Web Intents

W3C Editor's Draft 11 September 2012

This version:
https://dvcs.w3.org/hg/web-intents/raw-file/tip/spec/Overview-respec.html
Latest published version:
http://www.w3.org/TR/web-intents/
Latest editor's draft:
https://dvcs.w3.org/hg/web-intents/raw-file/tip/spec/Overview-respec.html
Editors:
Greg Billock, Google
James Hawkins, Google
Paul Kinlan, Google

Translators:
Yoichiro Tanaka, mixi, 3rd October, 2012
Source: http://dvcs.w3.org/hg/web-intents/raw-file/tip/spec/Overview.html

---

概要

この仕様は、Web Intentsと呼ばれるWebアプリケーションのための、サービスの発見と軽量なRPCメカニズムを定義します。
この文書は、Web Intentsメッセージを作成、受取、そして返信するためにクライアントおよびサービスのページによって使われるDOMインタフェースとマークアップや、そのプロセスを容易に実施するための手続きを定義します。

この文書のステータス

このセクションは、その公開の時間でのこの文書のステータスを説明します。他の文書は、この文章に優先するかもしれません。現在の*W3C*出版物やこの技術的なレポートの最新のバージョンは、*http://www.w3.org/TR/*の*W3C technical reports index*で見つけることができます。
この文書は、Web Applications (WebApps) Working Groupによって、Editor's Draftとして公開されました。もしあなたがこの文書に関するコメントをしたい場合は、public-web-intents@w3.org

(subscribe, archives)にそれらを送ってください。全てのフィードバックを歓迎します。
Editor's Draftとしての公開は、W3C会員による保証を暗示するものではありません。これはドラフト文書であり、いつでも更新や他の文書に置き換えや廃止がされるかもしれません。進行中の作業以外でこの文書を引用することは適切ではありません。
この文書は、5 February 2004 W3C Patent Policyの下でのグループ作業によって作られました。W3Cは、そのグループの提出物に関して作られたあらゆる特許情報開示の公開リストを保守します。そのページはまた、特許情報の開示のための指導も含みます。Essential Claim(s)を含み個人が信じる特許の実際の知識を有するその個人は、section 6 of the W3C Patent Policyに従って情報を開示しなければなりません。

目次

導入

Web Intentsは、Webアプリケーション間をリッチに統合することを可能とします。Web上で利用可能なサービスは、それらがそれらの仕事をす際に、あちこちにリッチなデータを渡す必要がますます出てきています。Web Intentsは、Webの先進性として実績のある祖結合やオープンアーキテクチャの類を維持しながら、この交換を容易にします。それらは純粋にクライアントサイドに存在し、ユーザエージェントを通じて仲介され、交換されるデータのセキュリティやプライバシーの制御をユーザに最大限許可します。
インテントは、サービスによって実行されるための権限を委譲されたユーザによって起こされるアクションです。それは、ユーザが実行されることを期待するアクティビティ（例：“share”や”edit”）の種類を示しサービスに伝えられる”action”文字列、サービスが期待するはずのデータペイロードを示す”type”文字列、そしてデータペイロードそれ自体から構成されます。
インテントのライフサイクルは、最初はクライアントがハンドルされるインテントを要求します。このインテントデータは、その時ユーザエージェントに渡され、ユーザは多くの潜在的に利用可能なサービスを選択することができます。その後サービスはそのインテントデータを渡され、インテントにおいて特定されるアクションを実行するためのUIがユーザエージェントによって提供されます。最後に、サービスはまたクライアントにアウトプットとしてデータを返すかもしれません。
Web Intentsは、サービスにハンドルするインテントを一覧化することを許可する宣言的な書式を提供します。この手法を使って、サービスはハンドル可能なアクションや期待するデータをマークアップします。

例

写真ホスティングアプリケーションを考えます。このアプリケーションは、共有された画像を選択し、それらの画像を編集し、さらに友人にそれらを共有することをユーザに提供します。そのアプリケーションは、ユーザに利用可能な写真の作成を中心に構築されていますが、編集や共有のインタフェースがビルトインされていません。しかし、添付のようなコードによって、各写真の横に**Edit**ボタンを配置することが可能です。

Example 1

```
document.getElementById('edit-photo').addEventListener("click", function() {
  var intent = new Intent({"action":"http://webintents.org/edit",
                           "type":"image/jpeg",
```

```
                              "data":getImageDataBlob(...)});
  navigator.startActivity(intent, imageEdited);
}, false);

function imageEdited(data) {
  document.getElementById('image').src = data;
}
```

このコードは、BLOBとして準備される画像を消費することができるサードパーティアプリケーションに画像の編集を委譲し、同じフォーマットにて結果を提供します。具体的には、あるそのようなエディタは、memeメーカー（ユーザが選択した写真上にユーモアなメッセージを置くことができるアプリケーション）になるかもしれません。
今、写真は選択されたサービスで編集され、mameテキストが追加され、ユーザは明らかに友人に結果を共有したがっています。繰り返しになりますが、写真ホスティングアプリケーションは、共有の機能を有していません。しかし、画像の近くに**Share**ボタンを追加することによって、そして添付のようなコードによって、この統合は達成できます：

Example 2

```
document.getElementById('share-photo').addEventListener("click", function() {
  var intent = new Intent({"action":"http://webintents.org/share",
                           "type":"text/uri-list",
                           "data":getPublicURIForImage(...)});
  navigator.startActivity(intent, imageEdited);
}, false);
```

このコードは、ユーザによって選択されたURLの共有ができる既存のサービスに、共有の機能を委譲します。(getPublicURIForImage()は、共有されるURLを得るアプリケーション特有の機能の一部のためのマーカーです) そしてユーザによって選択されたソーシャルネットワーキングサービスは、画像のサムネイルを伴ったステータス更新を生み出すでしょう。ブログサイトであれば、ユーザに写真を投稿するためのUIが提供されるでしょう。
この統合で注意することは、その他のより高いマインドのサービスも、ユーザにより選択されることが可能であるということです。奇妙な題目を追加するためのサービスを使う代わりに、ユーザは露出の調整、赤目の削除、または任意の数の画像に対する他の変換をするための洗練された写真編集アプリケーションを利用するかもしれません。ユーザは登録された多くのそのようなツールを持つことができ、そして与えられたいかなる画像の編集作業で使用する任意のセットを選択できます。その写真ホスティングアプリケーションは、ユーザが選択するアプリケーションを制御しません。それらのタスクを実施するために、それらが必要とするデータを提供することによって、そのようなアプリケーションは祖結合になり、そしてユーザがそのデータ上でそれらのアクティビティを起動することを実現します。
この方法において、インテントは二重のハイパーリンクのようです。ハイパーリンクでは、元のページは遷移先を正確なURLで指定します。インテントの場合、元のページは行われるタスクの種類を指定し、そしてユーザはタスクを完了するために使われる任意の数のアプリケーションから選択することができます。

サービス側では、そのページはWeb Intentsサービスとしてそれ自身の登録が必要であり、そして入ってくるインテントのデータをハンドルし、可能であれば結果を提供します。それは以下のようなコードで行われます:

Example 3

```
<html>
<head>
<title>Image Meme Editor</title>
</head>
<body>
<intent action="http://webintents.org/edit"
type="text/uri-list;type=image/*,image/*"></intent>
<script>
  window.addEventListener("load", function() {
    if (window.intent) {
      setImageContentURI(window.intent.data);
    }
  }, false);

  document.getElementById('save-button').addEventListsner("click", function()
{
    window.intent.postResult(getImageDataURI(...));
  }, false);
</script>
```

ここで仮定された例(クライアントおよびサーバページの両方)は、アプリケーションの画像表示を扱う機能であり、具体的に画像をキャンバスの中に配置してデータのURIとして再びそれを取り出します。また、そのサイト上で画像の公開URIを提供します。

標準的なパート

この仕様の標準的なパートは、APIの説明およびユーザエージェントの振るまいに関するセクションです。他の全てのパートは、参考情報となります。

専門用語

インテントは、データを伴うアクションであり、ユーザが選択したサービスによって実行されます。

アクター

インテントを生成してそれを伴ってユーザエージェントを呼び出すWebページはクライアントです。
ユーザエージェントはまた、Webページではない元がインテントを呼び出すことを許可します。具体的には、ユーザエージェントはインテントの配送を呼び出すUIを持つかもしれませんし、インテントをトリガーとして外部のOSのフックを登録するかもしれません。
インテントをハンドルすることができるWebページはサービスであり、もしかしたら呼び出したクライア

ントページへデータの一部を返します。(ユーザエージェントはWebページではないサービスへもインテントを配送する方法を持つかもしれません。それらは拡張API、プラグインヘルパー、外部OSのハンドラーなどかもしれません。)

インテントおよびサービスのライフサイクル

登録は、Webページ(またはもう一つの同一起源サービスページ)がどのようにインテントを受け取る機能を持つユーザエージェントに告げるかを表します。
呼び出しは、ハンドリングのためにクライアントページがインテントを配送するAPIを指します。
選択は、ユーザエージェントがどのサービスが特定のインテントを受け取るかを決定するメカニズムです。
配送は、ハンドリングのためにユーザエージェントがサービスページにインテントを渡すことを意味します。
返答は、ユーザエージェントを通じてクライアントページへデータを返送することによってサービスがインテントを返すことを意味します。

特定のインテント呼び出しの手順は非同期となります。サービスはインテント配送を受け取り、そして分離された実行コンテキスト内でその返答を準備します。その返答は、その際に非同期コールバックで呼び出したクライアントに返されます。

APIの説明

インテントのパラメータディクショナリ

このオブジェクトはインテントオブジェクトのオブジェクトリテラルにて使われます。そのオブジェクトリテラルのコンストラクターを使用する際に、**action**および**type**フィールドが必要となり、その他は任意です。

WebIDL

```
dictionary IntentParameters {
  DOMString action;
  DOMString type;
  any? data;
  sequence<Transferable>? transfer;
  URL? service;
  sequence<URL>? suggestions;
}
```

ディクショナリ**IntentParameters**メンバ

action - DOMString型
インテントのアクションタイプを示す任意の文字列です。この文字列は空文字であってはなり

ません。

data - any型
使われるデータペイロードは、Transferableを含む構造化された複製アルゴリズムが実行されることが可能なオブジェクトでなければなりません。

service - URL型
指定された場合、このフィールドは明示的な意志としてそのインテントをマークします。その値は絶対URLでなければなりません。

suggestions - sequence<URL>型
指定された場合、このフィールドはクライアントが知っていてそのインテントを受取可能である（絶対的な）提案されるサービスのURLのリストを提供します。

transfer - sequence<Transfarable>型
構造化された複製アルゴリズムの利用のためのTransfarableの一覧です。

type - DOMString型
データペイロードの型を示す文字列です。データペイロードはこのパラメータで説明されなければならず、空文字ではなりません。

インテントオブジェクト

インテントオブジェクトは、サービスによって受け取られるために要求されうる特定のタスクを形作ります。クライアントページは、同時に複数のインテントを呼び出すかもしれません。特定のインテントオブジェクトは、一度作られたら不変です。

WebIDL

```
[Constructor(IntentParameters params),
Constructor(DOMString action, DOMString type, optional any data,
optional sequence<Transferable> transferList)]
interface Intent {
    readonly attribute DOMString     action;
    readonly attribute DOMString     type;
    readonly attribute any           data;
    readonly attribute MessagePort[] ports;
    void postResult (any data, optional sequence<Transferable>
transferable);
    void postFailure (any data);
};
```

属性

action - DOMString型、読み取り専用
これは任意の文字列です。選択された文字列はURL名前空間規約によって名前空間が指定されるべきです。この文字列は空文字であってはならず、そうでなければそのコンストラクタは例外を投げなければなりません。

data - any型、読み取り専用
使われるオブジェクトは、Transfarableを含む構造化された複製アルゴリズムが実行できる

オブジェクトでなければならず、そうでなければコンストラクタは例外を投げなければなりません。

ports - MessagePortの配列型、読み取り専用

インテントオブジェクトがサービスページへ配送される時のみ指定します。呼び出しの間コンストラクタのtransferList内で使われるいかなるポートは、portsにおけるサービスページへ配送されるでしょう。[POSTMSG]をご覧ください。

type - DOMString型、読み取り専用

データペイロードは、typeパラメータによって説明されなければなりません。推奨されるtype文字列は、MIME文字列もしくは自身で文書化されるURLとなります。文字列は空文字であってはならず、そうでなければコンストラクタは例外を投げなければいけません。

インテントオブジェクトの注意: transferListパラメータは、生成後は利用することができません。それは、Transferableとしてdata引数のいくつかのフィールドを解釈するための、ユーザエージェントへの内部実装詳細の指示になります。dataおよびtransferList引数は、W3C Web Messaging仕様[POSTMSG]に従って実装されなければなりません。
ユーザエージェントは、インテントオブジェクトの生成において、構造化された複製アルゴリズムを実行しなければなりません。

呼び出しAPI

クライアントは、上記のインテントオブジェクトの生成およびそれを使ってnavigator.startActivity関数をコールすることでインテントを呼び出します。このメソッドに渡されるコールバックは、インテントがサービスによって受け取られたときに呼び出されます。ユーザエージェントは、要求されたインテントのアクションや型にマッチする登録されたサービスの一覧を列挙することで、サービス選択を仲介するでしょう。その時ユーザは、そのインテントを受け取るべきサービスの選択が可能になります。

WebIDL

```
[NoInterfaceObject]
interface Intents {
  void startActivity (Intent intent, optional IntentSuccessCallback
onSuccess, optional optional IntentFailureCallback onFailure);
};
```

メソッド

startActivity

インテントがサービスを呼び出すためにコールされます。インテントオブジェクトは上記で説明しました。onSuccessハンドラは、もし指定されれば、サービスが受け取り、インテントを処理し、そしてそれが受け取るインテントオブジェクトにおけるpostResultを呼び出した場合に、ユーザエージェントによって呼び出されるでしょう。onFailureハンドラは、もし指定されれば、サービスが受け取れなかった、ユーザが処理の選択を中止した、またはサービスが選択され、インテントが受け取られ、インテントを処理し、そしてそれが受け取るインテントオブジェクトにおいてpostFailureが呼ばれた場合に、ユーザエージェントによって呼び出されるでしょ

う。ハンドラは、一つのパラメータ（サービスから受信したデータ）を伴って呼び出されます。ユーザエージェントは、ユーザの明示的なジェスチャのコンテキスト内でのみ、このメソッドの呼び出しが正常に行われるように制限すべきです。インテントが無効（例：null）、またはメソッドがユーザの明示的なジェスチャの結果ではなく呼ばれた場合は、例外が投げられるべきです。

パラメータ	型	Null可能？	任意指定？	説明
intent	Intent	×	×	
onSuccess	IntentSuccessCallback	×	○	
onFailure	optional IntentFailureCallback	×	○	

返却値の型：void

この呼び出しAPIは、window.navigatorオブジェクトによって実装されます。

WebIDL

```
Navigator implements Intents;
```

Navigator型の全てのインスタンスは、Intentsインタフェースも実装することが定義されます。
startActivityに渡されるコールバックは、これらのシグネチャを提供しなければなりません。

WebIDL

```
callback IntentSuccessCallback = void (optional any data, optional
MessagePort[] ports);
```

Callback IntentSuccessCallbackのパラメータ

data - **any**型
渡されたdataは、配送されたインテントのpostResultメソッドに渡された構造化された複製可能なオブジェクトからペイロードデータになるでしょう。
ports - **MessagePort**の配列型
portsは、Web Messaging仕様によって、Transferableを使ったよる構造化された複製アルゴリズムの結果として、ペイロードにおいて任意のポートになるでしょう。[POSTMSG]

WebIDL

```
callback IntentFailureCallback = void (optional any data)
```

Callback IntentFailureCallbackのパラメータ

data - any型
　　渡されたdataは、配送されたインテントのpostFailureメソッドに渡されるペイロードデータになります。

配送、返答API

ユーザエージェントはインテント呼び出しを受け取るためにサービスページを読み込む際に、それはページのスコープ内でwindow.intentオブジェクトを配置します。

WebIDL

```
[NoInterfaceObject]
interface IntentProvider {
  readonly attribute Intent intent;
};
```

属性

intent - Intent型、読み取り専用
　　サービスページに配送されるインテントオブジェクト（postResultおよびpostFailureメソッドを含む）

WebIDL

```
Window implements IntentProvider;
```

Window型の全てのインスタンスは、IntentProviderインタフェースの実装としても定義されます。このオブジェクトは、それらがインテント呼び出しのコンテキスト内で読み込まれた際にのみサービスページで利用可能となるでしょう。同じURLが他の状況で読み込まれた場合には、ユーザエージェントはWindowにIntentProviderを実装させてはなりません。同様に、もしサービスページが登録されるが、ユーザエージェントがそれを読み込み、そしてユーザエージェントが配送しているインテントを受け取るために期待されるそれに印をつける宣言的なマークアップを含まない場合は、ユーザエージェントはIntentProviderをWindowに実装してはいけません。
window.intentオブジェクトは、サービスページの同一生成元リダイレクトを超えて利用可能にはなりません。
つまり、http://example.com/serviceからhttp://example.com/loginに移り、http://example.com/serviceに戻るリダイレクトシーケンスは機能するでしょう。この場合において、window.intentデータは3つ全てのページで読み込むことができます。
http://example.com/serviceからhttp://example.com/newserviceに行くのも同様に動作するでしょう。window.intentデータはどちらのページでも利用可能です。
http://example.com/serviceからhttp://login.example.comに行き、そしてhttp://example.com/serviceに戻るシーケンスにおいて、window.intentはexample.comのページで

のみ利用可能です。インテントのデータはhttp://login.example.comには提供されません。
もしユーザがサービスページが表示されたブラウザコンテキストに遷移し始めた場合、そして新しいページがユーザエージェントが配送しているインテント型を受け取るための宣言的マーク書式を持つ場合、window.intentオブジェクトは、最初に配送したページと同じ生成元のページで利用可能となる必要があります。
言い換えると、インテントが最初に配送されるブラウジングのコンテキストにおいて、インテントのデータは、リダイレクトや遷移シーケンスが、インテントを最初に配送した（そして可能なマークアップを持つ）同じ生成元である時に、それらのページにて利用可能にならなければならず、逆にいかなる他のページにおいて利用可能であってはなりません。これはサービスページにハンドリングの委譲、ログイン画面への遷移やその他ことをできるようにしますが、インテントのデータを出くわした同一の生成元ではないいかなるそのようなページで利用可能にはさせません。
サービスページ内のコードによる複数の呼び出しは、JavaScript例外を投げなければなりません。
ユーザエージェントは特定のインテントに対して複数回応答することは許可されません。

登録マークアップ

Webページ（もしくは他の同一起源のサービスページ）は、<intent>タグを使って、特定のインテントのアクションや型のために、機能的に受け取ることの提示として、宣言的にそれら自身をマークアップします。ユーザエージェントは、配送されるインテントに対して何のインテントがマッチして受け取ることができるかを説明したマークアップを含まないWebアプリケーションサービスページにインテントを配送してはなりません（配送のためのアクションや型のマッチングについてのセクションを参照ください）。

WebIDL

```
interface HTMLIntentElement : HTMLElement {
  attribute DOMString action;
  attribute DOMString type;
  attribute DOMString href;
  attribute DOMString title;
  attribute DOMString disposition;
};
```

属性

action - DOMString型
サービスがサポートする振る舞いの種類を示す任意の文字列です。それはユーザに完全修飾されたURIを使って勧められます。もしaction属性が未指定の場合、インテントのサービスはtype属性において提供された型の表示を処理することを仮定します。

disposition - DOMString型
disposition属性は、サービスに開かれるべきコンテキストの選択を可能にします。ユーザエージェントは、クライアントページにdispositionの変更をするいかなる能力も与えてはなりません。”window” dispositionは、サービスが新しいタブやウィンドウのコンテキストで開かれ

ることを意味します。”inline” dispositionは、ユーザエージェントがオーバーラップ手法でクライアントページのコンテキストに直接関連したコンテキスト内でサービスを開くべきであることを意味します。ユーザエージェントは、このUI面をクライアントページの制御下にしてはなりません。

href - DOMString型

完全修飾されたURIであるべきです。もしhref属性が未指定の場合、サービスURIはタグで見つけられるURIがセットされるでしょう。

title - DOMString型

ユーザエージェントが、ユーザにサービスを表示するために使うべき人間が読むことができるタイトルです。もしtitle属性が未指定の場合、ユーザエージェントはサービスのタイトルとして登録されたサービスページのタイトルを使います。

type - DOMString型

サービスが受け取ることができるペイロードデータの型を特定する文字列です。型識別子の空白区切りのリストである必要があります。これらがMIME型また”*/”や”*”によるMIMEワイルドカード型としてパースする場合、それらはそのように解釈されるでしょう（配送のためのアクションや型のマッチングに関するセクションを参照）。もしそうでない場合、それらは文字列リテラルの型識別子として解釈されるでしょう。type属性が空文字の場合、登録は有効ではありません。

このセクションは、*Web*サービスではない登録を管理するためにユーザエージェントができることが限定されると解釈するべきではありません。つまり、ユーザエージェントはローカルで、拡張機能を使って、もしくはサードパーティシステムのサービスを使ってインテント呼び出しを受け取るかもしれないのです。このセクションは単に*Web*アプリケーションサービスがどのように<*intent*>タグを使ってユーザエージェントを登録および登録解除するかを述べているに過ぎません。

同一生成元登録

ユーザエージェントは、同一生成元ではない全ての登録マークアップについて従ってはなりません。つまり、ページはそれと同じ生成元における他のサービスハンドラをhref属性で登録をするのみです。ページは、空のhref属性を残すことにより、サービスハンドラとしてそれ自身を登録することもできます。

クロスページ登録のために、href属性が異なるリソースを指す時、ユーザエージェントはインテント登録を追加的に解釈すべきです。つまり、タグからのインテントディスクリプタは、もしそれが指定されなければ、（もしユーザが追加を承認すれば）レジストリに登録されるでしょう。

登録解除

ユーザエージェントは、マークアップがhrefページでintent機能を指定しなかったように、actionおよびtype属性の両方がない如何なる登録マークアップも取り扱うべきです。つまり、ページは、全てのインテントタグの削除、またはactionやtype属性なしでタグの指定を明示的に維持することにより、暗黙的にそれ自身を登録解除できます。そのような明示的な登録解除は、同一生成元のhref属性を使った任意のタグに栄誉を与えるべきです。

サービスを受け取るページ自身のインテントタグは、信頼の元にユーザエージェントによって解釈されるべきです。つまり、もしユーザエージェントがサービスページを読み込み、そのインテントのディスクリプタがそのページのために登録された現在のセットにマッチしないとわかったとき、現在登録され

ているセットが、ページの現在の読み込みにおいて発見されるセットを使って置き換えられるでしょう。

サービスページの**HTTP**ステータスコード
もし登録されたサービスのページが20xではないエラーコード[HTTP11]を得る時は、ユーザエージェントは以下のアクションを取るべきです。
30x: 指示されたページにリダイレクトします。もしページがドメイン上にあり、インテントのマークアップを持つ場合は、そのインテントはリダイレクトページで利用することができるでしょう。もしリダイレクトが永続的（301）の場合、ユーザエージェントはそれを反映するためにそのサービス登録の記録を更新するかもしれません。
4xx: 任意の改善手法と共に（例：403の認証や402の課金）、ページが読み込まれなかったことをユーザに示します。410エラーの場合、ユーザエージェントはその内部レジストリからハンドラを登録解除すべきです。
50x: ページを読み込めなかったことをユーザに示します。
もし要求されて登録されたサービスページが読み込むことができなかった場合、ユーザエージェントは他のサービスをユーザに選択させるためのUIを表示すべきです。

ユーザエージェントの振る舞い

サービス登録

ユーザエージェントがマークアップの登録を持つページを読み込む時、Webインテントサービスとしてそのページを設定することをユーザに提供します。このプロセスの詳細は、ユーザエージェントに委ねられます。そのモデルは、ページがインテントを受け取るための能力を助言し、ユーザエージェントはそれを覚えます。ユーザエージェントは、ユーザがそれを許可する特定のアクションを行う前に、インテントをこの方法の中で発見されたサービスへ配送してはなりません。
ユーザエージェントは、列挙でも的確な問い合わせでも、特定のインテント、または任意のインテントを受け取るために設定したサービスをユーザが素直に発見できる機能をWebページに提供してはなりません。ユーザのためにWebページへこの情報を積極的に与える機構があるかもしれませんが、それを素直に利用可能にしてはなりません。
ユーザエージェントは、Webインテントサービスに追加のメカニズムを提供するかもしれません。例えば、外部アプリケーションによって、分離されたAPIを通じて、ダウンロードされたWebアプリケーション内で権限をバンドルした結果として、または予めバンドルされた場合です。（TODO: ローカルネットワークサービスの例を追加）
ユーザエージェントは、クライアントまたはサービスとして機能するかもしれません。例えば、ユーザエージェントは、登録されたサービスによって受け取られるインテントを直接起動する特定のアフォーダンスを実装するかもしれませんし、あるいはインテントを受け取るための他の登録されたサービスと同時に使われる自自身の機能を可能にするUIを提供するかもしれません。具体的には、ユーザエージェントはページ内のMicrodataから由来するデータ指向の制御に基づいて、または他のユーザエージェントレベルの機能に基づいて、インテントを直接配送することもあるかもしれません。

呼び出しと配送

クライアントWebアプリケーションによって呼び出されたインテントのために、ユーザエージェントは、ユーザの操作のコンテキスト内でそのような呼び出しを要求しなければなりません。ユーザエージェントはまた、他の機構を通じて呼び出されたインテントを配送するかもしれません。例えば、ハードウェアイベント（例：USBストレージデバイスの差し込み）やソフトウェアイベント（例：ダウンロードしているファイル）です。

クライアントページがインテントを呼び出したとき、ユーザエージェントはそれを選択されたサービスに配送します。この処理の詳細は、ユーザエージェントに委ねられています。ユーザエージェントは、インテントをWebアプリケーションサービス、ヘルパーアプリケーションに配送、または他のハードウェアへの接続を通じてそれらをプロキシするかもしれません。けれども、一般的に、ユーザエージェントはどのサービスにインテントを配送するかをユーザに設定させるための方法を提供しなければなりません。このプロセスは呼び出しごとの様式で設定されるべきです。初期ルールは、それらがユーザによって設定される限り、ユーザエージェントが各呼び出しを制御するフルのUIを提供せずに終わることが期待されます。

ページ上のスクリプトが、それらが読み込むインテントデータを処理できるようにするために、ユーザエージェントがインテントペイロードをWebアプリケーションに配送する時、ドキュメントが読み込まれてパースされた際にwindow.intentオブジェクトを利用可能にしなければなりません。ユーザエージェントは、インテントハンドラとしてそれら自身が宣言する登録メタデータを持たないページのスコープにてwindow.intentを利用可能にしてはなりません。これは、Webインテントハンドラとしてそれ自身を明示的に宣言していないページにおけるwindow.intentのいかなる利用も、ユーザエージェントによって書き換えられてはならないということを意味します。それはまた、サービスページが、インテントを登録するマークアップをページがパースする前に実行されるスクリプトにてwindow.intentにアクセスできないことも意味しています。もしそのようなスクリプトが後で読み出される関数を単に宣言していた場合は、それは動作するでしょう。しかし、登録マークアップがパースされる前に実行されるスクリプトは、ページの読み込み後は利用可能になるかもしれませんが、利用可能なwindow.intentを見つけられず、実行はエラーとなるでしょう。

新しいコンテキストがサービスページで開かれた時、ユーザエージェントは、navigator.startActivity呼び出しの実行においてユーザエージェントが受け取った登録されたハンドラに直列化されたペイロードを返すために、window.intentオブジェクトのpostResultおよびpostFailureメソッドを接続しなければなりません。もしユーザがサービスページが返される前にそれを閉じた場合は、ユーザエージェントはもしonFailureコールバックが指定されていればクライアントページ呼び出しにおいてそのコールバックを呼び出さなければなりません。もしユーザがインテントの配送の流れにおいてユーザエージェントが表示したサービス選択のためのUIコントロールをキャンセルした場合、ユーザエージェントはもしonFailuteコールバックが指定されていればクライアントページ呼び出しにおいてそのコールバックを呼び出さなければなりません。

ユーザエージェントは、任意の直列化されたオブジェクトやTransfarableオブジェクトをクライアントからサービスに、そしてサービスからクライアントに戻すことができるべきです。これは、Blob[BLOB]やMessagePortなどを含みます。ユーザエージェントは、インテントのペイロードを調査し、よく知られたインテント型を考慮して特別なUIを提供するかもしれません。例として、ユーザエージェントは、インテントが"share"タイプである、またはクライアントが画像を選択することをそれらに尋ねることを示す特殊化されたメッセージングをユーザに提供するかもしれません。

もしユーザがインテントの特定のタイプのために登録されたサービスを持っていなかった場合、ユーザがアクティビティを完了できるために、ユーザエージェントはそのインテントのタイプを受け取ることが可能であることを知っているサービスについてのデータの他のソースからオプションを表示するかもしれません。
ユーザエージェントは、知られていないインテント型の配送を断定的に禁止してはなりません。これは、例えば望まれないインテント呼び出し、攻撃ベクトルとして使われるインテント、そして他の間違った利用の抑制など、インテントに関してフィルタされる関数を実行するユーザエージェントを禁止することを意味しません。

明示的なインテント

明示的にマークされた（つまり、空ではない**service**フィールドを持つオブジェクトリテラルコンストラクタを使って生成された）インテントをハンドリングする時、期待されるユーザエージェントの振る舞いは、もしこの”service”属性が指定される場合、ユーザにサービス選択機構を表示すべきではありません。その代わり、サービスURLがインテントを受け取るために直接読み込まれるべきです。（これは強い制約ではありません。そのユーザエージェントは、ユーザに明確な呼び出しさえも割り込む方法を提供するかもしれません。）
配送の間、全ての制約はまだ適切な場所にあります。つまり、ユーザエージェントはページのスコープ内でwindow.intentオブジェクトの配置において上記の要求に従わなければならず、サービスページにおいて任意の宣言的メータデータに配慮しなければなりません。もしユーザエージェントがそのUIの準備において使用されるサービスについての情報（例：disposiiton、titleなど）を必要とする場合、それはサービスURLを読み込み、任意の宣言的メータデータからページを調べるかもしれません。
ユーザエージェントは、ちょうどページの任意の他の訪問のように、ユーザにこのサービスをインストールしたいかどうかを尋ねるかもしれません。

配送のためのアクションと型のマッチング

インテントが配送される時、ユーザエージェントは、インテントが配送されるサービス（）が、以下のこれらのステップによって、配送されるインテントのサポートしているアクションおよび型の登録の記録を検査しなければなりません：

1. インテントアクションを、呼び出されるインテントのactionフィールドとします。
2. インテントの型を、呼び出されるインテントのtypeフィールドとします。
3. 全ての区別可能な登録（つまり、ユニークなアクション/型フィールドを持つ）の指定のために、これらのステップに従います：
    a. フィールドの値を比較する時、”異なる”とは、比較される文字列はコードポイントの序列が異なることを意味します。
    b. フィールドの値がMIMEタイプかどうかを考える時、それは有効なMIMEタイプ[RFC2046]としてパースするか、またはMIMEワイルドカード文字列である”*”や”*/*”と等しいかが考慮されます。
    c. サービスのアクションを、サービス登録のactionフィールドの値とします。
    d. サービスの型を、サービス登録のtypeフィールドの値とします。
    e. もしサービスの型やサービスのアクションが空だった場合、その登録のレコードは無

効です。次の登録のレコードに進みます。
   f. もしインテントのアクションがサービスのアクションと異なる場合、次の登録のレコードに進みます。
   g. もしインテントの型がMIMEタイプであり、そしてサービスの型がそうでなかった場合は、次の登録のレコードに進みます。
   h. もしサービスの型がMIMEタイプであり、そしてインテントの型がそうでなかった場合は、次の登録のレコードに進みます。
   i. もしサービスの型およびアクションの型の両方がMIMEタイプである場合、そのときそのMIME識別子のオーバーラップの有無をチェックします。これは、トップレベルやサブレベルの型がまさにマッチする場合、またはーつあるいは両方がMIMEワイルドカード(”*”)によって表現される場合にtrueとなります。もし任意のMIMEパラメータがサービスの型またはアクションの型内で指定される場合、それらは両方指定され、そしてまさに一致しなければなりません。あるーつのみのサービスの型やアクションの型でのMIMEパラメータの指定は、一致とは見なされません。もしMIMEタイプがオーバーラップしない場合、次の登録のレコードに進みます。
   j. もしサービスの型もアクションの型もMIMEタイプではない場合、その時もしそれらが異なる場合、次の登録のレコードに進みます。
4. もし満足させるー致がサービスページの登録内で見つからなかった場合、インテントは配送されてはなりません。もし任意の満足させるー致が見つかった場合は、インテントはサービスページに配送されなければなりません。

インテント呼び出しからのサービス提案のハンドリング

もしユーザがユーザエージェントに登録された特定のインテントのための適正なサービスについての永続的な情報を持っていない場合、ユーザエージェントはユーザに呼び出したクライアントによって提案されるデフォルトのサービスから選択するオプションを提供すべきです(提案のパラメータより)。もしユーザエージェントがUIの準備で用いる提案されるサービスの全てあるいは任意の情報(例:disposition、title、iconなど)を必要とする場合、それは提案されたデフォルトのサービスURLを読み込み、そして<intent>タグやそのような情報を読み込んでページを調べるかもしれませんし、またはそのサイトからfaviconを読み込むかもしれません。ユーザエージェントは、もし全てあるいは任意の提案されるサービスをインストールを望む場合、ちょうどそれらのページの任意の他の訪問のためののうに、ユーザに尋ねるかもしれません。
ユーザエージェントは、ちょうど任意のインテントの配送のように、提案されるサービスへインテントを配送する前に、”配送でのアクションおよび型のマッチング”セクションのマッチングアルゴリズムに従わなければなりません。

ユースケースと要求

共有

Web Intentsは、ユーザがWebページ上でコンテンツに遭遇したときに、ユーザがコンテンツを共有できるようにして便利になるべきです。ユーザエージェントによって利用可能になった暗黙的な制御で

も、Webページ内で置き換えられた明示的な制御でも、Web Intentsはページ全体、あるいはページの特定のコンテンツを共有することを望むユーザのユースケースをハンドルすべきです。ユーザはこのタスクを達成するために共有アプリケーションを選択できるべきです。

ローカル**Web**アプリケーションの統合

ローカルWebアプリケーションは、インテントの呼び出しと受け取りができるべきです。

永続的な接続

Web Intentsの呼び出しは、RPCでモデル化されていますが、永続的な接続が求められる時があります。Web Intentsがこれをサポートするいくつかの異なった方法があります。一つは、iframe内でクライアントによって読み込み可能なURIを返して、他のWebプラットフォームの機能を使って伝達することです。もう一つは、知られているターゲットに直接他のインテントを対象にクライアントが使うことができるデフォルトトークンを返すことです。3つ目は、クライアントが後続の通信で使うことができるMessagePortを返すことです。
これらのケースでは、Web Intentsは特定のクライアントページを永続的に利用可能なサービスにユーザを接続するための方法として機能します。例えば、接続が永続的か一時的か、トークンがユーザにより消費されるか不透明なのか、どのようにこれが機能すべきかをまさに説明するのは、インテントの特定の型に委ねられています。

外部のアプリケーションとの統合

It should be possible for intents to be routed to external helper applications. For instance, a locally available photo editing tool could be configured to interact with the browser to handle an image edit intent. Or the browser could discover home networking equipment on the local network and make it available for particular types of intents.
外部のヘルパーアプリケーションに送られるインテントは可能になるべきです。具体的には、ローカルで利用可能な写真編集ツールは画像編集インテントを受け取るためにブラウザと相互作用するように設定されるでしょう。または、ブラウザはローカルネットワーク上でホームネットワーキング装備を発見し、インテントの特定の型が利用可能になるでしょう。

既存の**Web**プラットフォーム機能をインテントに置き換え

ローカルリソースと同じ立場でWebアプリケーションを配置するため、Web Intentsを使うにはいくつかの既存の機能が置き換え訳可能であるべきです。具体的には、ユーザがローカルディスクと同じようにクラウドにあるストレージのロッカーからファイルをアップロードすることを選択できる、というように、ユーザエージェントはファイル選択制御をインテントに置き換えられるべきです。これらのケースにおいて、ユーザエージェントは既存の機能に対応するために、ビルトインされたインテントハンドラを供給するかもしれません。
もう一つのユースケースは、Webアプリケーションがプラグインとして機能することを可能にします。例えば、ユーザエージェントがどのように表示するか知らないタイプのリソースを返すリンクは、そのリソースの型を読んでそれを表示する能力を持つWebアプリケーションをユーザに設定することをできるようなインテントに置き換えることができるでしょう。これはWebアプリケーションをプラグインとし

て機能させていることになります。

認証

Web Intentsのための多くのサービスが、ユーザはアカウントを持つことになるだろうと期待されます。そのサービスは、認証を実行するためにユーザエージェントが配置するコンテキスト内で標準的なログイン機構を利用することを可能にすべきです。つまり、インテントを受け取るサービスページは、同じクッキーの瓶やlocalStorageへのアクセスを伴って読み込まれるべきです。インテントデータは、ユーザエージェントによってログインリダイレクトを超えて永続化されるべきです。

プライバシーの考慮

ユーザは、サービスが対象の目的のためにアクションに関連づけられたデータを利用し、データを不適切に共有あるいは持ち続けない、ということについて安心できる必要があります[WEBAPP-PRIVACY-BESTPRACTICES]。この理由のために、特にどのサービスが特定のインテントを受け取るかを決定する選択機構において、ユーザがインテントのコントロール権を持つことが重要です。これは、ユーザのサービスに関する期待への配慮や、保持や二次利用に関するサービスポリシーを含む、サービスの選択に関連したユーザの決定および制御の可能性を提供します。これは制御と承認のプライバシー指針[DAP-PRIVACY-REQS]に関連します。この理由のために、ユーザは明示的にインテントを知るべきであり、それらを閲覧および変更できるべきです。そして、実装はこの機能の提供が推奨されるべきです。最低限のプライバシー指針[DAP-PRIVACY-REQS]に従って、サービスのために必要となる最小限の情報がインテントパラメータとして含まれるべきです。

謝辞

Robin Berjonにクールなツールが多くの感謝をします。彼のクールなツールは私たちの活動をより容易にしました。
webintents.orgにて、他の多くの例や、Web IntentsのサンプルJavaScript実装を見ることができます。

リファレンス

基本としたリファレンス

[BLOB]
Arun Ranganathan. Blob. 17 November 2009. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2009/WD-FileAPI-20091117/#dfn-Blob

[HTML5]
Ian Hickson; David Hyatt. HTML5. 29 March 2012. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/html5

[HTTP11]

R. Fielding; et al. Hypertext Transfer Protocol - HTTP/1.1. June 1999. Internet RFC 2616. URL: http://www.ietf.org/rfc/rfc2616.txt

[POSTMSG]

Ian Hickson. HTML5 Web Messaging. 13 March 2012. W3C Working Draft. (Work In Progress.) URL:http://www.w3.org/TR/webmessaging/

[RFC2046]

N. Freed; N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. November 1996. Internet RFC 2046. URL:http://www.ietf.org/rfc/rfc2046.txt

参考情報のリファレンス

[DAP-PRIVACY-REQS]

Alissa Cooper, Frederick Hirsch, John Morris. Device API Privacy Requirements 29 June 2010. W3C Note URL:http://www.w3.org/TR/2010/NOTE-dap-privacy-reqs-20100629/

[WEBAPP-PRIVACY-REQS]

Frederick Hirsch. Web Application Privacy Best Practices. W3C Working Group Note. URL: http://www.w3.org/TR/app-privacy-bp/