイテムキー%]
サイトアイテムキー条件
例 ⇒ "key: 9fdd%" サイトアイテムキーが「9fdd」で始まるすべてのアイテム
例 ⇒ "key:% 9fdd%" サイトアイテムキーに「9fdd」が含まれるすべてのアイテム
例 ⇒ "key:% 9fdd " サイトアイテムキーが「9fdd」で終わるすべてのアイテム

[name:%アイテム名%]
アイテム名条件
例 ⇒ "name:水着%" アイテム名が「水着」で始まるすべてのアイテム
例 ⇒ "name:%水着%" アイテム名に「水着」という文字列が含まれるすべてのアイテム

例 ⇒ "name:%水着"　アイテム名が「水着」で終わるすべてのアイテム

[price:[記号][数字]]
購入価格条件
例 ⇒ "price:>=300"　300 以上
例 ⇒ "price:<150"　150 未満

[registdate:[記号][yyyy/mm/dd hh:mm:ss]]
システムへのアイテム登録日条件
例 ⇒ "selldate:<=2009/12/15 23:59:59"　2009/12/15 以前登録アイテム

[status:puton, putoff]
着用中有無
例 ⇒ "status:puton"　着用中アイテム
例 ⇒ "status:putoff"　未着用中アイテム

TF include_sub 省略可
子カテゴリー情報を取得するかどうかのフラグ
[true]　子カテゴリーも含めて情報を取得します。(DEFAULT)
[false] 子カテゴリーは情報に含まれません。

return["value"]の値

Category Array の配列

category_closet_item_get

クロゼットカテゴリーに属するアイテム情報を取得します。子アイテムも含まれます。また、管理ツールから「無効」に設定したカテゴリーであっても、アイテムがある場合はアイテム情報を取得します。

引数

字 site_key
サイトの 32Byte 識別キー

字 user_code
番組側のユーザー識別キー

数 set_kubun 省略可
カテゴリー組番号（1～3、省略の場合は「1」）

数 category_index 省略可
カテゴリーのインデックス番号（指定しない場合はすべてのアイテムが対象になります。また、指定する場合は、指定したカテゴリーに属するアイテム情報を取得します。）

数 page 省略可
取得するアイテム情報のページ番号（1 ページから指定、省略する場合は 1 ページ）

数 pagePerRecord 省略可
１ページのアイテム数（省略する場合は、システム規定値の 10 個）

字 condition 省略可
取得条件、「category_closet_get」と同様

字 orderBy 省略可
アイテム取得順（例 ⇒ "price desc, selldate asc"）
[name asc]　　アイテム名順
[name desc]　　アイテム名逆順
[price asc]　　販売価格安順
[price desc]　　販売価格高順

[registdate asc]　登録日付古い順
[registdate desc] 登録日付最新順

TF include_sub
子カテゴリー情報を取得するかどうかのフラグ
[true]　子カテゴリーも含めて情報を取得します。(DEFAULT)
[false] 子カテゴリーは情報に含まれません。

return["value"]の値

closet["CLOSET_TOTAL_COUNT"]
取得条件に対する総アイテム数

closet["CLOSETS"]
Closet Array の配列
クロゼットアイテム詳細情報（Item Array と異なることにご注意してください。）

category_rank_item_get

カテゴリーに属するアイテム情報をアイテム購入ランキング順で取得します。

引数

字 site_key

サイトの 32Byte 識別キー

数 set_kubun 省略可

カテゴリー組番号（1～3、省略の場合は「1」）

数 category_index 省略可

カテゴリーのインデックス番号（指定しない場合はすべてのアイテムが対象になります。
また、指定する場合は、指定したカテゴリーに属するアイテム情報を取得します。）

数 page 省略可

取得するアイテム情報のページ番号（1 ページから指定、省略する場合は１ページ）

数 pagePerRecord 省略可

１ページのアイテム数（省略する場合は、システム規定値の 10 個）

字 condition 省略可

「category_get」と基本的には同様ですが、下記が追加されます。

[rankdate:[記号][yyyy/mm/dd hh:mm:ss]]

例）⇒ rankdate:>=2009/09/01(2009/09/01 以降のアイテム購入ランキング)

例）⇒ rankdate:>=2009/09/01, rankdate:<2010/03/01
(2009/09/01 から 2010/02/末日までのアイテム購入ランキング)

例）⇒ rankdate:>=2010/03/01 00:00:00, rankdate:<=2010/03/01 23:59:59
(2010/03/01 のアイテム購入ランキング)

※取得期間が長い場合は負荷がかかる恐れがありますのでご注意してください。(1 週間以内の指定をお勧め致します。)

TF include_sub 省略可

子カテゴリー情報を取得するかどうかのフラグ

[true] 子カテゴリーも含めて情報を取得します。(DEFAULT)

[false] 子カテゴリーは情報に含まれません。

return["value"]の値

category["ITEM_TOTAL_COUNT"]
取得条件に対する総アイテム数

category["ITEMS"]
Item Array の配列
アイテム詳細情報

備考

基本的な使い方は「category_item_get」と同様です。

item_info

該当アイテムの詳細情報を取得します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code

番組側のユーザー識別キー

配 字 item_keys

サイトアイテムキー（複数の場合は、配列として渡します。）

return["value"]の値

item_info["ITEM_TOTAL_COUNT"]

情報を返すアイテム数

item_info["ITEMS"]

Item Array の配列

アイテムの詳細情報

item_info_child

該当アイテムの子アイテム情報を取得します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code OPTION

番組側のユーザー識別キー

※必須項目ではなく、「""」を指定する場合はユーザーが特定できないのでクロゼットに該当アイテムが存在するかどうかの情報が取得できない。

字 item_key

サイトアイテムキー

数 page 省略可

取得するアイテム情報のページ番号（1 ページから指定、省略する場合は 1 ページ）

数 pagePerRecord 省略可

１ページのアイテム数（省略する場合は、システム規定値の 10 個）

return["value"]の値

item_info["ITEM_TOTAL_COUNT"]

情報を返すアイテム数

item_info["ITEMS"]

Item Array の配列

アイテムの詳細情報

item_info_parent

該当アイテムの親アイテム情報を取得します。

引数

字 site_key

サイトの 32Byte 識別キー

字 user_code OPTION

番組側のユーザー識別キー

※必須項目ではなく、「""」を指定する場合はユーザーが特定できないのでクロゼットに該当アイテムが存在するかどうかの情報が取得できない。

字 item_key

サイトアイテムキー

return["value"]の値

Item Array

アイテムの詳細情報（アイテム１個）

サンプル

PHP コード例

```
$return = $AVAFIT->item_info_parent($site_key,
                                    $user_code,
                                    $item_key);
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
else {
    //親アイテム情報
    $parent_info = $return["value"];

    //親アイテムの子アイテムが現在のアイテム以外にもある
    if ($parent_info["CHILD_COUNT"]>1){
        //親アイテムの子アイテム情報
        $return = $AVAFIT->item_info_child($site_key,
                                           $user_code,
                                           $parent_info["KEY"],
                                           1, 5);
        if (!$return["result"]) {
            $ERROR[] = $return["error_shot"];
        }
        else {
            //親の子アイテム情報
            $child_info = $return["value"];
            $children = $child_info["ITEMS"];
        }
    }
}
```

item_get_random

指定した条件に一致するアイテムに限り、アイテム情報をランダムで取得します。

引数

字 site_key
サイトの 32Byte 識別キー

数 set_kubun 省略可
カテゴリー組番号（1～3、省略の場合は「1」）

数 category_index 省略可
カテゴリーのインデックス番号（指定しない場合はすべてのアイテムが対象になります。
また、指定する場合は、指定したカテゴリーに属するアイテム情報を取得します。）

数 count 省略可
取得するアイテム数（指定しない場合は 1 個）

字 condition 省略可
取得条件、「category_get」と同様

return["value"]の値

item_info["ITEMS"]
Item Array の配列
アイテムの詳細情報

サンプル

PHP コード例

```
//ガチャガチャ用アイテムの取得条件（性別一致・100P 以上）
$cond  = "sex:c".$sex.",";
$cond .= "price:>=100";

//ガチャガチャ用アイテムをランダム取得
while(1) {
    $return = $AVAFIT->item_get_random($site_key, 1, "", 1, $cond);
    if ($return["result"]) {
        $rand_info = $return["value"];
        $rand_items = $rand_info["ITEMS"];

        //所有しているアイテムか確認
        $return = $AVAFIT->item_info($site_key, $user_code, $rand_items[0]["KEY"]);
        if ($return["result"]) {
            $item_info = $return["value"];
            $items = $item_info["ITEMS"];
            if (!$items[0]["IS_CLOSET"]) {
                //所有していないアイテムであれば OK
                $select_item_key = $rand_items[0]["KEY"];
                break;
            }
        }
    }

    $cnt++;
    //3 回以上回しても探せない場合は検索中止
    if ($cnt >= 3) {
        break;
    }
}
if ($select_item_key) {
    $AVAFIT->avatar_item_add($site_key, $user_code, $select_item_key, 100);
}
```

item_serch_other_client

異なるクライアントでの同一アイテムを探してサイトアイテムキーを返します。

引数

字 from_site_key
元サイトの 32Byte 識別キー

字 item_key
元サイトのサイトアイテムキー

字 to_site_key
検索対象サイトキー

return["value"]の値

item_key
検索対象サイトでのサイトアイテムキー

Ⅶ 設定ファイル

ＡＶＡＦＩＴでは、アバターに関するほとんどの管理は管理ツールを通して行うことになりますが、サーバー増設・出力画像フォーマットの調整、その他の各種制御など、環境に関する設定は設定ファイルで行います。ＡＶＡＦＩＴを導入していただく際には、ＷＥＢＦＩＴでサーバー設置を含めて各種の設定ファイルの調整も行ってお渡しすることになりますので、そのまま使っていただいて問題ございませんが、設置サーバーの変更などサーバーの環境的な情報が変更される場合は、下記情報を参考に設定し直してください。

ＡＶＡＦＩＴの設定ファイルは、機能ごとに API 用設定ファイル（**api_config.ini**）、画像処理用設定ファイル（**img_config.ini**）、管理ツール用設定ファイル（**config.ini**）として構成されています。

１．api_config.ini

位置 ⇒ API SERVER/config/api_config.ini

● _WF_SERVICE_MOBILE

携帯端末用合成イメージの利用有無

● _WF_SERVICE_PC

PC 端末用 HTML タグの利用有無

※PC サービスがない場合は、負荷軽減のため false にしてください。

● _WF_ANIMATION_GIF

携帯端末用アニメーション GIF 合成の利用有無

※アニメーション GIF 合成を有効にするためには、画像サーバーに ImageMagick＋Imagick Library のインストールを行う必要があります。このフラグが false の場合は GD Library を利用してアバターを合成します。

● _WF_ANIMATION_WITH_PC_ITEM

携帯端末用アニメーション GIF 合成を行う場合、合成に利用するアイテム画像として PC 用アイテム画像ファイルを使うかどうかのフラグ。普段、PC 用アイテムはアニメーション GIF を利用するパターンが多く、携帯端末用アニメーション GIF 合成で PC 用画像をそのまま利用することになります。このフラグが false の場合は、携帯端末でアニメーション効果を

有効にするためには、携帯端末用アイテムもアニメーション GIF を登録する必要があります。

● _WF_IMG_CONNECT_METHOD

画像サーバー接続方法(DIRECT / SOAP)

※DIRECT はすべての機能を 1 台に搭載する場合のみ有効です。

※SOAP を利用するためには SOAP ライブラリーが必要です。

● _WF_IMG_CONNECT_TIMEOUT

画像サーバー接続待ち時間(sec、SOAP の場合)

● _WF_LOG_SAVE

API 関数の呼び出しエラーをデータベースに記録し、管理画面から閲覧できるようにするかどうかのフラグ

● _WF_ITEM_WIDTH

アバターサイズ(横)

※「img_config.ini」での定義に合わせる必要があります。

● _WF_ITEM_HEIGHT

アバターサイズ(縦)

※「img_config.ini」での定義に合わせる必要があります。

● _WF_2SHOT_SHIFT_WIDTH

２ショットで、アバターを中央からずらすピクセル数

● _WF_DEF_ITEM_FORMAT

システムで使われるアイテムファイルのフォーマット（拡張子）

● $WF_AVATAR_TYPE

アバターの種類を定義します。

例）

```
$WF_AVATAR_TYPE[] = array("KEY" => "m", "NAME" => "男性", "COLOR"=>"0080FF");
$WF_AVATAR_TYPE[] = array("KEY" => "f", "NAME" => "女性", "COLOR"=>"FF0000");
$WF_AVATAR_TYPE[] = array("KEY" => "d", "NAME" => "犬",   "COLOR"=>"008000");
$WF_AVATAR_TYPE[] = array("KEY" => "c", "NAME" => "猫",   "COLOR"=>"FF00FF");
```

上記のように定義された場合、システムではアバター種類として４つの区分になり、アイテム登録からアバター作成までのすべての処理で使われることになります。

※"KEY"フィールドは１文字だけの扱いになります。また、データベースでの定義と一致させる必要がありますので、導入後に変更が必要な場合はお問合わせください。

● $WF_IMG_SERVER

画像サーバーの情報を記述します。

[KEY]

画像サーバーのキーを定義します。英数字の簡単な文字列で指定します。

例）"IMG1"

[URL]

AVAFIT の画像サーバーURL を記入します。

例）"http://www.xxxx.com/avafit_pro/img_server"

※最後に「/」はつけないでください。

[URL_CONNECT]

基本的には「URL」と同じで問題ございません。「URL」は API 関数からアイテムやアバター画像の URL 情報を渡す際にも使われ、ユーザーの端末でアバターやアイテムの画像が表示される際の URL になりますので、グローバル IP 又は同等のドメインになる必要があります。それに比べて、「URL_CONNECT」は API サーバーから画像サーバーに接続する際に使われるのでローカル IP 指定でもかまいません。AVAFIT システムを構成する各サーバー間の通信を少しでも早くするためにはローカル定義が望ましいです。

[ITEM]

アイテムの画像ファイルを保管するサブディレクトリ名

[AVATAR]

画像サーバーのアバター作成用サブディレクトリ

[TRYSET]

画像サーバーの試着画像作成用サブディレクトリ

[2SHOT]
画像サーバーの２ショット画像作成用サブディレクトリ
[ALBUM]
画像サーバーのアルバム保存用サブディレクトリ

例）

```
$WF_IMG_SERVER[] = array("KEY"         =>"IMG1",                              //画像サーバーキー
                         "URL"         =>"http://123.45.678.99/img_server",   //画像配信用 URL
                         "URL_CONNECT" =>"http://192.168.0.99/img_server",    //内部接続用 URL
                         "ITEM"        =>"items",                             //ITEM 画像 DIR
                         "AVATAR"      =>"data/avatar",                       //AVATAR 画像 DIR
                         "TRYSET"      =>"data/try",                          //試着画像 DIR
                         "2SHOT"       =>"data/2shot",                        //2Shot 画像 DIR
                         "ALBUM"       =>"data/album");                       //ALBUM 画像 DIR
$WF_IMG_SERVER[] = array("KEY"         =>"IMG2",
                         "URL"         =>"http:// 123.45.678.100/img_server",
                         "URL_CONNECT" =>"http://192.168.0.100/img_server",
                         "ITEM"        =>"items",
                         "AVATAR"      =>"data/avatar",
                         "TRYSET"      =>"data/try",
                         "2SHOT"       =>"data/2shot",
                         "ALBUM"       =>"data/album");
```

● $WF_DB_SERVER

データベースサーバーの接続情報を記述します。

※配列の一番目のレコードが自動的にマスタデータベースとして認識されますのでご注意してください。

なお、データベースをマルチ構成する場合、マスタデータベースは１台のみになります。

[HOST]
データベースサーバーのホスト名（IP アドレス指定可）
[DBNAME]
AVAFIT 用データベースのデータベース名
[USER]
アクセスユーザー名
[PASSWD]

アクセスユーザー用パスワード

[S_WEIGHT]

スレーブサーバー（Slave Server）としての処理割合（すべての合計が必ず 100 になる必要はありません。）

※マスタデータベースサーバーも、スレーブサーバーの役割を担当させることが可能です。

※マスタ／スレーブのレプリケーション機能を利用する環境では、CREATE／INSERT／UPDATE／DELETE／DROP などのようなデータに変更をもたらす SQL 構文はマスタ側に送られ、SELECT のような参照用の SQL 構文はスレーブに送られるようになります。ここでのスレーブサーバーとしての割合は、SELECT のような参照用 SQL 構文を処理する割合を示します。

例）

```
$WF_DB_SERVER[0] = array("HOST"     =>"192.168.0.4",     //最初が「MASTER」DB
                         "DBNAME"   =>"avafit",
                         "USER"     =>"avafit_api",
                         "PASSWD"   =>"********",
                         "S_WEIGHT" =>40);               //SLAVE サーバーとしての処理割合
$WF_DB_SERVER[1] = array("HOST"     =>"192.168.0.3",     //2 番目以降は「SLAVE」DB
                         "DBNAME"   =>"avafit",
                         "USER"     =>"avafit_api",
                         "PASSWD"   =>"********",
                         "S_WEIGHT" =>60);               //SLAVE サーバーとしての処理割合
```

２．img_config.ini

位置 ⇒ IMAGE SERVER/config/img_config.ini

● _WI_IMG_ITEM_DIR

アイテム保存ディレクトリ

※システムがそのディレクトリに書き込みできるファイルアクセス権限を与えてください。

● _WI_IMG_AVATAR_DIR

アバター作成用ディレクトリ（書き込み権限必要）

● _WI_IMG_TRY_DIR

アバターの試着画像作成用ディレクトリ（書き込み権限必要）

● _WI_IMG_2SHOT_DIR

２ショット画像作成用ディレクトリ（書き込み権限必要）

● _WI_IMG_ALBUM_DIR

アルバム保存用ディレクトリ（書き込み権限必要）

● _WI_IMG_ANI_MODULE

アニメーション合成処理モジュール（ImageMagick）の設置場所（環境によって設置ディレクトリが異なる場合に対応）

● _WI_IMG_ITEM_WIDTH

アバターサイズ(横 pixel)

● _WI_IMG_ITEM_HEIGHT

アバターサイズ(縦 pixel)

● _WI_IMG_JPG_IMAGE_QUALITY

アバター合成の際に JPEG 画像出力がある場合、その JPEG 画像の画質

※デフォルトは 75 になります。100 はフルー画質になりますが、その分合成速度低下及びファイルサイズの肥大化に繋がりますのでご注意ください。

● _WI_IMG_PNG_IMAGE_QUALITY

アバター合成の際に PNG 画像出力がある場合、その PNG 画像の画質
※0～9 までの数字になります。数字が低いほど高画質になります。0 は圧縮ないファイルになりますので、容量が極端に大きくなります。

● **$WI_REQUEST**

画像サーバーが作るアバター画像の詳細を定義します。

[出力形式]

下記のいずれかを指定してください。

_WI_IMG_TYPE_GIF　　GIF フォーマット
_WI_IMG_TYPE_PNG　　PNG フォーマット
_WI_IMG_TYPE_JPG　　JPG フォーマット

[元画始点 X]

アバター画像を作成する際に、元画像ファイル（レイヤー用アイテムファイル）から切り抜くための開始 X 座標（普段、全身表示アバターは「0」になります。また、サムネール画像の場合は顔の左側のどこかの座標を指定します。）

[元画始点 Y]

アバター画像を作成する際に、元画像ファイル（レイヤー用アイテムファイル）から切り抜くための開始 Y 座標（普段、全身表示アバターは「0」になります。また、サムネール画像の場合は顔の左側のどこかの座標を指定します。）

[元画切取 W]

アバター画像を作成する際に、元画像ファイル（レイヤー用アイテムファイル）から切り抜くための幅（普段、全身表示アバターは「**_WI_IMG_ITEM_WIDTH**」になります。また、サムネール画像の場合はその横幅になります。）

[元画切取 H]

アバター画像を作成する際に、元画像ファイル（レイヤー用アイテムファイル）から切り抜くための幅（普段、全身表示アバターは「**_WI_IMG_ITEM_HEIGHT**」になります。また、サムネール画像の場合はその縦幅になります。）

[出力画像 W]

基本的に、「**元画切取 W**」と同様に指定します。

[出力画像 H]

基本的に、「**元画切取 H**」と同様に指定します。

[拡張子]

出力画像ファイルの拡張子を設定します。システムはアバター用画像ファイルを作る際に、ここで指定している拡張子をそのまま付けてファイルを保存することになります。もし、同一フォーマットファイルを 2 個以上作る場合（全身用 GIF、サムネール用 GIF）は、ファイル名が同一にならないようにしておく必要があります。同じフォーマットの２つファイルが同じファイル名になっている場合は、最後に作られた場合が最初に作られたファイルを上書きしてしまいますのでご注意してください。なお、「**出力形式**」に拡張子を合わせてください。

[試着]

対象のイメージフォーマットに対して、試着画像を作るかどうかのフラグになります。作る場合は「1」、作らない場合は「0」を設定します。使う予定がない場合は、サーバー負荷を軽減するために、「0」にしてください。

[2SHOT]

対象のイメージフォーマットに対して、２ショット画像を作るかどうかのフラグになります。使う予定がない場合は、「0」にしてください。

[Album]

対象のイメージフォーマットに対して、アルバム画像を作るかどうかのフラグになります。使う予定がない場合は、「0」にしてください。

３．mng_config.ini（管理ツール用）

位置 ⇒ MANAGEMENT SERVER/config/mng_config.ini

● $WM_API_SERVER

[DIR]

ダイレクト接続（クラス参照）の際に使われます。他の通信手段を利用して API サーバーに接続する場合は必要ありません。

[URL]

SOAP 通信及び PHPRPC 通信で API サーバーに接続する場合は、この項目を定義する必要があります。

[ENCODE]

SOAP 通信で API サーバーを繋げる際のエンコードタイプ（デフォルトは「UTF-8」）

[LOGIN]

SOAP 通信で API サーバーを繋げる際の接続アカウント情報（API サーバーにベーシック認証が設定されている場合のみ使われます。）

[PASSWD]

SOAP 通信で API サーバーを繋げる際の接続パスワード情報（API サーバーにベーシック認証が設定されている場合のみ使われます。）

● $WM_DB_SERVER

ここで定義するデータベースサーバーは API 用データベースではありません。API 用データベースへのアプローチはすべて API を通すため、管理ツールより直接つなげる必要はありません。管理ツールより接続するデータベースは、管理ツール用の独自データベースになります。担当者登録、権限設定、アクセスログなどのデータを扱うデータベースとして API 用データベースとは分離されています。

● _WM_CNT_PER_PAGE

各一覧ページで表示するデフォルトレコード数です。ＡＶＡＦＩＴ管理ツールは担当者・画面ごとにページあたりレコード表示数を保存しておき、再度ご利用の際には設定が復元できるようになっています。デフォルトレコード数は、各ページを初めて利用する場合や表示レコード件数を選択していない場合に使われます。

● _WM_API_CONNECT_METHOD

管理ツールを API サーバーにつなげる方法を設定します。

● _WM_ITEM_TEMP_DIR

AVAFIT では画像サーバーを複数利用することができますが、各画像サーバーに保存されるアイテムファイルは同じ構成になります。すなわち、一つのアイテムファイルをシステムに登録するとすべての画像サーバーに自動的に格納される仕組みになっています。担当者が管理ツールを使ってアイテムファイルをアップロードすると、この項目で定義されているディレクトリにアイテムファイルが保存され、すべての画像サーバーからこのディレクトリを参照してアイテムファイルを取得することになります。そのために、ここで定義するディレクトリは画像サーバーから参照できるようにしておく必要があります。また、書き込みできるファイルアクセス権限を与えておく必要もあります。

● _WM_ITEM_UPLOAD_URL

基本的な役割は「_WM_ITEM_TEMP_DIR」と同様ですが、アイテムの一括登録の際に使われます。「_WM_ITEM_UPLOAD_URL」は必ずしも管理サーバーと同じサーバーになる必要がありません。外部サーバーの URL でも問題ございませんが、各画像サーバーから 80 番ポートで繋がる必要があります。

● _WM_CSV_SAVE_DIR

アイテム及びサイトアイテムの一覧情報を CSV で出力する際に使われるテンポラリーディレクトリ（アクセス権限が 777 である必要があります。）

● _WM_THUMBNAIL_SUFFIX

サムネール画像の接尾文字になります。画像サーバーの「img_config.ini」で定義する「$WI_REQUEST」の[拡張子]に合わせてください。

● _WM_ITEM_FILE_T

アイテム一括登録時に使われる「サムネール（一覧表示）」用アイテムファイル形式

● _WM_ITEM_FILE_T2

アイテム一括登録時に使われる「サムネール（一覧表示）[2 枚目]」用アイテムファイル形式

● _WM_ITEM_FILE_M_F

アイテム一括登録時に使われる「携帯正面」用アイテムファイル形式

- _WM_ITEM_FILE_M_B

アイテム一括登録時に使われる「携帯背面」用アイテムファイル形式

- _WM_ITEM_FILE_P_F

アイテム一括登録時に使われる「PC 正面」用アイテムファイル形式

- _WM_ITEM_FILE_P_B

アイテム一括登録時に使われる「PC 背面」用アイテムファイル形式

Ⅷ 運用・管理テクニック

１．携帯端末向けのアニメーション GIF アバター作成

ＡＶＡＦＩＴではアニメーション GIF 合成に対応しております。携帯端末へのアバターサービスにおいても、アニメーション効果を活かして動的なアバターをサービスすることが可能です。アニメーションアバターを作成するためには、幾つか前提条件があります。

- **素材ファイル（アイテム）がアニメーション GIF ファイル**
- **システムでの利用設定**
- **システムへのライブラリーインストール**

１）アニメーション GIF アイテム

当然のことですが、アイテムファイルをアニメーション GIF ファイルとしてご用意し、システムへ登録してください。もちろん、すべてのアイテムをアニメーション GIF として作る必要はなく、動かしたいアイテムだけをアニメーション効果の入った GIF ファイルとして作成して登録します。

一つ確認して頂きたいのは、本システムではレイヤー毎にアニメーション効果を使うか使わないかを選択する機能があり、アニメーション効果を利用すると設定したレイヤーのアイテムのみアニメーション合成処理を行いますので、アニメーション化するアイテムはレイヤー毎に用意するのが理想的です。

２）システムでの利用設定

設定する個所は２か所あります。管理ツールでの『レイヤー設定』画面および API サー

バーの設定ファイル（api_config.ini）になります。

管理ツールの『レイヤー設定』画面では、各レイヤーにアニメーション効果を使うか使わないかを設定することができるようになっていますので必要なレイヤーだけを設定してください。

※ アニメーション効果を使うレイヤーが多ければ多いほどイメージ合成に負荷がかかりますのでご注意してください。アニメーション効果を使うか使わないかは随時変更することができますので負荷状況を確認しながら調整したほうが良いでしょう。

また、API サーバーの設定ファイル（api_config.ini）での設定になりますが、下記２か所の設定を行います。詳しくは「Ⅶ 設定ファイル」をご確認ください。

- _WF_ANIMATION_GIF
- _WF_ANIMATION_WITH_PC_ITEM

３）システムへのライブラリーインストール

導入した AVAFIT バージョン、システム OS バージョンなどにより必要なライブラリーが異なる可能性がありますので、WEBFIT にお問い合わせください。

２．ダイナミック機能の活用

普段、アイテムは１個または２個のレイヤーで構成され、各レイヤーは１つの画像ファイルを持ちますが、１つのアイテムに対して複数の画像ファイルを登録しておいて、アバターを表示する際に状況に合わせて複数の画像ファイルから１つの画像を選んで利用することができれば、よりダイナミックなアバター演出ができます。

例えば、①笑う ②怒る ③泣く ④驚く を表現する４つのアイテムファイルを一つのアイテムとして登録しておいて、占いなどコンテンツの結果に応じてアイテムファイルを自動的に入れ替えてアバターを表示すれば、ユーザーの着替え操作がなくてもアバターが変わるのでより面白いコンテンツ提供が可能になります。

こういった機能を活用するためには

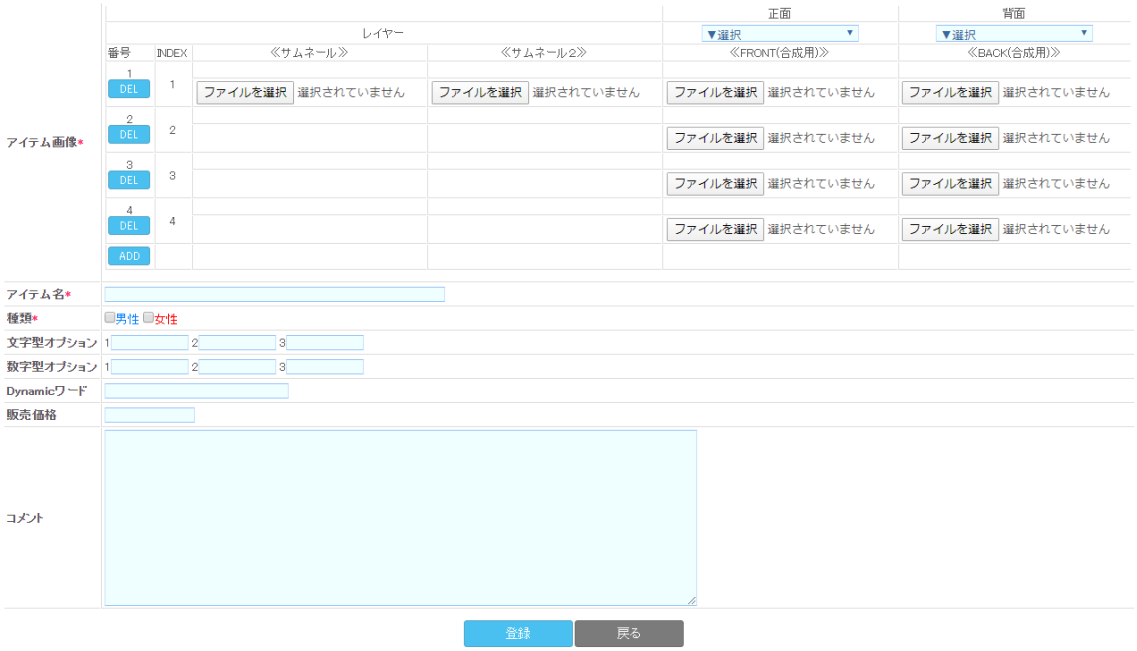
① アイテムを登録する際に複数の画像ファイルを一つのアイテムとして登録する。
② そのアイテムのダイナミックキーワードを決める。
③ いくつかの API 関数を使う際にダイナミック指定をする。

過程が必要になります。ここでは上記過程を詳しく説明します。

１）アイテム登録

管理ツールの「アイテム新規登録」画面に入ります。そこで、一つのアイテムとして登録する画像ファイルの枚数に合わせて「アイテム画像」登録欄を増やしておきます。画面上の「ADD」ボタンをクリックすれば登録欄が増えます。

ファイル記入欄にそれぞれの画像ファイルを指定します。



２）Dynamic キーワード入力

それからは、「Dynamic キーワード」欄にこのアイテムを認識するためのキーワードを入力します。上記例画面では「FACE」と入力しています。

３）API 関数の利用

上記の作業まで終わったら、API 関数を呼び出す際に「dyna_cond」パラメーターにダイナミックキーワードと表示するファイルのインデックス番号を記入するだけでダイナミック機能が使えます。

例）「FACE:3」⇒「FACE」といったダイナミックキーワードを持つアイテムを着用中の場合、
　　3 番目の画像を使ってアバターを作成

ダイナミックキーワードを使う関数は下記の通りです。すなわち、下記の関数のいずれかをコールしないとアバターの画像が自動的に変わることはありません。着替えを行わない場合でも画像を変更したい場合は、「avatar_rebuild」関数を呼び出します。

avatar_rebuild
avatar_item_puton
avatar_item_puton_force
avatar_item_putoff
avatar_item_putoff_force

avatar_item_puton_temp
avatar_item_putoff_temp
avatar_2shot

例）

```
$dyna_cond = "FACE:3";
$return = $AVAFIT->avatar_rebuild("e13579ae01173ae199aea5e8e902ce51",
                                  "hogehoge",
                                  $dyna_cond,
                                  "",
                                  true);
if (!$return["result"]) {
    $errMsg = $return["error"];
}
```

もし、ダイナミック機能を有するアイテムが複数ある場合は下記のように指定することですべてのダイナミックアイテムが機能します。

```
$dyna_cond = "FACE:3, TIME:23, SEASON:4,…";
```

ダイナミック機能を使うのにあたって、どのユーザーが現在ダイナミックアイテムを装着しているかどうかを調べる必要はありません。装着していない場合やダイナミックキーワードが異なる場合は無視されますので、ダイナミックアイテムを装着していないアバターには影響を与えません。

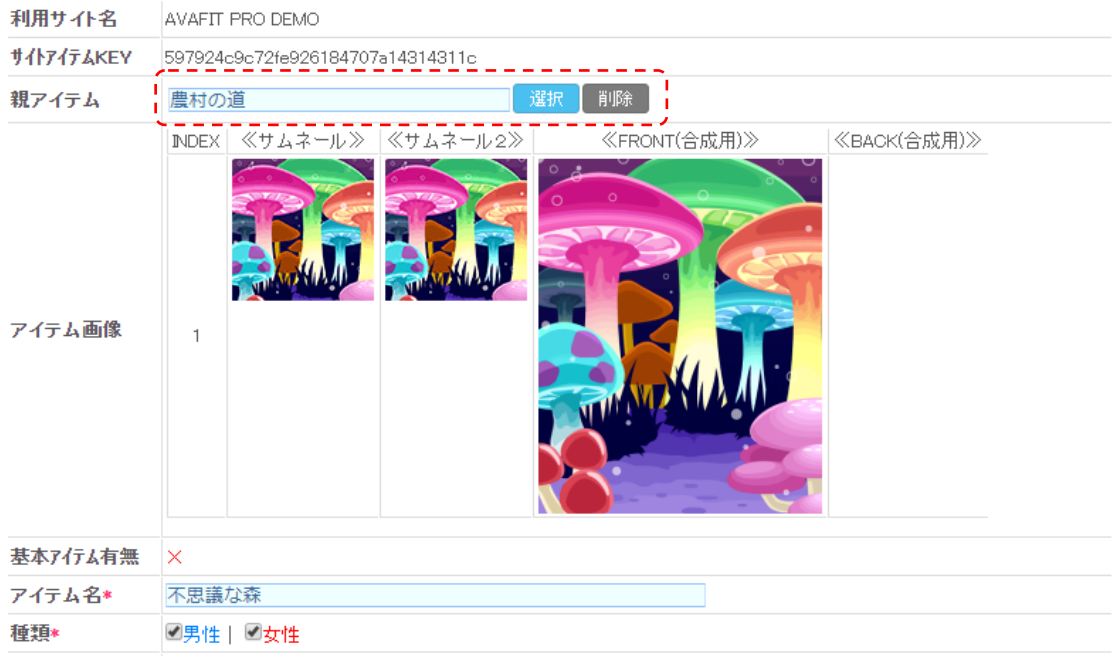
また、「dyna_cond」パラメーターを使う API 関数は、アバターを再作成する必要のある関数だけに限られますので、ご確認してください。

３．アイテムの親子関係利用

最近は、アイテムをユーザーに提供する際に、あるアイテムの詳細画面でそれに類似するアイテム（色違いアイテムなど）を表示し、バリエーション展開を図るような構成がよく見られます。このように、あるアイテムと他のアイテムを関連付け、必要に応じて一緒に表示することができます。

１）アイテムの親子関係設定

「サイトアイテム詳細」画面より、「親アイテム」の項目を設定して親子関係を作ります。多段階親子関係も作れます。



２）API 関数の利用

アイテムの詳細を返すすべてのＡＰＩ関数には、子アイテム数（CHILD_COUNT）が含まれますので、子アイテムがある場合は「item_info_child」関数を呼び出すことで子アイテムの情報も取得できます。つきはその利用例とＵＩサンプルになります。

例）

```
//アイテム情報
$return = $AVAFIT->item_info($site_key, $user_code, $item_key);
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
```

```
else {
    $item_info = $return["value"];
    $items = $item_info["ITEMS"];

    //類似アイテム（色違いアイテムなど）があるか確認
    if ($items[0]["CHILD_COUNT"]) {
        $return = $AVAFIT->item_info_child($site_key,
                                           $user_code,
                                           $item_key,
                                           $page,
                                           $item_view_count);
        if (!$return["result"]) {
            $ERROR[] = $return["error_shot"];
        }
        else {
            $child_info = $return["value"];
            $children = $child_info["ITEMS"];
        }
    }
}
```

■アイテム詳細■

所有ﾎﾟｲﾝﾄ：500P

雪の森
¥250 P

[購入]
[試着のままｼｮｯﾋﾟﾝｸﾞ継続]
[他のｱｲﾃﾑをみる]

※ TOPへ戻る場合は試着の情報が解除されます。

■類似アイテム■

雪うさぎの隠れ野
¥250 P
[詳細を見る]

積雪の丘
¥250 P
[詳細を見る]

1

ｱﾊﾞﾀｰTOPへ

ﾄｯﾌﾟﾍﾟｰｼﾞへ

(c)2009 AVAFIT Pro3 UI Sample

４．サーバー管理のテクニック

１）API サーバー管理

API サーバーには、導入時にインストールした静的ファイルだけが置かれています。システムから新しく作る動的ファイルは全くありませんので、その観点では管理が一番し易いサーバーになります。

API サーバーが複数ある場合、すべての API サーバーは同じファイル構成になりますので、複数ある API サーバーのうち一つのサーバーが壊れた場合は、ユーザーインターフェース及び管理ツール側の設定ファイルより壊れた API サーバーを削除するだけで故障のトラブルより回避できます。

ただ、API サーバーは画像サーバー及びデータベースサーバーに比べて利用頻度が高くなる可能性がありますので、負荷がどれほどかかっているのか常にチェックすることをおすすめ致します。

※API サーバーの負荷対策

一度作成されたアバター及びアルバムの URL 情報は削除しない限り変更されません。アバターの着替えを行った場合は、画像そのものは変更されますが、画像ファイルの URL が変わることはありません。そのため、アバター画像を取得するために毎回 API 関数（avatar_call）を呼び出す必要はありません。一度呼び出して URL を取得した場合は、コンテンツサーバー側にその情報を保存して再利用することで API サーバーの負荷を軽減できます。

また、「api_config.ini」設定ファイルの設定の際にも負荷を考慮しての設定ができます。モバイルサービスだけの場合は、「_WF_SERVICE_PC」を「0」に設定して、各 API 関数が要らない PC 用タグを返さないようにすることで少し負荷が軽減されます。その逆の場合も同様で、PC サービスのみある場合は「_WF_SERVICE_MOBILE」を「0」にしてください。画像の合成処理をしないので軽くなります。

２）画像サーバー管理

画像サーバーには、管理ツールより登録したアイテムファイル及びアバター作成、アルバム作成などで作られたアバターファイルが置かれます。こういった動的ファイルの中には再度作ることができるファイルもありますが、一度なくなったら復旧できないファイルもあります。

● **アイテムファイル**

サーバー故障で無くなったアイテムファイルを復旧することはできません。常にバックアップしておくことをおすすめ致します。アイテム画像ファイルは、管理ツールからアイテムを登録または編集するときに更新され、普段は変更されませんので定期的なバックアップ（例えば、１日１回）をしておけば、万が一故障が発生した場合でも復旧が可能になります。

また、画像サーバーが複数ある場合は、すべての画像サーバーのアイテムファイルが同一ですので故障していないサーバーよりアイテムファイルをコピーしておくことでも復旧可能です。

● **アバターファイル／アルバムファイル**

アイテム画像ファイルと異なって、アバター画像ファイル及びアルバム画像ファイルは複数の画像サーバーがある場合でもそれぞれファイル構成が異なります。一人ユーザーのアバター及びアルバムファイルは一つの画像サーバーに保存されます。そのため、複数ある画像サーバーより１台が故障した場合、残りの画像サーバーより画像ファイルをそのままコピーすることでは復旧できません。ただ、アバター画像ファイル及びアルバム画像ファイルは、アイテムファイルを合成して作られたファイルですので再作成することができます。アバター画像を再作成する場合は、「avatar_rebuild」関数を実行してください。また、API サーバー上の「/batch/avatar_rebuild.php」を実行すれば各サーバーの全員分のアバター及びアルバムを再作成することができます。

● **アバターファイル／アルバムファイルの復元処理**

例）

```
php -f /var/.../avatar_rebuild.php  0df…9ae  1  1000  IMG1  IMG1
```

0df…9ae　サイトキー（32 文字列、管理画面から確認）
1　インデックス番号（復旧開始位置）
1000　復旧アバター数（インデックス 1 より 1000 件）
IMG1　故障している画像サーバーID（1 番目、「api_config.ini」に定義されています。）
IMG1　復旧先画像サーバーID（2 番目、「api_config.ini」に定義されています。）

上記は、画像サーバーIMG1 が故障し、IMG1 サーバーのハードウェアを交換後、再度 IMG1 サーバーにデータを作成する場合の例です。

もし、画像サーバーを 2 台以上使う環境で IMG1 サーバーが故障したため、IMG1 を利用していたアバターデータをすべて IMG2 に移す場合は下記のようになります。

例）

```
php -f /var/.../avatar_rebuild.php  0df…9ae  1  1000  IMG1  IMG2
```

全体ユーザーが 10 万人で IMG サーバーを 3 台使っている場合、1 台はおおよそ 33000 人ほどになるかと思います。この場合は、

```
php -f /var/.../avatar_rebuild.php  0df…9ae  1  100  IMG1  IMG1
```

※まず、100 件ほどを試しに実行して頂き、どれほどのユーザーがいるのか、またどれほど時間がかかるのかをご確認してください。

32000 人（仮定）いることが確認できたら

```
php -f /var/.../avatar_rebuild.php  0df…9ae  1  10000  IMG1  IMG1
php -f /var/.../avatar_rebuild.php  0df…9ae  10001  10000  IMG1  IMG1
php -f /var/.../avatar_rebuild.php  0df…9ae  20001  10000  IMG1  IMG1
php -f /var/.../avatar_rebuild.php  0df…9ae  30001  10000  IMG1  IMG1
```

で IMG1 のすべてのアバターデータが復旧されます。最後のコマンドで「... 30001 2000 IMG1 IMG1」と正確な数字を指定しなくても問題ありません。

３）データベースサーバー管理

データベースサーバーは、すべてのデータ管理を行っており、サーバー故障などでデータベースが壊れた場合は、復旧できなくなる恐れがあります。そのため、データベースサーバーはマルチ構成（レプリケーション構成）することをおすすめ致しますが、マルチ構成をしない場合は、バックアップを取っておく必要があります。

５．その他の注意点

１）退会時の処理

アバターを保有するユーザーがサイトを退会したため、これ以上アバターが要らなくなった場合は、アバターを削除してください。お持ちのサイトから会員データのみを削除した場合は、後でアバターを削除しようとしてもユーザーID が分からないため削除できなくなります。退会したユーザーのアバターをそのまま残すとサーバーに余計な負荷がかかるだけなので必ず実施してください。

```
$return = $AVAFIT->avatar_delete($site_key, $user_code);
if (!$return["result"]) {
    $ERROR[] = $return["error_shot"];
}
```

また、一度削除されたアバターを取り戻すことはできませんので、ご注意してください。また、アバターが削除される場合は、所有アイテム情報、アルバム情報なども同時に削除されます。

２）API 呼び出しエラーログの確認

API 関数の呼び出しに間違いがある場合は、その関数の戻り値にエラーの内容が返されますが、それと同時に「api_config.ini」の「_WF_LOG_SAVE」が「1」に設定されている場合はデータベースにそのエラーログを記録し、管理画面（環境設定⇒ログ確認）からそのログをご確認できるようにしています。API 呼び出しエラーは API の利用間違いであり、余計なサーバー負荷になるだけですので、内容を確認して API 呼び出しエラーが発生しないようにロジックを見直して頂く必要があります。

また、ログは自動的に削除されません。ある程度ログが貯まっている場合は、ログの削除（環境設定⇒オプション設定）を行ってください。