FIDO UAF 应用 API 和传输绑定规范 v1.0

FIDO 联盟推荐标准 2014-.12-.08

当前版本:

https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-client-api-transport-v1.0-ps-20141208.html

之前版本:

https://fidoalliance.org/specs/fido-uaf-client-api-transport-v1.0-rd-20141008.pdf

编写者:

布拉德•希尔 (Brad Hill)，贝宝（PayPal, Inc.）

贡献者:

达维特•巴格达萨利安（Davit Baghdasaryan）, Nok Nok Labs, Inc.

比尔•布兰克（Bill Blanke）, Nok Nok Labs, Inc.

罗尔夫•林德曼博士（Dr. Rolf Lindemann）， Nok Nok Labs, Inc.

杰夫•霍奇斯（Jeff Hodges），贝宝（PayPal, Inc.）

翻译者:

芦馨雨（Xinyu Lu），联想（Lenovo）

常秉俭（Nick Chang），联想（Lenovo）

摘要

本规范介绍了客户端应用程序利用 FIDO UAF 时的 API 以及互操作性。它包括和网页平台及安卓应用程序上的 FIDO UAF 客户端通信的方法、传输要求以及向兼

容的服务器传送 FIDO UAF 消息的 HTTPS 互操作性简介。

文档状态

本章节描述了文档发布时的状态。本文档有可能会被其它文档所取代。当前 FIDO 联盟出版物的列表以及此技术报告的最新修订可在 FIDO 联盟规范索引上找到。网址：https://www.fidoalliance.org/specifications/.

本文档由 FIDO 联盟作为推荐标准发布。如果您希望就此文档发表评论，请联系我们。欢迎所有评论。

本规范中某些元素的实现可能需要获得第三方知识产权的许可，包括（但不限于）专利权。FIDO 联盟及其成员，以及此规范的其他贡献者们不能，也不应该为任何识别或未能识别所有这些第三方知识产权的行为负责。

本 FIDO 联盟规范是“按原样”提供，没有任何类型的担保，包括但不限于，任何明确的或暗示的不侵权、适销性或者适合某一特定用途的担保。

本文档已经由 FIDO 联盟成员评审并签署成为推荐标准。这是一篇稳定的文档，可能会作为参考材料被其它文档引用。FIDO 联盟的作用是引起对规范的注意并促进其广泛的分发。

目录

1. 注释

类型名称、属性名称和元素名称用代码形式书写。

字符串文本包含在双引号“”内，例如“UAF-TLV”。

公式中用 “|” 来表示按字节串联操作。

base64url 符号引用自*无填充*的“Base 64 Encoding with URL and Filename Safe Alphabet”[RFC4648]。

DOM APIs 使用 WebIDL [WebIDL-ED]中的 ECMAScript [ECMA-262]绑定来描述。

根据[WebIDL-ED],结构成员是可选的，除非他们被明确标注为 required。

WebIDL 的结构成员不得为空值。

除非特别声明，如果 WebIDL 的结构成员是 DOMString，则不得为空。

除非特别声明，如果 WebIDL 的结构成员是一个表单，则不得为一个空表单。

本文档中用到的 UAF 专用术语在 FIDO 术语表[FIDOGlossary]中均有定义。

此规范中的所有的图表、示例、注释都是非规范的。

> 注释
>
> 特定的结构成员需要遵从 FIDO 协议的要求。本文档中以 required 为标示，标注了这些词汇在 WebIDL 的定义。关键词 required 是在开发中版本[WebIDL-ED]提出，如果使用执行 WebIDL 开发程序的解析器[WebIDL]，则可删除在 WebIDL 中的关键词 required 并通过其他方式将这些字段填满。

1.1 关键字

本文档中的关键字：“必须”，“不得”，“要求”，“将”，“将不”，“应该”，“不应该”，“建议”，“可能”，“可选”都会按照[RFC2119]的描述来解释。

2. 概览

本节是非规范性的。

FIDO UAF 技术可以替代在线服务中传统的基于用户名和口令的鉴别方案，使在线认证更加强大、更加简单。UAF 协议的核心由 4 个 FIDO UAF 客户端和 FIDO 服务器之间的概念性会话组成：注册、鉴别、交易确认和注销。正如核心协议中规定的那样，使用何种网络传输方式传输这些消息并没有规定，也没有描述用户交互的应用软件该怎样使用 UAF。本文档描述了客户端应用程序与 FIDO UAF 客户端软件进行通信的 API，以及发送 UAF 协议消息到运程服务器的传输模式和安全性要求。

读者需要熟悉 FIDO 术语表[FIDOGlossary]和 UAF 协议规范 [UAFProtocol]。

2.1 受众

本文档的受众是想要使用 FIDO UAF 来实现客户端应用程序的开发者，以及网页浏览器、浏览器插件和 FIDO 客户端的开发者。本文档描述了需要向应用程序开发者公开的 API。

2.2 范围

本文档描述了：

- 支持 FIDO UAF 的网页浏览器，向客户端网页应用程序公开的本地 ECMAScript [ECMA-262] API。
- 安卓[ANDROID]应用程序用来发现并使用一个共享 FIDO UAF 客户服务端所需要的机制和 API。
- 应用程序初始化和传输 UAF 协议交换信息的总体安全要求。
- 在 HTTPS[RFC2818]上传输 FIDO UAF 消息的互操作简介。

下面的内容超出了本文档的范围：

- 底层 UAF 协议消息的格式和细节。
- FIDO 服务器软件和服务端应用程序堆栈之间交互的细节和 APIs。

这里描述传输 FIDO UAF 消息的标准 APIs 和互操作性简介，是为了提供如何开发支持 FIDO 的应用程序的实例。同时也促进了不同供应商之间互操作集成的易用性，从而建立一套完整的 FIDO UAF 解决方案。对任何给定的应用程序实例来说，这些特定的模式可能并不是完美的也不是强制的。应用程序可能使用其他的传输协议，将 UAF 协议消息和其他网络数据捆绑起来，或者发现并使用它们认为合适的其他 APIs。

2.3 结构

UAF 协议的整体结构及其不同操作在 FIDO UAF 协议规范[UAFProtocol]中有描

述。下面的结构简图说明了本文档涉及的交互和参与者：

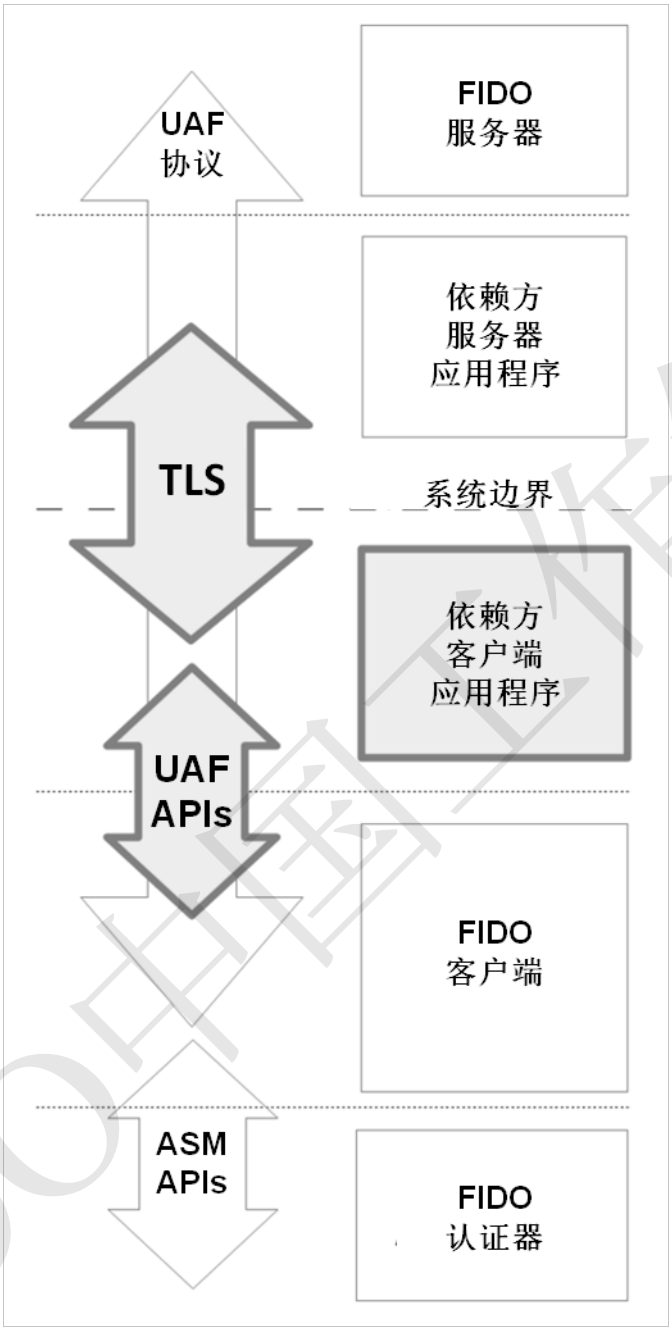

图 1 UAF 应用程序 API 结构和传输层

本文档描述了图 1 中的灰色组件。

2.3.1 协议会话

核心 UAF 协议包括 5 个概念性的阶段：

- **Discovery（发现）**允许依赖方服务器确定客户端 FIDO 功能的可用性，包

括可用认证器的元数据。

- **Registration（注册）**允许客户端生成新的密钥材料，并将其与依赖方服务器上的账户相关联。该过程要遵循服务器设置的规则和可接受的鉴证，认证器和注册信息要与规范中规定的相匹配。
- **Authentication（鉴别）**允许用户向依赖方服务器，提供一个账户的身份标识、与该身份标识相关联的已注册密钥材料的所有权证明、以及其它潜在证明资料。
- **Transaction Confirmation（交易确认）**允许服务器发出请求，使具有适当功能的 FIDO 客户端和认证器向用户显示某些信息，用户使用他们的 FIDO 认证器进行本地确认，并将之前已注册秘钥的所有权证明和确认鉴证返回到依赖方服务器。
- **Deregistration（注销）**在依赖方认定某个秘钥无效时，允许依赖方服务器通知认证器删除与该秘钥关联的本地管理的秘钥材料。

发现不涉及与 FIDO 服务器之间进行协议交换。然而，通过发现 APIs 获得的信息可以以特定于应用程序的方式传送回服务器。例如通过获取包含针对用户设备特定功能的认证器策略的 UAF 协议请求消息。

尽管 UAF 协议抽象的定义了 FIDO 服务器作为请求的发起人，但按照本文中描述，UAF 客户端应用程序总是通过客户端发起的请求/响应协议（如 HTTP）来传输 UAF 协议消息。

从依赖方客户端应用程序的角度来看，注册、鉴别、交易确认的协议流程如下：

1. 客户端应用程序或者明确的联系服务器以获得 UAF 协议请求消息；或者该消息与其它客户端应用程序内容一起发出。
2. 客户端应用程序调用合适的 API，异步地将 UAF 协议请求消息发送给 FIDO UAF 客户端，并且接收一组回调函数。
3. FIDO UAF 客户端与用户和认证器一起执行任何必要的交互，来完成请求。FIDO UAF 客户端使用回调函数来通知客户端应用程序一个错误，或者返回一个 UAF 响应消息。

4. 客户端应用程序通过传输协议（如 HTTP）将 UAF 响应消息发送给服务器。
5. 服务器可以选择性地返回操作指示和附加数据例如授权令牌或一个重定向。
6. 客户端应用程序可以使用适当的 API 通知 FIDO UAF 客户端操作的结果。这允许 FIDO UAF 客户端进行“内务处理”工作以提供更好的用户体验。例如，以后不会再次尝试使用服务器拒绝注册的密钥。
7. 客户端应用程序可以选择性的处理以特定于应用程序的方式返回给它的额外数据。例如，处理新的授权令牌、将用户重定向到新资源、或者解释错误编码来决定是否和应该如何重试失败的操作。

注销不涉及 UAF 协议的往返。如果依赖方服务器通知客户端应用程序执行注销操作，客户端应用程序仅仅简单地使用合适的 API，将 UAF 协议请求消息发送给 FIDO UAF 客户端。FIDO UAF 客户端不会向依赖方客户端应用程序或 FIDO 服务器返回注销操作的结果。

UAF 协议消息是 JSON [ECMA-404]结构的，但是客户端应用程序不能修改它们。这些消息可能包含嵌入式密码完整性保护方案。从 FIDO UAF 客户端或服务器的角度来看，任何修改都可能导致这些消息无效。

3. 通用定义

本节是规范性的。

这些元素是许多 APIs 和层次共享的。

3.1 UAF 状态码

下表列出了 UAF 协议的状态码。

> 注释
>
> 这些编码指示了 FIDO 服务器上的 UAF 操作的结果。它们不代表 HTTP[RFC7230]层或其他传输层。这些编码是为了供客户端网页应用和 FIDO

UAF 客户端使用，用来通知应用程序特定的错误报告、重操作和内务管理“内务处理”行为。

编码	意义
1200	成功。操作完成。
1202	接受。消息已被接受，但是此时还没有完成。依赖方可能需要时间来进行鉴证和风险评估等其他操作。服务器不得把认证令牌与 1202 响应一起发送。
1400	错误请求。服务器不能理解的消息。
1401	未授权。UserID 必须经过鉴别来执行此操作，或者该 KeyID 和此 UserID 不相关。
1403	禁止。UserID 不允许进行此操作。客户端不得重试。
1404	未找到。
1408	请求超时。
1480	未知 AAID。服务器无法定位该 AAID 的授权元数据。
1481	未知 KeyID。对给定的关系，服务器无法定位给定的 UserID 和 KeyID 组合的注册信息。 该错误表明，在用户设备上有一个无效的注册信息。当接收到此错误，建议 FIDO UAF 客户端从本地设备上删除该密钥。
1490	拒绝通道绑定。由于缺失通道绑定或通道绑定不匹配，服务器拒绝执行该请求。
1491	请求无效。服务器拒绝执行该请求，因为请求消息的随机数是未知的、过期的或者服务器之前已经处理过含有相同随机数和 userID 的消息。
1492	无效认证器。根据服务器策略，此认证器是无效的，例如因为服务器使用的功能注册表与客户端发现的功能是不同的。
1493	撤销的认证器。服务器认为该认证器已被撤销。
1494	无效密钥。使用的密钥是无效的。可能该密钥在已知的弱密钥列表中，或使用了不安全的参数选择。

1495	无效算法。服务器认为认证器端可以使用更加强的双方认可的算法，而不是在请求中出现的算法。
1496	无效鉴证。服务器不接受提供的鉴证。
1497	无效客户端功能。服务器不能或不愿使用客户端软件向认证器补充的必要功能。
1498	无效内容。消息内容有问题，服务器不愿意或无法处理它。
1500	内部服务器错误。

4. 共享定义

本节是规范性的。

> **注释**
>
> 本节定义了许多的 WebIDL[WebIDL-ED]指定的 JSON 结构。在多种目标平台中的 APIs 中共享这些结构。

4.1 UAFMessage 结构

UAFMessage 结构是一个封装对象，它包含未处理的 UAF 协议消息和附加的 JSON 数据，客户端应用程序或 FIDO UAF 客户端可以使用附加的 JSON 数据传输特定于应用程序的数据。

```
WebIDL
dictionary UAFMessage{
    required DOMString    uafProtocolMessage;
    Object                additionalData;
};
```

4.1.1 UAFMessage 结构成员

uafProtocolMessage 类型为 required DOMString

该项包含将在 FIDO UAF 客户端或服务器处理的 UAF 协议消息。如果客户端应用程序对消息进行修改可能导致消息无效。客户端应用程序可能会检查

消息中的内容，例如，判断消息是否是新生成的。消息结构的细节可以参考 UAF 协议规范[UAFProtocol]。

additionalData 类型为 Object

该项允许 FIDO 服务器或客户端应用程序以 JSON 对象的形式附加数据，提供给 FIDO UAF 客户端使用；或者，FIDO UAF 客户端或客户端应用程序附加数据，提供给客户端应用程序使用。

4.2 版本接口

描述了 UAF 协议或 FIDO UAF 客户端的版本兼容性校验。

```
WebIDL
dictionary Version {
    readonly    attribute unsigned short    major;
    readonly    attribute unsigned short    minor;
};
```

4.2.1 属性

major 类型为 unsigned short，只读

主要版本编号。

minor 类型为 unsigned short，只读

次要版本编号。

4.3 认证器接口

该接口会在 UAF 的多个阶段使用。Authenticator 接口公开了经过校验的元数据[UAFAuthnrMetadata]的一个子集，以及可用认证器状态的瞬时信息。

WebIDL

```
interface Authenticator {
    readonly    attribute DOMString                         title;
    readonly    attribute AAID                              aaid;
    readonly    attribute DOMString                         description;
    readonly    attribute Version[]                         supportedUAFVersions;
    readonly    attribute DOMString                         assertionScheme;
    readonly    attribute unsigned short                    authenticationAlgorithm;
    readonly    attribute unsigned short[]                  attestationTypes;
    readonly    attribute unsigned long                     userVerification;
    readonly    attribute unsigned short                    keyProtection;
    readonly    attribute unsigned short                    matcherProtection;
    readonly    attribute unsigned long                     attachmentHint;
    readonly    attribute boolean                           isSecondFactorOnly;
    readonly    attribute unsigned short                    tcDisplay;
    readonly    attribute DOMString                         tcDisplayContentType;
    readonly    attribute DisplayPNGCharacteristicsDescriptor[] tcDisplayPNGCharacteristics;
    readonly    attribute DOMString                         icon;
    readonly    attribute DOMString[]                       supportedExtensionIDs;
};
```

4.3.1 属性

title 类型为 DOMString，只读的

一个简短的、用户友好的认证器名。

> 注释
>
> 该文本必须本地化为当前区域设置。
>
> 如果 ASM 没有在 AuthenticatorInfo[UAFASM]对象中返回标题，FIDO UAF 客户端必须根据 AuthenticatorInfo 中的其他项，生成一个标题，因为 title 不能为空（参考章节 1. 注释）。

aaid 类型为 AAID，只读

认证器验证标识符，用来识别认证器的类型和批次。AAID 结构的定义参考 [UAFProtocol]。

description 类型为 DOMString，只读

认证器的一个用户友好的描述性字符串。

> 注释

该文本必须本地化为当前区域设置。

该项是为了用来向用户显示。它可能不同于认证器元数据声明[UAFAuthnrMetadata]中规定的描述。

如果 ASM 没有在 AuthenticatorInfo[UAFASM]对象中返回描述，FIDO UAF 客户端必须根据 AuthenticatorInfo 中的其它项，生成一个有意义的描述并提供给调用应用。因为 description 不能为空（参考章节 1.注释）。

supportedUAFVersion 类型为 *Version* 数组，只读

表示认证器支持的 UAF 协议版本。

assertionScheme 类型为 DOMString，只读

认证器使用的针对验证数据和签名的断言方案。

断言方案标识符在 UAF 注册表预定义值[UAFRegistry]中有定义。

authenticationAlgorithm 类型为 unsigned short，只读

支持的鉴别算法。该值必须与 UAF_ALG_SIGN 前缀的常量相关。

attestationTypes 类型为 unsigned short 数组，只读

支持的鉴证类型列表。该值在[UAFRegistry]中以 TAG_ATTESTATION 为前缀的常量定义。

userVerification 类型为 unsigned long，只读

一个比特位标识集合用来表示认证器支持的用户校验方法。该值由具有 USER_VERIFY 前缀的常量定义。

keyProtection 类型为 unsigned short，只读

一个比特位标识集合用来表示认证器使用的密钥保护方案。该值由具有 KEY_PROTECTION 前缀的常量定义。

matcherProtection 类型为 unsigned short，只读

一个比特位标识集合用来表示认证器使用的匹配器保护方案。该值由具有 MATCHER_PROTECTION 前缀的常量定义。

attachmentHint 类型为 unsigned long，只读

一个比特位标识集合用来表示认证器当前使用何种方式与 FIDO 用户设备连接。该值由具有 ATTACHMENT_HINT 前缀的常量定义。

注释

> 由于认证器的连接状态和拓扑结构是瞬时的，这些值仅仅是一个提示，用来在使用服务器提供的策略时指导用户体验。例如，更倾向于使用一个已经连接好并做好准备进行鉴别或确认小额交易的设备，而不是一个更安全但是需要更多用户工作量的设备。

这些值在认证器元数据中没有标识，也不能被依赖方依赖，尽管有些认证器模型可能提供有语义相近的测量证明作为 UAF 协议消息的一部分。

isSecondFactorOnly 类型为 boolean，只读

认证器是否仅能作为第二因子使用。

tcDisplay 类型为 unsigned short，只读

一个比特位标识集合用来指示交易确认信息显示的可用性和类型。该值由具有 TRANSACTION_CONFIRMATION_DISPLAY 前缀的常量定义。

如果认证器不支持交易确认，该值必须设为 0.

tcDisplayContentType 类型为 DOMString，只读

交易确认显示支持的 MIME 内容类型[RFC2045]，例如“text/plain”或“image/png”。

如果支持交易确认（当 tcDisplay 是非 0 时），该值必须不为 0。

tcDisplayPNGCharacteristics 类型为 DisplayPNGCharacteristicsDescriptor 数组 只读

（如果有的话）交易确认显示*目前*支持的 PNG 特性的集合。

> **注释：**
> 关于该字段的格式和 DisplayPNGCharacteristicsDescriptor 结构的定义，参考[UAFAuthnrMetadata]获取更多信息。

如果支持交易确认（当 tcDisplay 是非 0 时），该值必须不为空。

icon 类型为 DOMString，只读

认证器的 PNG[PNG]图标，编码为一个 “data:” url [RFC2397]。

> **注释**

> 如果 ASM 在 AuthenticatorInfo[UAFASM]对象中没有返回 icon，FIDO UAF 客户端必须设置一个默认图标，因为 icon 不允许为空（参考章节 1. 注释）。

supportedExtensionIDs 类型为 DOMString 数组 只读

UAF 协议扩展标识符的支持列表。这些值可能是供应商定义的。

4.3.2 认证器接口常量

定义了一些常量，用来与比特位标识字段 userVerification，keyProtection，attachmentHint 和 tcDisplay 一起使用。为了避免重复和矛盾，这些值在 FIDO UAF 注册表预定义值[UAFRegistry]中定义。

4.4 DiscoveryData 结构

WebIDL

```
dictionary DiscoveryData {
    required Version[]          supportedUAFVersions;
    required DOMString          clientVendor;
    required Version            clientVersion;
    required Authenticator[]    availableAuthenticators;
};
```

4.4.1 DiscoveryData 结构成员

supportedUAFVersions 类型为 required Version

客户端支持的 FIDO UAF 协议版本的列表。优先度高的在前。

clientVendor 类型为 required DOMString

FIDO UAF 客户端的供应商。

clientVersion 类型为 required Version

FIDO UAF 客户端的版本。对于客户端软件来说是特定于供应商的版本，而非 UAF 版本。

availableAuthenticators 类型为 required Authenticator 数组

一个包含描述可使用 UAF 认证器的认证器结构的数组。顺序是不重要的。该列表可能为空。

4.5 ErrorCode 接口

```
WebIDL
interface ErrorCode{
    const short    No_ERROR = 0x0;
    const short    WAIT_USER_ACTION = 0x1;
    const short    INSECURE_TRANSPORT = 0x2;
    const short    USER_CANCELLED = 0x3;
    const short    UNSUPPORTED_VERSION = 0x4;
    const short    NO_SUITABLE_AUTHENTICATOR = 0x5;
    const short    PROTOCOL_ERROR = 0x6;
    const short    UNTRUSTED_FACET_ID = 0x7;
    const short    UNKNOWN = 0xFF;
};
```

4.5.1 常量

NO_ERROR 类型为 short

在没有遇到错误的情况下完成操作。当收到该编码时，应用程序应该不再等待相关的 **UAFResponseCallback** 被发起。

WAIT_USER_ACTION 类型为 short

等待进行用户动作。例如，在 FIDO 客户端用户界面选择一个认证器，执行用户校验，或者使用一个认证器完成注册步骤。

INSECURE_TRANSPORT 类型为 short

window.location.protocol 不是“https”或 DOM 包含不安全的混合内容。

USER_CANCELLED 类型为 short

用户拒绝为完成注册而进行的任何必要的交互。

UNSUPPORTED_VERSION 类型为 short

FIDO UAF 客户端不支持 **UAFMessage** 中规定的协议版本。

NO_SUITABLE_AUTHENTICATOR 类型为 short

没有与 **UAFMessage** 中认证器策略相匹配的可用认证器来处理请求，或者用户拒绝使用合适的认证器。

PROTOCOL_ERROR 类型为 short

出现 UAF 协议违规现象。交互可能出现超时；与消息相关的来源可能与调用 DOM 环境的来源不匹配，或者协议消息可能是格式不正确的或被篡改的。

UNTRUSTED_FACET_ID 类型为 short

客户端拒绝处理操作，因为计算出的请求者的类型标识符不在请求消息中规定的应用标识符的可信列表里。

UNKNOWN 类型为 short

没有被上述列举的编码描述的错误情况。

5. DOM API

本节是规范性的。

本节描述了网页浏览器或浏览器插件，向执行在 Document[DOM]中的客户端网页应用公开的 API。

5.1 特征检测

FIDO UAF DOM APIs 根植于一个新的 fido 对象，这个 fido 对象是一个 window.navigator 编码的属性；它的存在和属性可能用于特征检测。

例 1: UAF APIs 的特征检测

```
例 1
    <script>
    if(!!window.navigator.fido.uaf) { var useUAF = true; }
    </script>
```

5.2 uaf 接口

window.navigator.fido.uaf 接口是和 FIDO UAF 客户端进行交互的基本方法。所有操作都是异步的。

WebIDL

```
interface uaf{
    void discover (DiscoveryCallback completionCallback, ErrorCallback
        errorCallBack);
    void checkPolicy (UAFMessage message, ErrorCallback cb);
    void processUAFOperation (UAFMessage message,
        UAFResponseCallback completionCallback, ErrorCallback
        errorCallback);
    void notifyUAFResult (int responseCode, UAFMessage uafResponse);
};
```

5.2.1 方法

discover

发现用户客户端软件和设备是否支持 UAF，如果认证器能力允许，它可能愿意接受鉴别。

参数	类型	可为空	可选	描述
completionCallback	**DiscoveryCallback**	✘	✘	该回调接收 FIDO UAF 客户端 **DiscoveryData**。
errorCallback	**ErrorCallback**	✘	✘	一个回调函数，用来接收错误和进程事件。

返回类型： void

checkPolicy

在不提示用户的情况下，询问浏览器或浏览器插件是否可以处理提供的请求消息。

与使用 **ErrorCallback** 的其它操作不同，该操作必须总是触发回调。当其认为消息可以处理并且有与内置的策略匹配的可用认证器时，返回 NO_ERROR，否则返回其他适当的 ErrorCode 值。

> **注释**
>
> 因为这个调用不会提示用户，它不能引起潜在的上下文切换混乱，即使 FIDO UAF 客户端在进程外实现的。

参数	类型	可为空	可选	描述
message	**UAFMessage**	✘	✘	包含需要检测的策略和操作的 **UAFMessage**。
cb	**ErrorCallback**	✘	✘	接收操作状态的回调函数。

返回类型： void

processUAFOperation

调用 FIDO UAF 客户端，传递控制权以便在必要时提示用户完成操作，并且向回调函数返回消息，消息要符合 UAFMessage 中标示的支持协议版本。

参数	类型	可为空	可选	描述
message	**UAFMessage**	✘	✘	FIDO 客户端软件使用的 **UAFMessage**。
completionCallback	**UAFResponse Callback**	✘	✘	用来接收 FIDO UAF 客户端的响应 **UAFMessage** 的回调，会被发送到依赖方服务器。

errorCallback	**ErrorCallback**	✘	✘	用来接收来自 FIDO UAF 客户端的错误和进程事件的回调。

返回类型： void

notifyUAFResult

用来显示远程服务器收到 FIDO UAF 消息后的结果状态码。当应用程序接收到来自服务器的 UAF 状态码时，必须调用该接口。这使得 FIDO UAF 客户端可以进行“内务处理”操作以提供更佳的用户体验，例如，不使用被服务器拒绝注册的密钥。

> **注释**
> 服务器是否发送并且如何发送状态码，针对不同的应用程序和不同的传输方式是不同的。在下面的 HTTPS 传输互操作性概况中，可以找到一个非规范的例子。

参数	类型	可为空	可选	描述
responseCode	int	✘	✘	**ServerResponse** 中的 uafResult 字段。
uafResponse	**UAFMessage**	✘	✘	到这条 responseCode 适用的 **UAFMessage**。

返回类型： void

5.3 UAFResponseCallback

FIDO UAF 客户端在一个异步操作成功完成时，使用 UAFResponseCallback 向客户端应用程序返回协议响应消息，然后客户端应用程序再将协议响应消息发给服务器。

WebIDL

```
callback UAFResponseCallback = void (UAFMessage uafResponse);
```

5.3.1 UAFResponseCallback 回调参数

uafResponse 类型为 **UAFMessage**

代表 FIDO UAF 客户端对服务器请求消息的响应消息和其他数据。

5.4 DiscoveryCallback

FIDO UAF 客户端在一个异步的发现操作成功完成时，使用 DiscoveryCallback 向客户端应用程序返回 DiscoveryData。

WebIDL

```
callback DiscoveryCallback = void (DiscoveryData data);
```

5.4.1 DiscoveryCallback 回调参数

data 类型为 **DiscoveryData**

用来向应用程序描述 FIDO UAF 客户端软件和可用认证器的当前状态。

5.5 ErrorCallback

FIDO UAF 客户端使用 ErrorCallback 来返回异步操作的进度和错误码。

WebIDL

```
callback ErrorCallback = void (ErrorCode code);
```

5.5.1 ErrorCallback 回调参数

data 类型为 **ErrorCode**

来自 ErrorCode 接口的值用来表示操作的结果。

对某些操作来说，ErrorCallback 可能被调用多次，例如返回

WAIT_USER_ACTION 代码的情况。

5.6 DOM API 的隐私性考虑

本节是非规范性的。

用户设备上 FIDO 能力的差异（和其它特性相比），可能会允许服务器“指纹验证”远程客户端，并尝试一直去验证它，甚至在缺乏任何明确的会话状态维护机制时也会如此。尽管这样可能会导致大量的信号被发送给服务器用来“指纹验证”客户端，Discovery API 暴露的属性被设计成一个匿名的大集合并且几乎不会产生新的隐私风险。尽管如此，一个 FIDO UAF 认证器的不常见的配置也许能有效地唯一识别出一个用户。

推荐用户代理向所有应用程序公开 Discovery API，默认地不要要求明确的用户意见。但是从隐私角度考虑，如果用户代理或 FIDO 客户端开发者愿意的话，应该给用户提供自愿退出发现的方法。

5.7 DOM API 的安全性考虑

本节是非规范性的。

5.7.1 不安全的混合内容

当 FIDO UAF APIs 被调用，并且操作在网页用户代理的 Document 上下文中执行的时候，这样的环境不得包含不安全的混合内容。“不安全的混合内容”的确切定义，对于不同的用户代理来说是不一样的，但是通常都包含脚本、插件和其他“活动”内容，是组成 DOM 的一部分或者能够使用 DOM，其本身并不通过 HTTPS 加载。

当一个 Document 上下文调用本文档中定义的任何 APIs 时，如果这个 Document 上下文不是被安全的传输方式加载的，并且（或者）包含不安全的混合内容，那

么 UAF APIs 必须马上触发 ErrorCallback 返回 INSECURE_TRANSPORT 错误码，并且终止任何其他的进一步操作。

5.7.2 同源策略、HTTP 重定向和跨源内容

当使用 HTTP 检索或传输 UAF 协议消息时，非常重要的一点是，document 上下文的网页源与 UAF 协议消息的源要保持一致。两个源如果不匹配，可能会导致协议失败，或者受到针对协议的攻击。因此：

FIDO UAF 消息不应该使用没有同源策略[SOP]的方法进行传输，例如在不同源 URLs 中使用<script src="url">，或者在服务器上设置 Access-Control-Allow-Origin 头。

当使用 XMLHttpRequest [XHR]传输 FIDO UAF 消息时，客户端不应该跟随重定向跳转到与请求 document 不同源的 URLs。

当 HTTP 响应的整个响应体被解析为合法的 ECMAScript，FIDO UAF 消息不应该被 HTTP 响应暴露出来。以这种方式暴露的资源可能会与不可信源的恶意应用程序通过使用<script src="url">进行跨源植入的方式进行未经授权的交互。

网页应用程序不应该通过诸如 postMessage() [webmessaging]的通道跨源分享 FIDO UAF 消息。

5.8 浏览器/插件开发者的注意事项

本节是非规范性的。

网页应用程序对 UAF 的利用取决于作为可信平台的网页浏览器提供的服务。网页应用程序的 API 不会为 FIDO 操作提供一种方式去声称一个源为一个应用程序标识符，基于浏览器对真正源环境的了解，浏览器将会为 FIDO UAF 客户端提供这些信息。

当应用程序标识符准确时，浏览器必须让网页源与 FIDO UAF 客户端通信。

并且，浏览器必须不能让含有不安全混合内容的资源实例使用 UAF DOM APIs。

6. 安卓 Intent API

本节是规范性的。

本节描述了安卓[ANDROID]客户端应用程序如何定位在主设备上安装的符合 FIDO UAF 规范的 FIDO 客户端，并与之通信。

> 注释
> 与网页应用程序一样，安卓平台上有多种可行的集成模式。在此处描述的 API 使应用程序运用安卓 Intents 这种松耦合的方式，与用户设备上共享的 FIDO UAF 客户端进行通信。

6.1 针对安卓特定的定义

6.1.1 org.fidoalliance.uaf.permissions.FIDO_CLIENT

运行在安卓 5 版本以前平台上的 FIDO UAF 客户端必须声明 org.fidoalliance.uaf.permissions.FIDO_CLIENT 许可，同时也必须声明相关的“use-permission”。参考下文中安卓应用的 manifest 文件中<permission/>和<uses-permission/>元素[AndroidAppManifest]中对许可描述的示例。

运行在安卓 5 或更新版本上的 FIDO UAF 客户端不得声明此许可，并且也不得声明相关的“use-permission”。

> 例 2
> ```
> <permission
>     android:name="org.fidoalliance.uaf.permissions.FIDO_CLIENT"
>     android:label="Act as a FIDO Client."
>     android:description="This application acts as a FIDO Client. It may
> ```

```
            access authentication devices available on the system, create and
            delete FIDO registrations on behalf of other applications."
        android:protectionLevel="dangerous"
/>
<uses-permission
        android:name="org.fidoalliance.uaf.permissions.FIDO_CLIENT"/>
```

注释

- 由于 FIDO 客户端会执行安全性相关的任务（例如，校验 AppID/FacetID 的关系并请求用户许可），因此用户应该谨慎选择它们所使用的 FIDO 客户端。要求应用程序作为 FIDO 客户端进行声明并且使用这个许可向用户证明自己的合规性。
- 没有任何 FIDO 客户端资源需要基于 FIDO_CLIENT 许可的“保护”。让 FIDO 客户端声明 FIDO_CLIENT 许可的理由，仅仅是因为用户应该能够谨慎地决定安装哪个 FIDO 客户端。
- 安卓 5 改变了当多个应用程序声明相同许可的情况的处理方法[Android5Changes]；它会阻止之后声明了相同许可的应用的安装。
- *在安卓版本 5 标记一个应用作为 FIDO 客户端的最好方式还有待决定。*

6.1.2 org.fidoalliance.uaf.permissions.ACT_AS_WEB_BROWSER

安卓应用程序可以根据它们自己的身份，来请求 FIDO UAF 客户端的服务。或者在调用 FIDO UAF 客户端时，安卓应用程序通过明确声明作为一个远程服务器源的 RFC6454[RFC6454]序列化的方式来充当用户代理。

作为单一实体运行的应用程序，不得设置明确的源。省略明确的源将会使 FIDO UAF 客户端为会决定调用者的标识为 android:apk-key-hash:<hash-of-public-key>。FIDO UAF 客户端将会用它来与目标 AppID 的授权应用程序类型列表进行比对，如果它在列表中标识为可信的，则可以继续执行操作。

> 注释
>
> 参考 UAF 协议标准[UAFProtocol]，获取关于应用程序和类型标识符的更多信息。

如果应用程序是明确在任意数量的远程应用程序的环境下作为用户代理使用（当实现了一个完整的网页浏览器时），它可能会将自己的源设置为 RFC6454[RFC6454]Unicode 序列化的远程应用程序源。在设置源之前，应用程序必须满足传输安全要求中描述的必要条件来鉴别远程服务器。

使用源参数需要应用程序声明 org.fidoalliance.uaf.permissions.ACT_AS_WEB_BROWSER 许可，并且在执行操作之前，FIDO UAF 客户端必须校验请求应用程序有该许可。

> 例 3

```
<permission
    android:name="org.fidoalliance.uaf.permissions.ACT_AS_WEB_BROWSER"
     android:label="Act as a browser for FIDO registrations."
     android:description="This application may act as a web browser,
creating new and accessing existing FIDO registrations for any domain."
  android:protectionLevel="dangerous"
/>
```

6.1.3 channelBindings

本节是非规范性。

在 DOM API 中，浏览器或浏览器插件主要负责向 FIDO 客户端提供可以使用的通道绑定信息。但是作为传输通道的直接拥有者，安卓应用程序必须自己提供该信息。

channelBindings 的数据结构为：

Map<String,String>

附带 UAF 协议规范[UAFProtocol]中为 channelBindings 结构定义的密钥。

使用 TLS 中绑定通道，可以帮助服务器确认传递 UAF 协议消息的通道与合法的客户端使用的通道是相同的，并且也可以确认消息没有被恶意方转发。

UAF 定义了对[RFC5929]中的 tls-unique 和 tls-server-end-point 绑定的支持，以及对服务器证书和 ChannelID [ChannelID]绑定的支持。客户端应该提供它能够使用的所有通道绑定信息。

缺少通道绑定信息或无效的通道绑定信息可能会导致一个依赖方服务器拒绝交易。

6.1.4 UAFIntentType 枚举

该枚举描述了执行安卓 API 的 intent 的操作类型。

> 注释
>
> 即使同一个设备上有可能安装了多个 FIDO 客户端，UAF 仅使用单一的 intent 来简化行为。在这种情况下，可能会提示用户来选择使用手机上安装的哪一个 FIDO UAF 客户端来处理一个隐式的 intent。

如果用户选择不同的 FIDO UAF 客户端默认地来处理代表不同阶段的 intent，可能会导致不一致的结果或者彻底运行失败。

如果应用程序的工作流程需要多次调用客户端（经常是这样的），应用程序应该从 startActivityForResult()接口返回的 intent extra 中读取 componentName，并且在接下来的 intent 中使用 setComponent()接口传入该值，从而确保它们的请求始终是同一个 FIDO UAF 客户端处理的。

```
WebIDL
enum UAFIntentType {
    "DISCOVER",
    "DISCOVER_RESULT",
    "CHECK_POLICY",
    "CHECK_POLICY_RESULT",
    "UAF_OPERATION",
    "UAF_OPERATION_RESULT",
    "UAF_OPERATION_COMPLETION_STATUS"
};
```

枚举描述	
DISCOVER	发现。
DISCOVER_RESULT	发现结果。
CHECK_POLICY	进行一个空操作检查，看消息是否可以被处理。
CHECK_POLICY_RESULT	策略检查的结果。
UAF_OPERATION	处理注册、鉴别、交易确认或注销消息。
UAF_OPERATION_RESULT	UAF 操作结果。
UAF_OPERATION_COMPLETION_STATUS	通知 FIDO UAF 客户端注册、鉴别、交易确认或注销消息在服务器端处理之后的完成状态。

6.2 org.fidoalliance.intent.FIDO_OPERATION Intent

安卓上 FIDO UAF 客户端和应用程序之间所有的交互，都是通过单一的安卓 intent 来进行：

org.fidoalliance.intent.FIDO_OPERATION

操作的具体细节通过包含在 intent 中的 MIME 媒介类型和各种 extra 数据进行承载。

本文档描述的操作是属于 application/fido.uaf_client+jsonMIME 媒介类型，而且必须将所有 intent 的 type 参数设为该类型。

> 注释
>
> 客户端应用程序可以将该 intent 通过使用 PackageManager.queryIntentActivities(Intent intent, int flags)来找可用的活动，从而发现系统里是否有一个（或多个）可用的 FIDO UAF 客户端。

extra	类型	描述
UAFIntentType	String	**UAFIntentType** 枚举值之一，来描述 intent。
discoveryData	String	**DiscoveryData** JSON 结构。
componentName	String	响应 FIDO UAF 客户端的组件名。必须使用 ComponentName.flattenString()进行序列化。
errorCode	short	操作的 **ErrorCode** 值。
message	String	进行测试或处理的 **UAFMessage** 请求依赖于 **UAFIntentType**。
origin	String	如果请求者有 org.fidoalliance.permissions.ACT_AS_WEB_BROWSER 许可，该值是为请求提供的一个 RFC6454 网页源[RFC6454]的字符串。
channelBindings	String	操作的通道绑定 JSON 结构。
responseCode	short	ServerResponse 的 uafResult 字段。

根据 UAFIntentType 项的 extra 值，下表列出了其他需要出现的 intent extra：

UAFIntentType 值	discoveryData	componentName	errorCode	message	origin	channelBindings	responseCode
"DISCOVER"							
"DISCOVER_RESULT"	可选	必须	必须				
"CHECK_POLICY"				必须	可选		

"CHECK_POLICY_RESULT"		必须	必须				
"UAF_OPERATION"				必须	可选	必须	
"UAF_OPERATION_RESULT"		必须	必须	可选			
"UAF_OPERATION_COMPLETION_STATUS"				必须			必须

6.2.1 UAFIntentType.DISCOVER

该安卓 intent 调用 FIDO UAF 客户端来发现可用的认证器及能力。FIDO UAF 客户端通常不会显示与处理该 intent 相关的用户界面，而是会马上返回 JSON 结构体。然而，请求应用程序不能依赖于此，因为出于私密性的考虑，FIDO UAF 客户端可能会显示一个用户界面，让用户选择是否向请求应用程序公开或具体公开哪个（哪些）认证器。

该 intent 必须使用 startActivityForResult()调用。

6.2.2 UAFIntentType.DISCOVER_RESULT

当接收到 DISCOVER 类型的 intent 时，FIDO UAF 客户端应该将带有该类型的 intent 作为参数返回给 onActivityResult()接口。

如果发到 onActivityResult()的 resultCode 是 RESULT_OK，并且 intent 的 extra 中 errorCode 值是 NO_ERROR，那么该 intent 具有一个 extra 为 discoveryData，它是一个 **DiscoveryData** 格式的 JSON 结构字符串，指明可用的认证器和功能。

6.2.3 UAFIntentType.CHECK_POLICY

在不提示用户的情况下，该 intent 调用 FIDO UAF 客户端来发现它是否可以处理提供的消息。处理该 intent 的动作不得在用户界面显示。

该 intent 需要如下的 extra：

- message，是一个 **UAFMessage** 类型的字符串，包含需要测试的请求消息。
- origin，是一个可选的 extra，使具有 org.fidoalliance.uaf.permissions.ACT_AS_WEB_BROWSER 许可的调用者提供一个 RFC6454 源[RFC6454]字符串来代替应用程序本身的身份。

该 intent 必须使用 startActivityForResult()调用。

6.2.4 UAFIntentType.CHECK_POLICY_RESULT

当接收到 CHECK_POLICY 类型的 intent 时，FIDO UAF 客户端应该将该安卓 intent 作为参数返回给 onActivityResult()接口。

除了向 onActivityResult()返回 resultCode 外，该 intent 含有一个 extra 为 errorCode，包含一个 ErrorCode 值，指明具体的错误情况，或者当 FIDO UAF 客户端能够处理消息时，返回 NO_ERROR。

6.2.5 UAFIntentType.UAF_OPERATION

该安卓 intent 调用 FIDO UAF 客户端来处理提供的请求消息，并且返回一个准备发给 FIDO UAF 服务器的响应消息。

发送者应该假设 FIDO UAF 客户端会显示一个用户界面让用户来处理该 intent，例如，提示用户完成校验过程。

该 intent 需要下列 extra：

- message，是一个 **UAFMessage** 类型的字符串包含需要处理的请求消息。
- channelBindings，包含 FIDO UAF 协议规范[UAFProtocol]中定义的 ChannelBinding 结构的 JSON 字符串表达形式。

- origin，是一个可选参数，使具有 org.fidoalliance.uaf.permissions.ACT_AS_WEB_BROWSER 许可的调用者提供一个 RFC6454 源[RFC6454]字符串来代替应用程序本身的身份。

该 intent 必须使用 startActivityForResult()调用。

6.2.6 UAFIntentType.UAF_OPERATION_RESULT

当接收到 UAF_OPERATION 类型的 intent 时，FIDO UAF 客户端应该将该安卓 intent 作为参数返回给 onActivityResult()接口。

如果返回给 onActivityResult()的 resultCode 是 RESULT_CANCELLED，该 intent 将含有一个 extra 为 errorCode 参数，包含指示特定错误情况的 **ErrorCode** 值。

如果返回给 onActivityResult()的 resultCode 是 RESULT_OK，并且 errorCode 值是 NO_ERROR，那么这个 intent 含有一个 message 包含 **UAFMessage** 类型的字符串，将作为 UAF 协议响应消息发送给 FIDO 服务器。

6.2.7 UAFIntentType.UAF_OPERATION_COMPLETION_STATUS

该 intent 必须发送给 FIDO UAF 客户端，来指示发送给远程服务器的 FIDO UAF 消息的处理状态，该操作是非常重要的，特别是当一个新的注册请求被客户端挂起时，直到服务器允许其注册才能解除挂起状态。

6.3 安卓开发的安全性考虑

本节是非规范性的。

安卓应用程序可能至少会选择下列两种方法来实现 FIDO 的用户交互部分：

- 通过使用安卓内置的用户界面组件来编写一个安卓活动，或
- 使用基于 HTML 的方法，通过加载安卓网页视图并通过 addJavaScriptInterface()方法将 UAF DOM APIs 注入其中。

一个应用程序如果选择将 UAF 接口注入一个网页视图的话，必须满足适用于 DOM APIs 的使用和适用于用户代理实现者的所有适当的安全考虑。

特别地，注入 API 的网页视图的内容必须只能从授信的本地内容加载、或通过传输安全要求中指定的安全通道加载，并且不能包含不安全的混合内容。

只有应用程序需要为未知数量的第三方应用程序作用户代理的情况下，才可以声明 ACT_AS_WEB_BROWSER 许可。如果一个安卓应用程序与依赖方应用程序有明确的关系，为这些应用程序进行访问控制的最优方法是将安卓应用程序的身份标识列为可信类型。参考 UAF 协议规范[UAFProtocol]获取关于应用程序和类型标识符的更多信息。

为了防护恶意应用程序将它自己配置为 FIDO UAF 客户端的攻击，依赖方应用程序可以获取响应应用程序的标识符，并且可以在鉴别或交易事件过程中，利用它进行风险管理决策。

例如，依赖方可以维护一份已知为恶意软件应用程序的标识符列表，并且拒绝接受这样的客户端完成的操作，或者维护一份风险评估为首选的已知优秀应用程序的标识符列表。

运行在安卓 5 之前版本的依赖方应用程序必须确认 FIDO UAF 客户端具有 org.fidoalliance.uaf.permissions.FIDO_CLIENT 的“使用许可”。运行在安卓 5 以上的依赖方应用程序不需要执行此检查。

> **注释**
>
> 依赖方应用程序应该在安卓 5 及之前版本上执行此检查，通过使用包管理器来验证 FIDO 客户端确实声明了 org.fidoalliance.uaf.permissions.FIDO_CLIENT 许可（见下文实例）。依赖方应用程序不应该使用 FIDO_CLIENT 的“使用许可”。

例 4

```
boolean checkFIDOClientPermission(String packageName)
                                        throws NameNotFoundException {
```

```
        for (String requestedPermission :
                getPackageManager().getPackageInfo(packageName,
                  PackageManager.GET_PERMISSIONS).requestedPermissions) {
            if (requestedPermission.matches(
                    "org.fidoalliance.uaf.permissions.FIDO_CLIENT"))
                return true;
        }
      return false;
    }
```

7. iOS 自定义 URL API

本节是规范性的。

本节描述了 iOS 依赖方应用程序如何定位在主设备上安装的符合 FIDO 规范的 FIDO UAF 客户端，并与之通信。

注释

由于沙箱技术和没有真正的多任务支持，iOS 操作系统提供非常有限的方法来进行进程间通信（IPC）。

FIDO UAF 客户端的任何 IPC 解决方案必须满足：

1. 识别出调用应用程序，从而提供 FacetID(类型标识符)的校验。
2. 在没有用户介入的情况下，允许跳转到另一个应用程序。

目前，iOS 上唯一满足上述要求的 IPC 方法是自定义 URL 处理程序。

自定义 URL 处理程序使用 iOS 操作系统来处理来自发送者的 URL 请求，启动接收应用程序，然后将请求传送给接受应用程序进行处理。通过给两个不同的应用程序运用自定义 URL 处理程序，可以使它们之间实现双向的 IPC，一个自定义 URL 处理程序从应用程序 A 发送数据到应用程序 B，另一个自定义 URL 处理程序从应用程序 B 发送数据到应用程序 A。

因为 iOS 没有真正的多任务支持，所以必须有一个应用转换来处理每一个请求和响应。过多的应用转换会对用户体验产生负面影响，所以依赖方应用程序必

须小心抉择何时必须查询 FIDO UAF 客户端。

7.1 iOS 专有定义

7.1.1 X-Callback-URL 传输

当依赖方应用程序与 FIDO UAF 客户端通信时，会发送一个带有标准 x-callback-url 格式的 URL（参考 x-callback-url.com）：

例 5

```
FidoUAFClient1://x-callback-url/[UAFxRequestType]?x-
success=[RelyingPartyURL]://x-callback-url/
        [UAFxResponseType]&
        key=[SecretKey]&
        state=[STATE]&
        json=[Base64EncodedJSON]
```

- FidoUAFClient1 是 FIDO UAF 客户端使用的 iOS 自定义 URL 方案。根据 x-callback-url 规范中的规定的，传输层的版本信息使用 URL 方案本身进行编码（在此种情况下，为 FidoUAFClient1）。所以，其它应用程序可以调用 canOpenURL 来检查是否支持版本 1.0。
- [UAFxRequestType]是在请求操作时使用的类型，下文中对其有描述。
- [RelyingPartyURL]是依赖方应用程序注册的 URL，用来接收响应。根据 x-callback-url 规范，使用 x-success 参数来定义它。
- [UAFxResponseType]是在响应操作时使用的类型，下文中对其有描述。
- [SecretKey]是请求应用程序随机生成的密钥，使用 base64url 编码，无填充。

  来自 FIDO UAF 客户端的响应会被此密钥加密，从而阻止恶意应用程序通过欺骗返回的 URL 来获取信息。

- [STATE]是用来将请求与响应匹配起来的数据。

- 最后，[Base64EncodedJSON]包含了需要发送给 FIDO UAF 客户端的消息。

  使用 JSON 格式存储条目，然后使用无填充的 base64url 进行编码。

对 FIDO UAF 客户端来说，自定义 URL 方案处理程序入口函数是 openURL()函数：

例 6

```
(BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
```

这里，上述的 URL 通过 url 参数接收。出于安全性考虑，sourceApplication 参数包含依赖方应用程序的 iOS bundle ID。该 bundle ID 必须用来验证应用程序的 FacetID（类型标识符）。

相反地，当 FIDO UAF 客户端响应请求时，它会使用标准 x-callback-url 格式返回下述 URL：

例 7

```
[RelyingPartyURL]://x-callback-url/
        [UAFxResponseType]&
        state=[STATE]&
        json=[Base64EncodedJWE]
```

响应中的参数和请求中的类似，除了[Base64EncodedEncryptedJSON]参数在进行 base64url 无填充编码前，需要用公钥进行加密。[STATE]和请求中发送的 STATE 相同，它会返回给发送者来检验响应是否匹配。

在依赖方应用程序的 openURL()处理程序中，url 参数是之前列出的 URL，sourceApplication 参数为提供给 FIDO 客户端应用程序的 iOS bundle ID。

7.1.2 秘钥的生成

请求应用程序必须在每次向 FIDO UAF 客户端发送请求时，生成一个新的秘钥。FIDO UAF 客户端必须在响应请求者之前，使用这个密钥来加密响应消息。

加密的响应消息必须使用 JSON 网络加密(JWE)、JSON 序列化（JWE 章节 7.2）格式。

"A128CBC-HS256" 中规定的加密算法： JWE“密钥管理模式（ Key Management Mode）”采用“直接加密（Direct Encryption）”；JWE“内容加密密钥（ Content Encryption Key（CEK））”是请求应用程序生成的秘钥，并且通过请求中的 key 参数传送给 FIDO UAF 客户端。

例 8

```
{ "unprotected": {"alg": "dir", "enc": "A128CBC-HS256"}, "iv": "...", "ciphertext": "...", "tag": "..." }
```

7.1.3 源

iOS 应用程序可以使用自己的身份向 FIDO 客户端请求服务，或者也可以通过明确声明一个远程服务器源的 RFC6454[RFC6454]序列作为用户代理调用 FIDO UAF 客户端。

作为单一实体运行的应用程序，不得设置明确的源。省略明确的源将会使 FIDO UAF 客户端判定调用者的形式的身份标识为“ios:bundle-id”。FIDO UAF 客户端将会用它来与目标应用标识符的授权应用程序类型列表进行比对，如果它在列表中标识为可信的，则可以继续执行操作。

参考 UAF 协议标准[UAFProtocol]，获取关于应用程序和类型标识符的更多信息。

如果应用程序是明确在任意数量的远程应用程序的环境下作为用户代理使用（当实现了一个完整的网页浏览器时），它可能会将自己的源设置为

RFC6454[RFC6454]Unicode 序列化的远程应用程序源。在设置源之前，应用程序必须满足传输安全要求中描述的必要条件来鉴别远程服务器。

7.1.4 channelBindings

本节是非规范的。

在 DOM API 中，浏览器或浏览器插件主要负责向 FIDO 客户端，提供可以使用的通道绑定信息。但是作为传输通道的直接拥有者，iOS 应用程序必须自己提供该信息。

通道绑定的数据结构为 Map<String,String>，附带 UAF 协议规范[UAFProtocol]中为 ChannelBinding 结构定义的密钥。

使用 TLS 绑定通道，可以帮助服务器确认传递 UAF 协议消息的通道与合法的客户端使用的通道是相同的，并且也可以确认消息没有被恶意方转发。UAF 定义了对[RFC5929]中的 tls-unique 和 tls-server-end-point 绑定的支持，以及对服务器证书和 ChannelID [ChannelID]绑定的支持。客户端应该提供它能够使用的所有通道绑定信息。

缺少通道绑定信息或无效的通道绑定信息可能会导致一个依赖方服务器拒绝交易。

7.1.5 UAFxType

该值描述了实现 iOS API 的 x-callback-url 操作的操作类型。

```
WebIDL
enum UAFxType {
    "DISCOVER",
    "DISCOVER_RESULT",
    "CHECK_POLICY",
    "CHECK_POLICY_RESULT",
    "UAF_OPERATION",
    "UAF_OPERATION_RESULT",
    "UAF_OPERATION_COMPLETION_STATUS"
};
```

枚举描述	
DISCOVER	发现
DISCOVER_RESULT	发现的结果
CHECK_POLICY	进行一个空操作检查，看消息是否可以被处理。
CHECK_POLICY_RESULT	策略检查的结果
UAF_OPERATION	UAF 消息操作类型（例如 Registration）。
UAF_OPERATION_RESULT	UAF 操作结果。
UAF_OPERATION_COMPLETION_STATUS	通知 FIDO UAF 客户端 UAF 操作（例如 Registration）最终在服务器端处理之后的完成状态。

7.2 JSON 值

UAFxType 操作的具体细节通过编码为 json x-callback-url 参数的各种 JSON 值进行承载。

JSON 值	类型	描述
discoveryData	String	**DiscoveryData** JSON 结构
errorCode	short	操作的 **ErrorCode** 值。
message	String	依赖于 **UAFxType**，进行测试或处理的 **UAFMessage** 请求。

origin	String	为请求提供的一个 RFC6454 网页源[RFC6454]的字符串。
channelBindings	String	操作的通道绑定 JSON 结构。
responseCode	short	ServerResponse 的 uafResult 字段。

根据 UAFxType x-callback-url 操作的值，下表列出了其他需要出现的 JSON 值：

UAFIntentType 值	discoveryData	errorCode	message	origin	channelBindings	responseCode
"DISCOVER"						
"DISCOVER_RESULT"	可选	必须				
"CHECK_POLICY"			必须	可选		
"CHECK_POLICY_RESULT"		必须				
"UAF_OPERATION"			必须	可选	必须	
"UAF_OPERATION_RESULT"		必须	可选			
"UAF_OPERATION_COMPLETION_STATUS"			必须			必须

7.2.1 DISCOVER

该操作调用 FIDO UAF 客户端来发现可用的认证器及功能。FIDO UAF 客户端通常不会显示与处理该操作相关的用户界面，而是仅仅返回结果 JSON 结构。

然而，请求应用程序不能依赖于此，因为出于私密性的考虑，FIDO UAF 客户端可能会显示一个用户界面，让用户选择是否向请求应用程序公开或具体公开哪个（哪些）认证器。

> 注释
>
> IOS 自定义 URL 方案处理程序总是为每一个请求和响应要求一个应用程序跳

> 转，即使没有任何用户界面被显示。

7.2.2 DISCOVER_RESULT

当接收到 DISCOVER 类型的 x-callback-url 操作时，FIDO UAF 客户端应该将带有该类型的操作作为参数返回。

如果 resultCode 是 RESULT_OK，并且 errorCode 的 JSON 值是 NO_ERROR，那么该操作具有一个 JSON 值------ discoveryData，它是一个 **DiscoveryData** 格式的 JSON 字符串，指明可用的认证器和功能。

7.2.3 CHECK_POLICY

在不提示用户的情况下，该操作调用 FIDO UAF 客户端来发现它是否可以处理提供的消息。

处理该操作的相关 Action 不得向用户显示界面。

> 注释
>
> IOS 自定义 URL 方案处理程序总是为每一个请求和响应要求一个应用程序跳转，即使没有任何用户界面被显示。

该 x-callback-url 操作需要如下的 JSON 值：

- message，是一个 **UAFMessage** 类型的字符串，包含需要测试的请求消息。
- origin，是一个可选的 JSON 值，使调用者提供一个 RFC6454 源[RFC6454]字符串来代替应用程序本身的身份。

7.2.4 CHECK_POLICY_RESULT

当接收到 CHECK_POLICY 类型的 x-callback-url 操作时，FIDO UAF 客户端应该将该操作返回。

除了返回 resultCode 外，该 x-callback-url 操作含有一个 JSON 值为 errorCode，包含一个 **ErrorCode** 值，指明具体的错误情况，或者当 FIDO UAF 客户端能够处理消息时，返回 NO_ERROR。

7.2.5 UAF_OPERATION

该操作调用 FIDO UAF 客户端来处理提供的请求消息，并且返回一个准备发给 FIDO UAF 服务器的响应消息。发送者应该假设 FIDO UAF 客户端会显示一个用户界面让用户来操纵该 x-callback-url 操作，例如，提示用户完成校验过程。

该 x-callback-url 操作需要下列 JSON 值：

- message，是一个 **UAFMessage** 类型的字符串，包含需要处理的请求消息。
- channelBindings，包含 FIDO UAF 协议规范[UAFProtocol]中定义的 ChannelBinding（通道绑定）结构的 JSON 字符串的表达形式。
- origin，是一个可选的 JSON 值，使调用者提供一个 RFC6454 源[RFC6454]字符串来代替应用程序本身的身份。

7.2.6 UAF_OPERATION_RESULT

当接收到 UAF_OPERATION 类型的 x-callback-url 操作时，FIDO UAF 客户端应该将该 x-callback-url 操作返回。

如果 resultCode 是 RESULT_CANCELLED，该 x-callback-url 操作将含有一个 JSON 值为 errorCode，包含指示特定错误情况的 **ErrorCode** 值。

如果 resultCode 是 RESULT_OK，并且 errorCode 的 x-callback-url JSON 值是 NO_ERROR，那么这个 x-callback-url 操作含有一个 JSON 值为 message，包含 **UAFMessage** 类型的字符串，将作为 UAF 协议响应消息发送给 FIDO 服务器。

7.2.7 UAF_OPERATION_COMPLETION_STATUS

该 x-callback-url 操作必须发送给 FIDO UAF 客户端，来指示发送给远程服务器的 FIDO UAF 消息的处理状态，该操作是非常重要的，特别是当一个新的注册请求被客户端挂起时，直到服务器允许其注册才能解除挂起状态。

7.3 iOS 实现的开发指导

每个基于 iOS 自定义 URL 的请求都会导致用户可察觉的 APP 到 FIDO UAF 客户端的上下文转换，反之亦然。当调用 DISCOVER 和 CHECK_POLICY 请求时这种转换最为明显，因为这些请求是在没有用户参与的情况下自动调用的。这种上下文转换会影响用户体验，因此，建议避免这两种请求方式并且在集成 FIDO 时必使用他们。

7.4 iOS 实现的安全性要求

本节是非规范性的。

在 iOS 中自定义 URLs 的安全性考虑是任何应用都可以注册任何自定义的 URL。如果多个应用程序注册相同的自定义 URL，在 iOS 中处理 URL 调用的行为是未定义的。

从 FIDO UAF 客户端来看，自定义的 URL 方案处理程序的问题是通过使用提供了 URL 发起者的 bundle ID 的 sourceApplication 参数解决的。只要设备没有被越狱，并且苹果公司会不断地审核应用商店中的应用是否是具有欺骗性 bundle ID 的恶意软件，那么这种方法是有效的。sourceApplication 参数可以与类型标识符列表中的项进行匹配，来确保请求应用程序是被准许使用依赖方的证书的。

从依赖方应用来看，加密的方法可以阻止恶意软件欺骗依赖方应用的请求 URL。在每一次请求时，依赖方应用生成随机的加密密钥并发给 FIDO 客户端。然后 FIDO 客户端使用该密钥加密响应消息。在这种情况下，只有依赖方应用可以解密响应消息。即使恶意软件能够欺骗依赖方应用的请求 URL 并拦截响应消息，它也不能对其进行解码。

为了防护潜在的恶意应用程序注册它们自己来处理 FIDO UAF 客户端自定义 URL 方案的攻击，依赖方应用程序可以获取响应应用程序的 bundle ID，并在鉴别或交易事件中，利用它来进行风险管理决策。例如，依赖方可能会维护一个已知恶意软件的 bundle ID 的列表，并且拒绝接受这些客户端完成的操作；或者维护一份风险评估为首选的已知优秀应用程序的 bundle ID 列表。

8. 传输绑定概述

本节是规范性的。

本节介绍了客户端应用程序如何传输 FIDO UAF 协议消息的一般规范性安全要求，给出了传输层安全[TLS]的特定要求，并且还介绍了在 TLS[RFC2818]上使用 HTTP 与 FIDO UAF 协议的互操作性的简介。

8.1 传输安全要求

本节是非规范性的。

UAF 协议没有内在的手段来识别依赖方服务器，也没有内在的手段来进行 UAF 协议消息的端到端保护。为了进行安全的 UAF 协议交换，需要满足下面的抽象要求：

1. 客户端应用程序必须安全鉴别服务器终端的授权，从客户端角度来看，代表了客户端应用程序发送给 FIDO UAF 客户端的网页源[RFC6454] (方案:主机:端口的三元组)。多数情况下通过使用 TLS 来验证服务器证书的合法性、宣称正确的 DNS 名称、链接到客户端平台信任的根证书上等方式来解决。客户端也可能使用其它的方法来鉴别服务器，例如，通过跟随应用程序分发的预先配置证书或密钥，或者通过像 Kerberos [RFC4120]一样的其他网络鉴别协议。
2. UAF 协议消息的传输机制必须保证消息的保密性，从而防止向未授权的第三方泄露其内容。该保护措施应该与上文中描述的服务器标识的证明进行密码学绑定。
3. UAF 协议消息的传输机制必须保护消息的完整性，防止未授权的第三方

的篡改。该保护措施应该与上文中描述的服务器标识的证明进行密码学绑定。

8.2 TLS 安全要求

本节是非规范性的。

如果通过 TLS([RFC2246] [RFC4346], [RFC5246] or [TLS13draft02])使用 HTTP 来传输 UAF 协议消息，需要满足下列特定的要求：

1. 如果有 TLS 错误，不管是"warning"还是"fatal"还是任何其它的 TLS 连接错误，HTTP 客户端必须终止连接，并且无需提示用户。例如包括 HTTP 客户端进行证书有效性校验的过程中出现的任何错误，例如通过 TLS 服务器标识校验[RFC6125]、证书吊销列表（CRLs）[RFC5280]，或者通过在线证书状态协议（OCSP）[RFC2560]。
2. 无论何时对显示的 TLS 服务器标识（在 TLS 握手时显示的，通常在服务器证书中）和预期的原始 TLS 服务器标识（例如，用户输入的，或嵌入在链接中的）进行对比时，[RFC6125]必须进行服务器标识校验。出现任何错误情况，客户端必须马上终止连接，并且不需要提示用户。
3. TLS 服务器证书必须明确规定为带外的（例如，作为"固定证书"与 app 一起打包）；或通过链接到操作系统或浏览器的证书库中的根证书的方式来被信任，以此来满足他们的根证书存储方案要求。如果在建立信任链时出现任何错误的情况，客户端必须马上终止连接，并且不需要提示用户。
4. 不允许使用"anon"和"null"加密套件，并且应该避免使用 TLS 中的不安全的密码学算法(例如，MD4, RC4, SHA1)（参考 NIST SP800-131A [SP800-131A]）。
5. 客户端和服务器应该使用最新的可用的 TLS 版本。
6. 客户端应该提供通道绑定信息，并且服务器应该验证所有的通道绑定信息是否可用，包括一个 ChannelID[ChannelID]公钥，tls-unique 和 tls-server-end-point 的绑定[RFC5929]，以及 TLS 服务器证书绑定[UAFProtocal]。这些信息防护了某些类型的网络攻击，也会阻止协议消息的转发，并且

服务器可以拒绝缺少信道绑定或信道绑定错误的消息。

8.3 HTTPS 传输互操作性概况

本节是规范性的。

符合规范的应用程序可能支持本节的内容。

高度优化的复杂应用程序使用 UAF 时，经常会将 UAF 协议消息嵌入到其他应用程序协议消息中进行传输。这里定义使用 HTTPS 传输 UAF 协议消息的概况，是为了：

- 提供一个互用性概况，使不同厂商开发的客户端应用程序库与服务器端的实现，能够更容易地整合为支持 FIDO UAF 的产品。
- 详细地说明传输层和 HTTP 接口的特定的必要安全特性，特别是当他们与浏览器承载的应用程序进行交互时。
- 此概况也可以在本文档附录示例中使用。本概况在实现中是可选的。本节使用 RFC 2119 关键字来指出开发时必要的安全性和其它为了使用该概况来互操作的属性。

> **注释**
>
> 某些 FIDO UAF 操作，特别是交易确认，会始终需要应用程序特有的实现。该互操作性概况仅仅提供了一个结构框架，适合于替换用户名/口令认证。

8.3.1 包含 UAF 请求消息

支持 UAF 的网页应用可能通常将请求消息作为包含其他应用内容的响应主体的一部分发送，例如，在一个脚本块中：

例 9

```
...
```

```
<script type=”application/json”>
{“initialRequest”: {
// initial request message here
},
“lifetimeMillis”: 60000; // hint: this initial request is valid for 60 seconds
}
</script>
...
```

然而，请求消息的有效时间是有限的，并且一个已安装的应用程序不能与请求一起传送，所以客户端应用程序通常需要能够检索新请求的能力。

当通过 HTTPS，使用 XMLHttpRequest [XHR]或其他的 HTTP API 发送请求消息时：

1. 服务器终端的 URI，以及与客户端通信的方式，根据不同的应用程序是不同的。
2. 客户端必须将 HTTP 方法设为 POST。[RFC7231]
3. 客户端必须将 HTTP“Content-Type”报头设为“application/fido+uaf; charset=utf-8”。[RFC7231]
4. 客户端应该在 HTTP“Accept”报头中，包含“application/fido+uaf”作为媒体类型。[RFC7231]
5. 客户端可能需要提供附加的报头（诸如一个 HTTP Cookie [RFC6265]），以一个应用程序特有的方式展示它执行请求的授权。
6. 完整的 POST 主体必须完全由 GetUAFRequest 结构描述的 JSON [ECMA-404]结构构成。
7. 服务器响应应该将 HTTP “Content-Type”设为“application/fido+uaf; charset=utf-8”。
8. 客户端应该使用 UTF-8 解码响应比特字符串，并且有错误处理。[HTML5]

9. 解码后的响应主体必须完全由 ReturnUAFRequest 接口描述的 JSON 结构构成。

8.3.2 Operation 枚举

描述了 FIDO UAF 消息的操作类型或请求一个消息。

```
WebIDL
enum Operation {
    "Reg",
    "Auth",
    "Dereg"
};
```

枚举描述	
Reg	注册
Auth	鉴别或交互确认
Dereg	注销

8.3.3 GetUAFRequest 结构

```
WebIDL
dictionary GetUAFRequest {
    Operation      op;
    DOMString      previousRequest;
    DOMString      context;
};
```

8.3.3.1 *GetUAFRequest* 结构成员

op 类型为 *Operation*

期望的 UAF 请求消息的类型。允许的字符串值在操作枚举中有定义。此字段是可选的，但是如果服务器不能通过其它上下文知道该操作的话，就必须设置它，例如，一个特定于操作的 URL 终端。

previousRequest 类型为 DOMString

如果应用程序因为上一个请求过期，而请求一个新的 UAF 请求消息时，该可选关键字可以包含前一个请求来协助服务器定位任意需要与新的请求消息重新关联的状态（如果之前发送过一个相关请求）。

context 类型为 DOMString

任何服务器生成正确的请求消息时可能有用或必须的附加上下文信息。该关键字是可选的，并且此数据的格式和性质是根据应用程序而不同的。

8.3.4 ReturnUAFRequest 结构

```
WebIDL
dictionary ReturnUAFRequest {
    required unsigned long      statusCode;
    DOMString                   uafRequest;
    Operation                   op;
    long                        lifetimeMillis;
};
```

8.3.4.1 *ReturnUAFRequest* 结构成员

statusCode 类型为 required unsigned long

该操作的 UAF 状态码（参考 3.1 UAF 状态码）。

uafRequest 类型为 DOMString

可选，当服务器决定生成一个新 UAF 请求消息时使用。

op 类型为 *Operation*

为客户端提供的一个可选的消息操作类型提示。如果服务器返回的是一个与请求不同的类型，那么该项是有用的。例如，如果与一个鉴别请求相关的密钥不再有效，服务器可能会返回一个注销消息。允许的字符串值在操作枚举中有定义。

lifetimeMillis 类型为 long

当服务器返回一个 uafRequest 时，该项是一个可选的提示用来通知客户端应用程序该消息的生命周期（以毫秒为单位）。

8.3.5 SendUAFResponse 结构

```
WebIDL
dictionary SendUAFResponse {
    required DOMString      uafResponse;
    DOMString               context;
};
```

8.3.5.1 *SendUAFResponse* 结构成员

uafResponse 类型为 required DOMString

UAF 响应消息。它必须被设置为 FIDO UAF 客户端返回的 UAFMessage.uafProtocolMessage。

context 类型为 DOMString

任何服务器生成正确的请求消息时可能有用或必须的附加环境信息。该关键字是可选的，并且此数据的格式和性质是根据应用程序而不同的。

8.3.6 发送 UAF 响应

尽管这不是唯一的可行模式，但是异步的 HTTP 请求是一个有效的方法，为网页应用程序或独立的应用程序向远程服务器发送 UAF 响应。

当通过 HTTPS，使用 XMLHttpRequest [XHR]或其他 API 发送响应消息时：

1. 服务器端的 URL 以及与客户端通信的方式，根据不同的应用程序是不同的。
2. 客户端必须将 HTTP 方法设为 POST。[RFC7231]
3. 客户端必须将 HTTP“Content-Type”报头设为“application/fido+uaf; charset=utf-8”。[RFC7231]
4. 客户端应该在 HTTP“Accept”报头中，包含“application/fido+uaf”作为媒体类型。[RFC7231]
5. 客户端可能需要提供附加的报头（诸如一个 HTTP Cookie [RFC6265]） ，以一个应用程序特有的方式展示它执行请求的授权。
6. 完整的 POST 主体必须完全由 GetUAFResponse 结构描述的 JSON

[ECMA-404]结构构成。

7. 服务器响应应该将 HTTP “Content-Type”设为“application/fido+uaf; charset=utf-8”。 响应主体必须完全由 ServerResponse 接口描述的 JSON 结构构成。

8.3.7 ServerResponse 接口

ServerResponse 接口代表成功处理注册、鉴别或交易确认操作后的完成状态和应用程序特有的附加数据。该消息不是 UAF 协议的规范部分，但是 statusCode 应该被发送到 FIDO UAF 客户端，使用 notifyUAFResult()操作进行“内务处理”服务。

WebIDL

```
interface ServerResponse {
    readonly    attribute int           statusCode;
    [Optional]
    readonly    attribute DOMString     description;
    [Optional]
    readonly    attribute Token[]       additionalTokens;
    [Optional]
    readonly    attribute DOMString     location;
    [Optional]
    readonly    attribute DOMString     postData;
    [Optional]
    readonly    attribute DOMString     newUAFRequest;
};
```

8.3.7.1 属性

statusCode 类型为 int，只读

FIDO UAF 响应状态码。需要注意的是，这个状态码描述的是处理隧道中 UAF 操作的结果，而不是外部 HTTP 传输的状态码。

description 类型为 DOMString，只读

描述状态码或向用户提供附加信息的详细消息。

additionalTokens 类型为 *Token* 数组，只读

此关键字包含给客户端的新的鉴别或授权令牌，它们不能被 HTTP 传输本身处理。应该在处理 location 之前处理此令牌。

location 类型为 DOMString，只读

如果出现该值，则提示客户端网页应用程序在处理完其他令牌之后，应该将 Document 上下文定位到该字段中包含的 URI。

postData 类型为 DOMString，只读

如果该值与 location 一起出现，则提示客户端在处理完其他令牌之后，应该将内容 POST 到指定的位置。

newUAFRequest 类型为 DOMString,

服务器可能会用它来返回一个新的 UAF 协议消息。该消息可能被用来为暂时失效的响应提供一个新的请求来重试操作，或者为交易请求追加确认，或者在永久性失败响应中发送注销消息。

8.3.8 Token 接口

> 注释
>
> UAF 服务器不负责创建附加的令牌作为 UAF 响应的一部分。这些令牌是为了在响应成功的 UAF 操作时，为依赖方应用程序提供更新客户端鉴别/授权状态的方法。例如，这些字段能够用来让 UAF 作为一个联合协议的初始鉴别步骤，但是任何这种联合的适用范围和细节超出了 UAF 的讨论范围。

WebIDL

```
interface Token {
    readonly    attribute TokenType   type;
    readonly    attribute DOMString   value;
};
```

8.3.8.1 属性

type 类型为 *TokenType*，只读

附加鉴别/授权令牌的类型。

value 类型为 DOMString，只读

附加鉴别/授权令牌的字符串值。

8.3.9 TokenType 枚举

WebIDL

```
enum TokenType {
    "HTTP_COOKIE",
    "OAUTH",
    "OAUTH2",
    "SAML1_1",
    "SAML2",
    "JWT",
    "OPENID_CONNECT"
};
```

枚举描述	
HTTP_COOKIE	如果用户代理是标准网页浏览器或其它的具有 cookie 存储的 HTTP 本地客户端，不得使用 TokenType。Cookies 应该直接用 Set-Cookie HTTP 报头来设置，提供给用户代理进行处理。对于非 HTTP 或非浏览器的环境，该值表示一个用来设为 HTTP cookie 的令牌。[RFC6265]例如，一个使用 UAF 进行鉴别的内置 VPN 客户端可能会使用该 TokenType 来自动向浏览器 cookie 库（jar）添加一个 cookie。
OAUTH	表示令牌是 OAUTH 类型的。[RFC5849]。
OAUTH2	表示令牌是 OAUTH2 类型的。[RFC6749]。
SAML1_1	表示令牌是 SAML1.1 类型的。[SAML11]。
SAML2	表示令牌是 SAML2.0 类型的。[SAML2-CORE]。
JWT	表示令牌是 JSON Web Token(JWT)类型的[JWT]。
OPENID_CONNECT	表示令牌是一个 OppenID Connect“id-token”。[OpenIDConnect]。

8.3.10 安全性考虑

本节是非规范性的。

非常重要的是，客户端设定以及服务器要求的方法必须是 POST，“Content-Type”HTTP 报头必须是正确的值。因为响应主体是有效的 ECMAScript，可以阻止未经授权的跨源访问，所以服务器一定不能响应能够使用 script 标签生成的请求类型，例如<script src=”https://example.com/fido/uaf/getRequest”>。用户代理使用这种嵌入方法生成的请求不能设置自定义的报头。

同样的，通过要求自定义的“Content-Type”报头，在没有触发跨域资源共享（CORS）访问预检时，跨源请求不能用 XMLHttpRequest[XHR]生成 [CORS]。

因为 FIDO UAF 消息只有在使用相同源时才是有效的，所以服务器不应该在响应时提供“Access-Control-All-Origin”[CORS]报头使非同源的内容读取到它。

为了阻止一些跨源的基于浏览器的分布式拒绝服务攻击，请求终端不需要再做其他操作，应该直接忽略有“Access-Control-Request-Method” [CORS]HTTP 报头或者有不正确的“Content-Type”HTTP 报头的请求。

如果服务器选择去响应以 GET 方法发出的并且没有附带自定义“Content-Type”报头的请求，应该使用一个前缀字符串，例如在所有主体中使用“while(1)”；或者 “&&&BEGIN_UAF_RESPONSE&&&” 来响应，从而阻止通过嵌入跨源<script>标签来被读取。在解析 JSON 内容之前，合法的同源调用者需要（并且仅能够）取出前缀字符串。

A. 参考文献

A.1 参考规范

[AndroidAppManifest]

Android App Manifest. Google, Inc., the Open Handset Alliance and the Android Open Source Project (Work in progress)

URL:http://developer.android.com/guide/topics/manifest/manifest-intro.html

[ChannelID]

D. Balfanz *Transport Layer Security (TLS) Channel IDs*. (Work In Progress) URL: http://tools.ietf.org/html/draft-balfanz-tls-channelid

[DOM]

Anne van Kesteren; Aryeh Gregor; Ms2ger; Alex Russell; Robin Berjon. *W3C DOM4*. 10 July 2014. W3C Last Call Working Draft. URL:http://www.w3.org/TR/dom/

[ECMA-262]

ECMAScript Language Specification, Edition 5.1. June 2011. URL:http://www.ecma-international.org/publications/standards/Ecma-262.htm

[ECMA-404]

The JSON Data Interchange Format. 1 October 2013. Standard. URL: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[FIDOGlossary]

R. Lindemann, D. Baghdasaryan, B. Hill, J. Hodges, *FIDO Technical Glossary*. FIDO Alliance Proposed Standard. URLs:
HTML: fido-glossary-v1.0-ps-20141208.html
PDF: fido-glossary-v1.0-ps-20141208.pdf

[HTML5]

Ian Hickson; Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Edward O'Connor; Silvia Pfeiffer. *HTML5*. 28 October 2014. W3C Recommendation. URL: http://www.w3.org/TR/html5/

[JWT]

M. Jones; J. Bradley; N. Sakimura. *JSON Web Token (JWT)*. 6 July 2012. Internet Draft. URL: http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-01

[OpenIDConnect]

OpenID Connect. OpenID Foundation (Work in Progress)

URL:http://openid.net/connect/

[PNG]

Tom Lane. *Portable Network Graphics (PNG) Specification (Second Edition)*. 10 November 2003. W3C Recommendation.

URL:http://www.w3.org/TR/PNG

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL:https://tools.ietf.org/html/rfc2119

[RFC2397]

L. Masinter. *The "data" URL scheme*. August 1998. Proposed Standard.

URL: https://tools.ietf.org/html/rfc2397

[RFC2818]

E. Rescorla. *HTTP Over TLS*. May 2000. Informational.

URL:https://tools.ietf.org/html/rfc2818

[RFC4648]

S. Josefsson, *The Base16, Base32, and Base64 Data Encodings (RFC 4648)*, IETF, October 2006, URL:http://www.ietf.org/rfc/rfc4648.txt

[RFC5849]

E. Hammer-Lahav, *The OAuth 1.0 Protocol (RFC 5849)*, IETF, April 2010, URL: http://www.ietf.org/rfc/rfc5849.txt

[RFC5929]

J. Altman, N. Williams, L. Zhu, *Channel Bindings for TLS (RFC 5929)*, IETF, July 2010, URL: http://www.ietf.org/rfc/rfc5929.txt

[RFC6125]

P. Saint-Andre, J. Hodges, *Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS) (RFC 6125)*, IETF, March 2011, URL:http://www.ietf.org/rfc/rfc6125.txt

[RFC6265]

A. Barth. *HTTP State Management Mechanism*. April 2011. Proposed Standard. URL: https://tools.ietf.org/html/rfc6265

[RFC6454]

A. Barth. *The Web Origin Concept*. December 2011. Proposed Standard. URL: https://tools.ietf.org/html/rfc6454

[RFC6749]

D. Hardt, Ed., *The OAuth 2.0 Authorization Framework (RFC 6749)*, IETF, October 2012, URL: http://www.ietf.org/rfc/rfc6749.txt

[RFC7230]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7230

[RFC7231]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7231

[SAML11]

E. Maler, P. Mishra and R. Philpott, *The Security Assertion Markup Language (SAML) v1.1*. OASIS, October 2003, URL:https://www.oasis-open.org/standards#samlv1.1

[SAML2-CORE]

Scott Cantor; John Kemp; Rob Philpott; Eve Maler. *Assertions and Protocols for SAML V2.0* 15 March 2005. URL: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[UAFAuthnrMetadata]

B. Hill, D. Baghdasaryan, J. Kemp, *FIDO UAF Authenticator Metadata Statements v1.0*. FIDO Alliance Proposed Standard. URLs:

HTML: fido-uaf-authnr-metadata-v1.0-ps-20141208.html

PDF: fido-uaf-authnr-metadata-v1.0-ps-20141208.pdf

[UAFProtocol]

R. Lindemann, D. Baghdasaryan, E. Tiffany, D. Balfanz, B. Hill, J. Hodges, *FIDO UAF Protocol Specification v1.0*. FIDO Alliance Proposed Standard. URLs:

HTML: fido-uaf-protocol-v1.0-ps-20141208.html

PDF: fido-uaf-protocol-v1.0-ps-20141208.pdf

[UAFRegistry]

R. Lindemann, D. Baghdasaryan, B. Hill, *FIDO UAF Registry of Predefined Values*. FIDO Alliance Proposed Standard. URLs:

HTML: fido-uaf-reg-v1.0-ps-20141208.html

PDF: fido-uaf-reg-v1.0-ps-20141208.pdf

[WebIDL-ED]

Cameron McCormack, *Web IDL*, W3C. Editor's Draft 13 November 2014. URL: http://heycam.github.io/webidl/

A.2 参考资料

[ANDROID]

The Android™ Operating System. Google, Inc., the Open Handset Alliance and the Android Open Source Project (Work in progress)

URL: http://developer.android.com/

[Android5Changes]

Android 5.0 Changes. Google, Inc., the Open Handset Alliance and the Android Open Source Project (Work in progress)

URL:http://developer.android.com/about/versions/android-5.0-changes.html

[CORS]

Anne van Kesteren. *Cross-Origin Resource Sharing*. 16 January 2014. W3C Recommendation. URL: http://www.w3.org/TR/cors/

[RFC2045]

N. Freed; N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. November 1996. Draft

Standard. URL: https://tools.ietf.org/html/rfc2045

[RFC2246]

T. Dierks, E. Rescorla, *The TLS Protocol Version 1.0*, IETF, January 1999, URL: http://www.ietf.org/rfc/rfc2246.txt

[RFC2560]

M. Myers; R. Ankney; A. Malpani; S. Galperin; C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. June 1999. Proposed Standard. URL:https://tools.ietf.org/html/rfc2560

[RFC4120]

C. Neuman, T. Yu, S. Hartman, K. Raeburn, *The Kerberos Network Authentication Protocol (V5) (RFC 4120)*, IETF, July 2005, URL:http://www.ietf.org/rfc/rfc4120.txt

[RFC4346]

T. Dierks, E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.1*, IETF, April 2006, URL: http://www.ietf.org/rfc/rfc4346.txt

[RFC5246]

T. Dierks, E. Rescorla, *The Transport Layer Security (TLS) Protocol*, IETF, August 2008, URL: http://www.ietf.org/rfc/rfc5246.txt

[RFC5280]

D. Cooper, S. Santesson, s. Farrell, S.Boeyen, R. Housley, W. Polk;*Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, IETF, May 2008, URL:http://www.ietf.org/rfc/rfc5280.txt

[SOP]

Same Origin Policy for JavaScript. Mozilla Developer Network, January 2014 URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Same_origin_policy_for_JavaScript

[SP800-131A]

E. Barker, A. Roginsky, *NIST Special Publication 800-131A: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*. National Institute of Standards and Technology, January 2011,

URL:http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

[TLS13draft02]

T. Dierks, E. Rescorla, *The Transport Layer Security (TLD) Protocol Version 1.3 (draft 02)*, IETF, July, 2014,

URL:https://tools.ietf.org/html/draft-ietf-tls-tls13-02

[UAFASM]

D. Baghdasaryan, J. Kemp, R. Lindemann, B. Hill, R. Sasson, *FIDO UAF Authenticator-Specific Module API*. FIDO Alliance Proposed Standard.

URLs:

HTML: fido-uaf-asm-api-v1.0-ps-20141208.html

PDF: fido-uaf-asm-api-v1.0-ps-20141208.pdf

[WebIDL]

Cameron McCormack. *Web IDL*. 19 April 2012. W3C Candidate Recommendation. URL: http://www.w3.org/TR/WebIDL/

[XHR]

Anne van Kesteren. *XMLHttpRequest*. Living Standard .

URL:https://xhr.spec.whatwg.org/

[webmessaging]

Ian Hickson. *HTML5 Web Messaging*. 1 May 2012. W3C Candidate Recommendation. URL: http://www.w3.org/TR/webmessaging/