互联网工程任务组（IETF）
征求意见：7519
类别：标准轨道J
ISSN：2070-

M.琼斯
微软
.布拉德利
1721平 身份
N.崎村
NRI

2015 年
5 月

JSON 网络令牌 （JWT）

摘要

JSON 网络令牌（JWT）是一种结构紧凑、URL 安全的方法，用于表示双方之间要传输的权利要求。 JWT 中的权利要求被编码为 JSON 对象，用作 JSON 网络签名（JWS）结构的有效载荷或 JSON 网络加密（JWE）结构的明文，从而能够对权利要求进行数字签名或使用消息验证码（MAC）进行完整性保护和/或加密。

本备忘录的现状

这是一份互联网标准跟踪文件。

本文档是互联网工程任务组 （IETF） 的产品。 它代表了 IETF 社区的共识。 它已接受公众审查，并已由互联网工程指导小组 （IESG）批准发布。 有关互联网标准的更多信息，请参阅 RFC 5741 第 2 节。

有关本文件的当前状态、任何勘误以及如何提供反馈的信息，请访问 http://www.rfc-editor.org/info/rfc7519。

版权声明

目录

1. 导言

JSON 网络令牌（JWT）是一种紧凑的请求表示格式，适用于空间受限的环境，如 HTTP 授权标头和 URI 查询参数。 JWT 将请求编码为 JSON [RFC7159] 对象进行传输，该对象可用作 JSON Web Signature（JWS）[JWS] 结构的有效载荷或 JSON Web Encryption（JWE）[JWE] 结构的明文，从而使请求能够通过数字签名或消息验证码（MAC）进行完整性保护和/或加密。 始终使用 JWS 紧凑序列化或 JWE 紧凑序列化表示。

JWT 的建议发音与英语单词 "jot "相同。

1.1. 符号约定

关键词 "MUST"、"MUST NOT"、"REQUIRED"、"SHALL"、"SHALL NOT"、"SHOULD"、"SHOULD NOT"、"RECOMMENDED"、"NOT RECOMMENDED"、"MAY "和 "SHOULD"。本文档中的 "OPTIONAL "应按照 "RFC 中用于表示要求级别的关键词"[RFC2119]中的描述进行解释。该解释仅适用于以大写字母出现的术语。

2. 术语

术语 "JSON Web 签名（JWS）"、"Base64url 编码"、"标头参数"、"JOSE 标头"、"JWS 紧凑序列化"、"JWS 有效载荷"、"JWS 签名 "和 "不安全的 JWS "由 JWS 规范 [JWS] 定义。

术语 "JSON Web Encryption (JWE)"、"Content Encryption Key (CEK)"、"JWE Compact Serialization"、"JWE Encrypted Key "和 "JWE Initialization Vector "由 JWE 规范 [JWE] 所定义。

密文"、"数字签名"、"消息验证码 (MAC) "和 "明文 "等术语由《互联网安全术语表，第 2 版》[RFC4949]定义。

这些术语由本规范定义：

JSON 网络令牌 (JWT)
以 JWS 或 JWE 编码的 JSON 对象形式表示一组权利要求的字符串，可对权利要求进行数字签名或 MAC 和/或加密。

JWT 索赔集
一个 JSON 对象，包含 JWT 传递的声明。

索赔
索赔是由索赔名称和索赔值组成的名称/值对。

索赔名称
索赔名称总是一个字符串。

索赔价值
索赔值可以是任何 JSON 值。

嵌套 JWT
在嵌套 JWT 中，JWT 分别用作外层 JWS 或 JWE 结构的有效载荷或明文值。

无担保 JWT
声明未受完整性保护或加密的 JWT。

防撞名称
名称空间中的名称，其分配方式使名称与其他名称发生碰撞的可能性极低：域名、ITU-T X.660和 X.670 Recommendation 系列中定义的对象标识符 (OID) 以及通用唯一标识符 (UUID) [RFC4122]。 在使用行政委托名称空间时，名称定义者需要采取合理的预防措施，确保他们控制着用于定义名称的名称空间部分。

StringOrURI
StringOrURI值作为区分大小写的字符串进行比较，不进行任何转换或规范化

数字日期

一个 JSON 数值，表示从 1970-01-01T00:00:00Z UTC 到指定 UTC 日期/时间的秒数、

这等同于 IEEE Std1003.1, 2013 Edition [POSIX.1] 定义的 "Seconds Since the Epoch"（自纪元起的秒数），其中每一天正好用 86400 秒来表示，但可以表示非整数值。 有关日期/时间特别是 UTC 的详细信息，请参见 RFC 3339 [RFC3339]

3. JSON 网络令牌 (JWT) 概述

这个JSON 对象就是JWT索赔集。

根据 RFC 7159 [RFC7159] 第 4 节，JSON 对象由零个或多个名称/值对（或成员）组成，其中名称是字符串，值是任意 JSON 值。 这些成员是 JWT 所代表的权利要求。

根据 RFC 7159[RFC7159] 第 2 节，该 JSON 对象可以在任何 JSON 值或结构字符前后包含空白和/或换行符

JWT 索赔集中的成员名称称为 "索赔名称"， 相应的值称为 "索赔值"。

JOSE 标头的内容描述了应用于 JWT 索赔集加密操作。 如果 JOSE 标头用于 JWS，则 JWT 表示为 JWS，索赔经过数字签名或 MAC 加密，JWT 索赔集为 JWS 有效负载。 如果 JOSE 标头是用于 JWE，则 JWT 表示为 JWE，并对权利要求进行加密，而 JWT 权利要求集则是 JWE 加密的明文。 JWT 可括入另一个 JWE 或 JWS 结构，以创建嵌套 JWT，从而能够执行嵌套签名和加密。

每个部分都包含一个 base64url 编码值。JWT 中的部分数量取决于使用 JWS 紧凑型序列化或 JWE 紧凑型序列化生成的 JWS 的表示形式。

3.1. JWT 示例

下面的 JOSE 标头示例声明编码对象是 JWT，JWT 是使用 HMAC SHA-256 算法进行 MAC 的 JWS：

```
{"typ": "JWT"、
 "算法": "HS256"}
```

为消除上述 JSON 对象表示法中可能存在的歧义，下面还包含了本例中 JOSE 标头实际使用的 UTF-8 表示法的八进制序列。（请注意，由于不同平台对换行符（CRLF 与 LF）的表示法不同、行首和行尾间距不同、最后一行是否有终止换行符以及其他原因可能会产生歧义。

在本例中使用的表示法中，第一行没有前导空格或尾部空格，第一行和第二行之间有一个 CRLF 换行符（13，10），第二行有一个前导空格（32），没有尾部空格，最后一行没有终止换行符）：

```
[123, 34, 116, 121, 112, 34, 58, 34, 74, 87, 84, 34, 44, 13, 10, 32,
34, 97, 108, 103, 34, 58, 34, 72, 83, 50, 53, 54, 34, 125]
```

Base64url 对 JOSE 标头的 UTF-8 表示形式的八位字节进行编码，就得到了这个编码后的 JOSE 标头值：

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
```

下面是一个 JWT 索赔集示例：

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

下面的八位位组序列就是 JWS 有效负载，它是本例中用于上述 JWT 索赔集的 UTF-8 表示形式：

```
[123, 34, 105, 115, 115, 34, 58, 34, 106, 111, 101, 34, 44, 13, 10,
32, 34, 101, 120, 112, 34, 58, 49, 51, 48, 48, 56, 49, 57, 51, 56,
48, 44, 13, 10, 32, 34, 104, 116, 116, 112, 58, 47, 47, 101, 120, 97,
109, 112, 108, 101, 46, 99, 111, 109, 47, 105, 115, 95, 114, 111,
111, 116, 34, 58, 116, 114, 117, 101, 125]
```

对 JWS 有效载荷进行 Base64url 编码后，就得到了这个编码后的 JWS 有效载荷（为便于显示，使用了换行符）：

```
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly
9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ
```

使用 HMAC SHA-256 算法计算编码后的 JOSE 标头和编码后的 JWS 有效载荷的 MAC，并按照 [JWS] 中规定的方式对 HMAC 值进行 base64url 编码，即可生成编码后的 JWS 签名：

```
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

按此顺序将这些编码部分连接起来，在各部分之间加上句点（'.'）字符，就得到了完整的 JWT（为便于显示，使用了换行符）：

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
.
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
.
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

JWS] 附录 A.1 更详细地说明了这种计算方法 。

4. JWT 索赔

JWT 索赔集内的索赔名称必须是唯一的；JWT 解析器必须拒绝具有重复索赔名称的 JWT或使用 JSON 解析器，根据 ECMAScript 5.1 [ECMAScript] 第 15.12 节（"JSON 对象"）的规定，只返回词法上最后一个重复的成员名称。

JWT 必须包含哪些声明才能被视为有效，这与上下文有关，不属于本规范的范围。JWT 的特定应用会要求实现以特定方式理解和处理某些声明。 但是，如果没有此类要求，则必须忽略实现无法理解的所有声明。

JWT 索赔名称分为三类：注册权利要求名称、公共权利要求名称和私有权利要求名称。

4.1. 注册的索赔名称

以下定义不是在所有情况下都必须使用或实施的，而是为一组有用的、可互操作的权利要求提供了一个起点。 使用 JWT 的应用程序应定义它们使用的具体权利要求，以及它们是必需的还是可选 所有名称都很简短，因为 JWT 的核心目标是使表示简洁。

4.1.1. "fs"（发行人）索赔

iss"（签发者）声明用于标识签发 JWT 的委托人。iss "值是一个区分大小写的字符串，包含一个 StringOrURI 值 。

4.1.2. "子"（主题）索赔

JWT 中的要求通常是关于主体的声明 主体值必须在签发者上下文中本地唯一性或全球唯一性 "sub "值是一个区分大小写的字符串，其中包含一个 StringOrURI 值。 该声明的使用是可选的。

4.1.3. "观众"（Audience）要求

每个打算处理 JWT 的委托人都必须用受众声明中的一个值来标识自己 如果当 "aud "声明存在时，处理声明的委托人没有用该声明中的一个值来标识自己，那么 JWT 必须被拒绝。 在一般情况下，"aud "值是一个区分大小写的字符串数组，每个字符串都包含一个 StringOrURI 在 JWT 只有一个受众的特殊情况下，"aud "值可以是一个区分大小写的字符串，其中包含一个 StringOrURI 值 受众值的解释通常与具体应用有关。本声明的使用是可选的。

4.1.4. "exp"（到期时间）索赔

exp"（过期时间）要求确定 JWT 在过期时间或过期时间之后不得接受处理的 过期时间。

其值必须是一个包含 NumericDate 值数字。此声明的使用是可选的

4.1.5. "nbf"（非之前）索赔

处理 "nbf "声明要求当前日期/时间必须在 "nbf "声明中列出的 "非之前 "日期/时间之后或相等于 "非之前 "日期/时间。 其值必须是一个包含 NumericDate 值的数字。 此声明的使用是可选的

4.1.6. "iat"（签发日期）索赔

iat" (issued at) 索赔标识 JWT 签发的时间。 该索赔可用于确定 JWT 的年龄。 其值必须是包含 NumericDate 值的数字 此索赔的使用是可选的

4.1.7. "jti"（JWT ID）声称

jti"（JWT ID）要求为 JWT 提供了一个唯一标识符。标识符值的分配方式必须能确保同一值被意外分配给不同数据对象的可能性微乎其微；如果应用程序使用多个签发者，还必须防止不同签发者产生的值之间发生碰撞。 "jti "权利要求可用于防止 JWT 被重放。 "jti "值是一个区分大小写的字符串 此权利要求的使用是可选

4.2. 公共索赔名称

不过， ，防止碰撞，任何新的权利要求名称都应在第 10.1 节建立的 IANA "JSON 网络令牌权利要求 "注册表中注册，或者是一个公共名称：一个包含防碰撞名称的值 在每种情况下，名称或值的定义者都需要采取合理的预防措施，以确保他们能够控制他们用来定义权利要求名称的名称空间部分

4.3. 私人索赔名称

JWT 的生产者和消费者可以同意使用属于私人名称权利要求名称：非注册权利要求名称（

第 4.1 节）或公共权利要求名称（第 4.2 节）的名称。 与公共要求不同

索赔名称、私人索赔名称可能会发生碰撞，应谨慎使用。

5. JOSE Header

根据JWT 是 JWS 还是 JWE，JOSE 标头值的相应规则也会适用。

本规范进一步规定了在 JWT 是 JWS 和 JWE 的情况下使用以下标头参数。

5.1. "typ"（类型）标头参数

[JWS]和[JWE]定义的 "typ"（类型）头参数由 JWT 应用程序使用，用于声明此完整 JWT 的媒体类型[IANA.MediaTypes ，当非 JWT 的值也可能出现在可包含 JWT 对象的应用程序数据结构中时，该值供 JWT 应用程序；应用程序可使用该值来区分可能存在的不同类型的对象。 JWT 实现会忽略该参数；JWT 应用程序会对该参数进行任何处理。 如果存在，建议其值为 "JWT"，以表明该对象是 JWT。 虽然媒体类型名称不区分大小写，但建议"JWT "始终使用大写字母拼写，以便与传统实现兼容
此标头参数的使用是可选的

5.2. "cty"（内容类型）标头参数

本规范使用 [JWS] 和 [JWE] 定义的 "cty"（内容类型）标头参数来传达 JWT 的结构信息。

在不使用嵌套签名或加密操作的正常情况下，不推荐使用此标头参数。 在使用嵌套签名或加密的情况下，必须存在此标头参数；在这种情况下，其值必须为 "JWT"，以表示此 JWT 中包含嵌套 JWT。 虽然媒体类型名称不区分大小写，但建议 "JWT "始终用大写字母拼写，以便与传统实现兼容 有关嵌套 JWT 的示例，请参阅附录 A.2嵌套 JWT 的示例。

5.3. 将索赔复制为标题参数

在某些使用加密 JWT 的应用程序中，对某些主张进行未加密表示是非常有用的。例如，在应用程序处理规则中，可以使用 来确定是否以及如何在解密前处理 JWT。

本规范允许 JWT 索赔集中的索赔根据应用程序的需要在作为 JWE 的 JWT 中作为标头参数复制。 如果存在这种复制的索赔，接收它们的应用程序应该验证它们的值是否相同，除非应用程序为这些索赔定义了其他特定的处理规则。 应用程序有责任确保只有那些可以安全地以未加密方式传输的索赔才作为标头参数值复制到 JWT 中。

本规范第 10.4.1 节注册了 "iss"（签发人）、"sub"（主体）和 "aud"（受众）标题参数名称，目的是在加密 JWT 中为需要这些名称的应用程序提供这些权利要求的未加密副本。 其他规范也可根据需要将已注册权利要求名称的其他名称注册为标题参数名称。

6. 无担保 JWT

为了支持 JWT 内容通过 JWT 中包含的签名和/或加密（如对包含 JWT 的数据结构进行签名）以外的其他方式进行加密的用例，也可以创建没有签名或加密的 JWT。无担保 JWT 是一种使用 "alg "头参数值 "none "且其 JWS 签名值为空字符串的 JWS，如 JWA 规范 [JWA] 所定义；它是一种以 JWT 索赔集作为 JWS 有效负载的无担保 JWS。

6.1. 无担保 JWT 示例

下面的 JOSE 标头示例声明编码对象是不安全的 JWT：

```
{"算法"："无"}
```

Base64url 对 JOSE 标头的 UTF-8 表示形式的八位字节进行编码，就得到了这个编码后的 JOSE 标头值：

```
eyJhbGciOiJub25lIn0
```

下面是一个 JWT 索赔集示例：

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

Base64url 对 JWT 索赔集的 UTF-8 表示形式的八位字节进行编码后，就得到了这个编码后的 JWS 有效负载（为便于显示，使用了换行符）：

```
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
```

编码后的 JWS 签名为空字符串。

按此顺序将这些编码部分连接起来，在各部分之间加上句点（'.'）字符，就得到了完整的 JWT（为便于显示，使用了换行符）：

```
eyJhbGciOiJub25lIn0
.
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
.
```

7. 创建和验证 JWT

7.1. 创建 JWT

要创建 JWT，需要执行以下步骤。在各步骤的输入和输出之间不存在依赖关系的情况下，步骤的顺序并不重要。

1. 创建一个 JWT 索赔集，其中包含所需的索赔。注意，表示法中明确允许留白，编码前无需执行规范化。
2. 让信息成为 JWT 索赔集的 UTF-8 表示形式的八位位组。
3. JWT 必须符合[JWS]或[JWE]规范。请注意，表示法中明确允许空白，编码前无需进行规范化。

4. 根据 JWT 是 JWS 还是 JWE，有两种情况：

   * 如果 JWT 是 JWS，则使用消息作为 JWS 有效载荷创建 JWS；必须遵循 [JWS] 中规定的创建 JWS 的所有步骤。

   * 否则，如果 JWT 是 JWE，则使用信息作为 JWE 的明文创建 JWE；必须遵循 [JWE] 中规定的创建 JWE 的所有步骤。

5. 如果要执行嵌套签名或加密操作，则让信息成为 JWS 或 JWE，然后返回步骤 3，在该步骤创建的新 JOSE 标头中使用 "cty"（内容类型）值 "JWT"。

6. 否则，让生成的 JWT 成为 JWS 或 JWE。

7.2. 验证 JWT

在验证 JWT 时，需要执行以下步骤。在步骤的输入和输出之间不存在依赖关系的情况下，步骤的 顺序并不重要。如果列出的任何步骤失败，则 JWT 必须被拒绝，即被应用程序视为无效输入。

1. 验证 JWT 是否至少包含一个句点（'.'）字符。

2. 让编码 JOSE 标头成为 JWT 中第一个句点（'.'）字符之前的部分。

3. Base64url 对编码后的 JOSE 标头进行解码，但不得使用换行符、空白或其他附加字符。

4. 验证生成的八进制数序列是符合 RFC 7159 [RFC7159] 的完全有效 JSON 对象的 UTF-8 编码表示；让 JOSE 标头成为该 JSON 对象。

5. 确认生成的 JOSE 标头只包含语法和语义均可理解和支持的参数和值，或指定在不理解时忽略的参数和值。

6. 使用 [JWE] 第 9 节中描述的任何方法确定 JWT 是 JWS 还是 JWE。

7. 根据 JWT 是 JWS 还是 JWE，有两种情况：

   * 如果 JWT 是 JWS，请按照 [JWS] 中规定的步骤验证 JWS 让信息成为对 JWS 有效负载进行 base64url 解码的结果。

   * 否则，如果 JWT 是 JWE，则按照 [JWE] 中规定的步骤验证 JWE 让信息成为生成的明文。

8. 如果 JOSE 标头的 "cty"（内容类型）值为 "JWT"，则消息是 JWT，是嵌套签名或加密操作的对象。 在这种情况下，使用消息作为 JWT ，返回步骤 1。

9. 否则，base64url 将按照不使用换行符、空白或其他附加字符的限制对报文进行解码。

10. 验证生成的八进制数序列是符合 RFC 7159 [RFC7159] 的完全有效的 JSON 对象的 UTF-8 编码表示；让 JWT 索赔集成为该 JSON 对象。

即使 JWT可以成功验证，除非 JWT 中使用的算法是应用程序可以接受的，否则它应该拒绝接受 J

7.3. 字符串比较规则

处理 JWT 不可避免地需要将已知字符串与 JSON 对象中的成员和值进行比较 例如，在检查算法时，将根据 JOSE 标头中的成员名称检查 Unicode [UNICODE] 字符串编码 "alg"，以查看是否存在匹配的标头参数名称。

RFC 7159 [RFC7159] 第 8.3 节描述了进行成员名比较的 JSON 规则。 由于执行的字符串比较操作只有相等和不等式，因此可以使用相同的规则将成员名和成员值与已知字符串进行比较。

在本规范 中，只有 "typ "和 "ty "成员值不使用这些比较规则。

应用程序可能会在大小写敏感值中包含大小写不敏感的信息，例如将 DNS 名称作为 "iss"（签发人）声明值的一部分在这种情况下，如果有多方可能需要生成相同的值以便进行比较，则应用程序可能需要定义用于表示大小写不敏感部分的规范大小写约定例如小写。 （但是，如果其他各方都使用生产方逐字输出的值，而不试图将其与独立生产的值进行比较，那么生产方使用的大小写就无关紧要了）。

8. 实施要求

，使用本规范的应用程序可对其使用的 实施提出额外要求例如，一个应用程序可能要求支持加密 JWT 和嵌套 JWT，而另一个应用程序可能要求支持使用 P-256 曲线和 SHA-256 哈希算法（"ES256"）的椭圆曲线数字签名算法 (ECDSA) 签名 JWT

在 JSON Web 算法 [JWA] 中指定的签名和 MAC 算法中，只有 HMAC SHA-256 算法（"HS256"）和 "无 "算法必须由符合要求的 JWT 实现来实现。 建议实施也支持使用 SHA-256 哈希算法（"RS256"）的 RSASSA-PKCS1-v1_5，以及使用 P-256 曲线和 SHA-256 哈希算法（"ES256"）的 ECDSA。 对其他算法和密钥大小的支持是可选的。

对加密 JWT 的支持是可选的。 如果实施提供了加密功能，那么在 [JWA] 中指定的加密算法中，只有使用 2048 位密钥的 RSAES-PKCS1-v1_5（"RSA1_5"）、使用 128 位和 256 位密钥的 AES Key Wrap（"A128KW "和 "A256KW以及使用 AES-CBC 和 HMAC SHA-2 的复合认证加密算法（"A128CBC-HS256 "和 "A256CBC-HS512"）必须支持。
由符合要求的实现来实施。 建议实施也支持使用椭圆曲线 Diffie-Hellman Ephemeral Static (ECDH-ES)来商定用于封装内容加密密钥的密钥（"ECDH-ES+A128KW "和 "ECDH-ES+A256KW"） ，以及使用 128 位和 256 位密钥的伽罗瓦/计数器模式 (GCM) 下的 AES（"A128GCM "和 "A256GCM"）。 对其他算法和密钥大小的支持为可选项。

对嵌套 JWT 的支持是可选的。

9. 用于声明内容是 JWT 的 URI

本规范注册 URN "urn:ietf:params:oauth:token-type:jwt"，供使用 URI（而不是媒体类型等）声明内容类型的应用程序使用，以表明所引用的内容是 JWT。

10. IANA 考虑因素

10.1. JSON 网络令牌索偿登记处

本节为 JWT 索赔名称建立 IANA "JSON Web 令牌索赔"注册表。 注册表记录索赔名称和定义该名称的规范的引用。 本节注册第 4.1 节中定义的索赔名称

根据一位或多位指定专家的建议，在 jwt-reg-review@ietf.org 邮件列表上进行为期三周的审查后，按 "规范要求"[RFC5226] 注册值。
不过，为了在公布前分配数值，指定专家一旦确信将公布此类规范，即可批准注册。

发送到邮件列表供审查的注册请求应使用适当的主题（如 "请求注册索赔：示例"）。

在审查期内，指定专家将批准或拒绝注册申请，并将此决定通知审查名单和 IANA。驳回决定应包括解释，并在适用情况下就如何使申请成功提出建议。
未确定时间超过以下期限的登记申请
21 天后可提请 IESG 注意（使用 iesg@ietf.org 邮件列表），以便解决。

指定专家应采用的标准包括确定拟议的注册是否与现有功能重复、是否可能具有普遍适用性或是否仅对单一应用有用，以及注册说明是否清晰。

IANA 必须只接受指定专家的注册更新，并应将所有注册请求发送到审查邮件列表。

建议指定多位指定专家，他们能够代表使用本规范的不同申请的观点，以便对注册决定进行广泛知情的审查。 在注册决定可能出现以下情况时

如果某位专家被认为存在利益冲突，该专家应服从其他专家的判断。

10.1.1. 注册模板 索赔名称：

所请求的名称（如 "iss"）。 由于本规范的核心目标是使生成的表述简洁，因此建议名称简短，即不要超过 8 个字符，且没有令人信服的理由。该名称区分大小写。除非指定专家声明有令人信服的理由允许例外，否则名称不得以不区分大小写的方式与其他注册名称匹配。

索赔说明：
索赔简介（如 "发行人"）。

变更控制员：

对于标准跟踪 RFC，请列出 "IESG"。 对于其他 RFC，请提供负责方的名称。也可包括其他详细信息（如邮政地址、电子邮件地址、主页 URI）

规格文件：

指定参数的一份或多份文件的引用，最好包括可用于检索文件副本的 URI。 也可包括相关章节的说明，但不是必需的。

10.1.2. 初始注册表内容

- 索赔名称："iss"
- 索赔说明：签发人
- 变更控制人：IESG
- 规范 文件：部分 4.1.1 的 RFC 7519

- 索赔名称："子"
- 索赔说明：主题
- 变更控制人：IESG
- 规范文件：部分 4.1.2 的 RFC 7519

- 索赔名称："审计"
- 索赔说明：受众
- 变更控制人：IESG
- 规范文件：部分 4.1.3 的 RFC 7519

- 索赔名称："exp"
- 索赔说明：过期时间
- 变更控制人：IESG
- 规范 文件：部分 4.1.4 的 RFC 7519

- 索赔名称："nbf"
- 索赔说明：之前没有
- 变更控制人：IESG
- 规范文件：部分 4.1.5 的 RFC 7519

- 索赔名称："iat"
- 索赔说明：签发日期
- 变更控制人：IESG
- 规范文件：部分 4.1.6 的 RFC 7519

- 索赔名称："jti"
- 索赔说明：JWT ID
- 变更控制人：IESG
- 规范文件：部分 4.1.7 的 RFC 7519

10.2. urn:ietf:params:oauth:token-type:jwt 的子命名空间注册

10.2.1. 注册表内容

本节在 "An IETF URN Sub-Namespace for OAuth" [RFC6755] 建立的 IANA "OAuth URI "注册表中注册了值 "token-type:jwt"，该值可用于表示内容是 JWT。

- URN: urn:ietf:params:oauth:token-type:jwt
- 通用名称：JSON 网络令牌 (JWT) 令牌类型
- 变更控制人：IESG
- 规范文件：RFC 7519

10.3. 媒体类型注册

10.3.1. 注册表内容

本节以 RFC 6838 [RFC6838] 中描述的方式在 "媒体类型 "注册表 [IANA.MediaTypes] 中注册了 "application/jwt "媒体类型 [RFC2046]，该类型可用于表明内容是 JWT。

- 类型名称：应用程序
- 子类型名称： jwt
- 所需参数：不适用
- 可选参数：无
- 编码注意事项：8 位；JWT 值编码为一系列 base64url 编码值（其中一些可能是空字符串），由句点（'.'）字符分隔。
- 安全注意事项：请参阅 RFC 7519 的安全考虑部分
- 互操作性考虑因素：不适用
- 已发布规范：RFC 7519
- 使用此媒体类型的应用程序：OpenID Connect、Mozilla Persona、Salesforce、Google、Android、Windows Azure、Amazon Web Services 及其他众多应用程序
- 片段标识符考虑因素：不适用
- 其他信息：

  魔法编号：不适用 文件扩展名：不适用
  Macintosh 文件类型代码：不适用

- 联系人和电子邮件地址，以获取更多信息：迈克尔-B-琼斯，mbj@microsoft.com
- 预期用途：通用
- 使用限制：无
- 作者：Michael B. Jones,迈克尔-B-琼斯，mbj@microsoft.com
- 变更控制人：IESG
- 临时注册? 没有

10.4. 标头参数名称注册

本节在 [JWS] 建立的 IANA "JSON 网络签名和加密标头参数 "注册中心注册第 4.1 节中定义的特定权利要求名称，以便根据第 5.3 节在 JWE 中作为标头参数复制的权利要求使

用。

10.4.1. 注册表内容

- 标头参数名称："iss
- 标头参数说明：发行者
- 标头参数使用位置：JWE
- 变更控制人：IESG
- 规范文件：RFC 7519 第 4.1.1 节

- 标题参数名称："子"
- 标题参数 说明：主题
- 标头参数使用位置：JWE
- 变更控制人：IESG
- 规范文件：RFC 7519 第 4.1.2 节

- 标题参数名称："aud
- 标题参数说明：受众
- 标头参数使用位置：JWE
- 变更控制人：IESG
- 规范文件：RFC 7519 第 4.1.3 节

11. 安全考虑因素

JWT/JWS/JWE/JWK 代理必须解决与任何加密应用相关的所有安全问题。 这些问题包括保护用户的非对称私人密钥和对称秘钥，以及对各种攻击采取反制措施。

特别是，[JWS] 第 10.12 节（"JSON 安全考虑因素 "）和第 10.13 节（"Unicode 比较安全考虑因素"）同样适用于 JWT 索赔集，与适用于 JOSE 标头的方式相同。

11.1. 信任决定

尤其是，用于签署和/或加密 JWT的密钥通常需要可验证处于 JWT 签发方的控制之下。

11.2. 签名和加密顺序

虽然从语法上讲，嵌套 JWT 的签名和加密操作可以按任意顺序进行，但如果同时需要签名和加密，通常情况下，制作者应先签名，然后

这样可以防止签名被剥离，只留下加密信息的攻击同时也为签名者提供了隐私保护。此外，在许多司法管辖区，加密文本上的签名被认为是无效的。

请注意，JWS 和 JWE 的基本规范已经解决了与签名和加密操作顺序有关的潜在安全问题；特别是，由于 JWE 只支持使用经过验证的加密算法，因此在许多情况下适用的加密后签名的潜在需求并不适用于本规范。

12. 隐私考虑因素

WT 可能包含对隐私敏感的信息。在这种情况下，必须采取措施防止这些信息泄露给非指定方实现这一目标的方法之一是使用加密的 JWT 并对接收方进行身份验证。

另一种方法是确保包含未加密隐私敏感信息的 JWT 只使用支持端点验证的加密协议（如传输层安全协议 (TLS)）进行传输。从 JWT 中省略隐私敏感信息是最大限度减少隐私问题的最简单方法。

13. 参考资料

13.1. 规范性参考文献

[ECMAScript] (英文)
Ecma International，《ECMAScript 语言规范》、5.1 版"，ECMA 标准 262，2011 年 6 月、<http://www.ecma-international.org/ecma-262/5.1/ECMA-262.pdf>。

[IANA.媒体类型]
IANA，"媒体类型"、<http://www.iana.org/assignments/media-types>。

[JWA]Jones ，M.，"JSON Web 算法（JWA）"，RFC 7518，DOI 10.17487/RFC7518，2015 年 5 月、<http://www.rfc-editor.org/info/rfc7518>。

[JWE] Jones、M. 和 J. Hildebrand，"JSONWeb Encryption(JWE)"，RFC 7516， DOI 10.17487/RFC7516，2015 年 5 月、<http://www.rfc-editor.org/info/rfc7516>。

[JWS] Jones，M.、Bradley，J.和 N. Sakimura，"JSON Web Signature（JWS）"，RFC 7515，DOI 10.17487/RFC，2015 年 5 月、<http://www.rfc-editor.org/info/rfc7515>。

[RFC20] Cerf，V.，"网络交换的 ASCII 格式"，STD80，RFC 20 DOI 10.17487/RFC0020，1969 年 10 月、<http://www.rfc-editor.org/info/rfc20>。

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046、DOI 10.17487/RFC2046，1996 年 11 月、<http://www.rfc-editor.org/info/rfc2046>。

[RFC2119]Bradner ，S.，"RFC 中用于指示要求级别的关键词"，BCP 14，RFC 2119、DOI 10.17487/RFC2119，1997 年 3 月、<http://www.rfc-editor.org/info/rfc2119>。

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66、RFC 3986，DOI 10.17487/RFC3986，2005 年 1 月、<http://www.rfc-editor.org/info/rfc3986>。

[RFC4949] Shirey，R.，"互联网安全词汇，第 2 版"，FYI 36，RFC 4949，DOI 10.17487/RFC4949，2007 年 8 月、<http://www.rfc-editor.org/info/rfc4949>。

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>。

[UNICODE] 统一码联盟，"统一码标准"、<http://www.unicode.org/versions/latest/>。

13.2. 信息参考

[CanvasApp] Facebook, "Canvas Applications", 2010、<http://developers.facebook.com/docs/authentication/canvas>。

[JSS] Bradley, J. and N. Sakimura (editor), "JSON Simple Sign",

September 2010, <http://jsonenc.info/jss/1.0/>。

[魔法签名]
Panzer, J., Ed., Laurie, B., and D. Balfanz, "Magic Signatures", January 2011、<http://salmon-protocol.googlecode.com/svn/trunk/draft-panzer-magicsig-01.html>。

[OASIS.saml-core-2.0-os]
Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, 2005 年 3 月、<http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>。

[POSIX.1] IEEE，"The Open Group Base Specifications Issue 7"，IEEE Std 1003.1，2013 年版，2013 年、<http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15>。

[RFC3275] Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, DOI 10.17487/RFC3275, March 2002、<http://www.rfc-editor.org/info/rfc3275>。

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet：时间戳"，RFC 3339，DOI 10.17487/RFC3339，2002 年 7 月、<http://www.rfc-editor.org/info/rfc3339>。

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005、<http://www.rfc-editor.org/info/rfc4122>。

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008、<http://www.rfc-editor.org/info/rfc5226>。

[RFC6755] Campbell、B. 和 H. Tschofenig，"An IETF URN Sub-Namespace for OAuth"，RFC 6755，DOI 10.17487/RFC6755，2012 年 10 月、<http://www.rfc-editor.org/info/rfc6755>。

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type

Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013、<http://www.rfc-editor.org/info/rfc6838>。

[SWT] Hardt, D. and Y. Goland, "Simple Web Token (SWT)", Version 0.9.5.1, November 2009, <http://msdn.microsoft.com/en-us/library/windowsazure/hh781551.aspx>。

[W3C.CR-xml11-20060816]
Cowan, J., "Extensible Markup Language (XML) 1.1 (Second Edition)", World Wide Web Consortium Recommendation REC-xml11-20060816, 2006 年 8 月、<http://www.w3.org/TR/2006/REC-xml11-20060816>.

[W3C.REC-xml-c14n-20010315]
Boyer, J., "Canonical XML Version 1.0", World Wide Web Consortium Recommendation REC-xml-c14n-20010315, March 2001, <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>.

附录 A. JWT示例

本节包含 JWT 示例。 有关其他 JWT 示例，请参阅本文档第 6.1 节和 [JWS] 附录 A.1 - A.3。

A.1. 加密 JWT 示例

本示例使用 RSAES-PKCS1-v1_5 和 AES_128_CBC_HMAC_SHA_256 加密向收件人发送与第 3.1 节中相同的请求。

下面的 JOSE 标头示例声明

- 内容加密密钥使用 RSAES-PKCS1-v1_5 算法加密后发送给接收方，生成 JWE 加密密钥。
- 使用 AES_128_CBC_HMAC_SHA_256 算法对明文进行验证加密，生成 JWE 密文和 JWE 验证标记。

```
{"alg":"RSA1_5","enc":"A128CBC-HS256"}
```

除了使用第 3.1 节中 JWT 索赔集的 UTF-8 表示形式的八位位组作为明文值外，该 JWT 的计算与第 3.2 节中 JWE 的计算完全相同。
JWE] 附录 A.2，包括使用的密钥。

本例中的最终结果（换行符仅供显示之用）是

```
eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.QR1Owv2ug2WyP
BnbQrRARTeEk9kDO2w8qDcjiHnSJflSdv1iNqhWXaKH4MqAkQtM
oNfABIPJaZm0HaA415sv3aeuBWnD8J-Ui7Ah6cWafs3ZwwFKDFUUsWHSK-IPKxLG
TkND09XyjORj_CHAgOPJ-Sd8ONQRnJvWn_hXV1BNMHzUjPyYwEsRhDhzjAD26ima
sOTsgruobpYGoQcXUwFDn7moXPRfDE8-NoQX7N7ZYMmpUDkR-Cx9obNGwJQ3nM52
YCitxoQVPzjbl7WBuB7AohdBoZOdZ24WlN1lVIeh8v1K4krB8xgKvRU8kgFrEn_a
1rZgN5TiysnmzTROF869lQ.
AxY8DCtDaGlsbGljb3RoZQ.MKOle7UQrG6nSxTLX6Mqwt0orbHvAKeWnDYvpIAeZ72deHx
z3roJDXQyhxx0wKaM HDjUEOKIwrtkHthpqEanSBNYHZgmNOV7sln1Eu9g3J8.
fiK51VwhsxJ-siBMR-YFiA
```

A.2. 嵌套 JWT 示例

在此示例中，JWT 索赔集首先被签名，然后被加

密。

内部签名 JWT 与 [JWS] 附录 A.2 中的示例相同，因此 这里不再重复计算。

RSAES-PKCS1-v1_5 和 AES_128_CBC_HMAC_SHA_256。

下面的 JOSE 标头示例声明

- 内容加密密钥使用 RSAES-PKCS1-v1_5 算法加密后发送给接收方，生成 JWE 加密密钥。
- 使用 AES_128_CBC_HMAC_SHA_256 算法对明文进行验证加密，生成 JWE 密文和 JWE 验证标记。
- 明文本身就是 JWT。

```
{"alg":"RSA1_5","enc":"A128CBC-HS256","cty":"JWT"}
```

Base64url 对 JOSE 标头的 UTF-8 表示形式的八位字节进行编码，就得到了这个编码后的 JOSE 标头值：

```
eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiY3R5IjoiSldUIn0
```

该 JWT 的计算方法与 [JWE] 附录 A.2 中 JWE 的计算方法相同，只是使用了不同的 JOSE标头、明文、JWE 初始化向量和内容加密密钥值 （使用的 RSA 密钥相同）

使用的明文是 [JWS] 附录 A.2.1 末尾 JWT 的 ASCII [RFC20] 表示法的八进制数（去掉所有空白和换行符），即 458 个八进制数的序列。

使用的 JWE 初始化向量值（使用 JSON 数组符号）为

```
[82, 101, 100, 109, 111, 110, 100, 32, 87, 65, 32, 57, 56, 48, 53,
50]
```

本例使用的内容加密密钥由下面的 base64url 编码值表示：

```
GawgguFyGrWKav7AX4VKUg
```

该嵌套 JWT 的最终结果（为便于显示，使用了换行符）是

```
eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2IiwiY3R5IjoiSldU
In0.
g_hEwksO1Ax8Qn7HoN-BVeBoa8FXe0kpyk_XdcSmxvcM5_P296JXXtoHISr_DD_M
qewaQSH4dZOQHoUgKLeFly-9RI11TG-_Ge1bZFazBPwKC5lJ6OLANLMd0QSL4fYE
b9ERe-epKYE3xb2jfY1AltHqBO-PM6j23Guj2yDKnFv6WO72tteVzm_2n17SBFvh
DuR9a2nHTE67pe0XGBUS_TK7ecA-iVq5COeVdJR4U4VZGGlxRGPLRHvolVLEHx6D
YyLpw30Ay9R6d68YCLi9FYTq3hIXPK_-dmPlOUlKvPr1GgJzRoeC9G5qCvdcHWsq

JGTO_z3Wfo5zsqwkxruxwA。

UmVkbW9uZCBXQSA5ODA1Mg。
VwHERHPvCNcHHpTjkoigx3_ExK0Qc71RMEParpatm0X_qpg-w8kozSjfNIPPXiTB
BLXR65CIPkFqz4l1Ae9w_uowKiwyi9acgVztAi-pSL8GQSXnaamh9kX1mdh3M_TT
-FZGQFQsFhu0Z72gJKGdfGE-OE7hS1zuBD5oEUfk0Dmb0VzWEzpxxiSSBbBAzP10
l56pPfAtrjEYw-7ygeMkwBl6Z_mLS6w6xUgKlvW6ULmkV-uLC4FUiyKECK4e3WZY
Kw1bpgIqGYsw2v_grHjszJZ-_I5uM-9RA8ycX9KqPRp9gc6pXmoU_-27ATs9XCvr
ZXUtK2902AUzqpeEUJYjWWxSNsS-r1TJ1I-FMJ4XyAiGrfmo9hQPcNBYxPz3GQb2
8Y5CLSQfNgKSGt0A4isp1hBUXBHAndgtcslt7ZoQJaKe_nNJgNliWtWpJ_ebuOpE
l8jdhehdccnRMIwAmU1n7SPkmhIl1HlSOpvcvDfhUN5wuqU955vOBfkBOh5A11U
zBuo2WlgZ6hYi9-e3w29bR0C2-pp3jbqxEDw3iWaf2dc5b-LnR0FEYXvI_tYk5rd
_J9N0mg0tQ6RbpxNEMNoA9QWk5lgdPvbh9BaO195abQ.AVO9iT5AV4CzvDJCdhSFl
Q
```

附录 B. JWT 与 SAML 断言的关系安全断言标记语言（SAML ）2.0

[OASIS.saml-core-2.0-os] 为创建安全令牌提供了一个标准，与 JWT 相比，它具有更强的表达能力和更多的安全选项 不过，这种灵活性和表达能力的代价是规模和复杂性。 SAML 对 XML [W3C.CR-xml11-20060816]和 XML 数字签名 (DSIG) [RFC3275] 的使用增加了 SAML 断言的大小；对 XML 的使用，特别是 XML 标准化 [W3C.REC-xml-c14n-20010315] 增加了 SAML 断言的复杂性。

JWT 旨在提供一种简单的安全令牌格式，这种格式足够小，可以放入 HTTP 标头和 URI 中的查询参数。为此，它支持比 SAML更简单的令牌模型，并使用 JSON [RFC7159] 对象编码语法。 它还支持使用比 XML DSIG 更小（更不灵活）的格式，使用消息验证码 (MAC) 和数字签名确保令牌安全。

因此，虽然 JWT 可以完成 SAML 断言的某些功能，但 JWT 并不打算完全取代 SAML 断言，而是作为一种标记格式，在考虑到易于实施或紧凑性的情况下使用。

JWT 通常以同样的方式使用，发表声明的实体由 "iss"（签发者）声明表示，主体由 "sub"（主体）声明表示。

不过，由于这些要求是可选的，因此也允许使用 JWT 格式的其他用途。

附录 C. JWT 与简单网络令牌（SWT）的关系

JWT 和 SWT [SWT]的核心都是应用程序之间传输一组权利要求。 对于 SWT，权利要求名称和权利要求值都是字符串 对于 JWT，虽然权利要求名称是字符串，但权利要求值可以是任何 JSON 类型。 两种令牌类型对其内容提供加密保护：SWT 采用 HMAC SHA-256 算法，而 JWT 则可选择多种算法，包括签名、MAC 和加密算法。

致谢

作者承认，JWT 的设计有意受到 SWT [SWT] 的设计和简洁性以及 Dick Hardt 在 OpenID 社区中讨论的 JSON 令牌想法的影响。

此前，Magic Signatures [MagicSignatures]、JSON Simple Sign [JSS] 和 Canvas Applications [CanvasApp] 等公司都探索过 JSON 内容的签名解决方案，所有这些公司都对本文档产生了影响。

本规范是 OAuth 工作组的心血结晶，该工作组由数十名积极而敬业的参与者 ，特别是以下个人提供的想法、反馈和措辞对本规范产生了影响：

Dirk Balfanz、Richard Barnes、Brian Campbell、Alissa Cooper、Breno de Medeiros、Stephen Farrell、Yaron Y. 戈兰德、迪克-哈特、乔-希尔德布兰德、杰夫-霍奇斯、埃德蒙-杰伊、沃伦-库马里、本-劳里、巴里-雷巴、特德-莱蒙、詹姆斯-曼格、普尔提克-米什拉、凯瑟琳-莫里亚利蒂、托尼-纳达林、阿克塞尔-能克、约翰-潘泽、伊曼纽尔-拉维亚特、戴维-雷科东、埃里克-雷斯科拉、吉姆-沙德、保罗-塔扬、汉内斯-茨乔费尼格、肖恩-特纳和汤姆-于。

Hannes Tschofenig 和 Derek Atkins 担任 OAuth 工作组主席，Sean Turner、Stephen Farrell 和 Kathleen Moriarty 担任本规范创建期间的安全区域总监。

作者地址

迈克尔-B-琼斯 微软

EMail: mbj@microsoft.com
URI: http://self-issued.info/

约翰-布拉德利-平的
身份

EMail: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/

Nat Sakimura
野村综合研究所

EMail: n-sakimura@nri.co.jp URI
: http://nat.sakimura.org/