3 OpenID Protocol and Messages

OpenID è un sistema che consente a un utente di utilizzare una Uniform Resource Identifier o URI (come l'URL del sito web) per scopi di autenticazione. Questo URI viene utilizzato come username o identità per un persona e viene anche chiamato "**identifier**".

3.1 Terminologia OpenID

End User

L'utente finale è l'utente reale che utilizza il sistema OpenID per effettuare il login su siti web diversi utilizzando la propria credenziale memorizzate presso il provider di identità.

Consumer or Relying Party (RP)

I consumatori "c*onsumer"* sono i siti web dove si effettuerà il login utilizzando OpenID. Si chiamano "consumatori" perché che consumano, usano, le credenziali OpenID fornite dal provider di identità. Tutti i siti web che supportano il login con OpenID sono consumatori detti anche Relying Party or RP (Parte Fidata).

Identifier

Identificatore "*Identifier"*è l'URL che identifica l'identità digitale dell'End User.

Identity Provider or IdP (OP)

Provider di identità "Identity Provider or IdP" è l'host dove vengono memorizzate le credenziali dell'utente. L'URI OpenID indica il provider di identità. Durante il processo di autenticazione, il consumer convaliderà un ID scambiando alcuni messaggi con il provider di identità. A volte viene anche chiamato come OpenID Server, OpenID Provider o semplicemente OP.

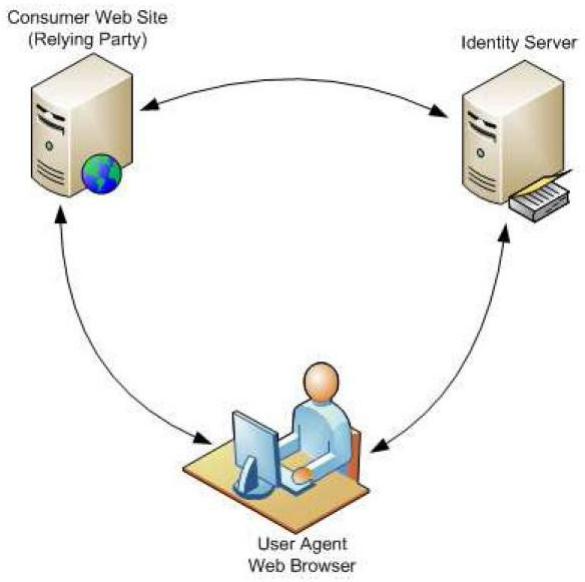
User Agent

Il "User Agent" non è altro che il browser dell'utente, con il quale comunica direttamente.

3.2 Comunicazione tra componenti in un sistema OpenID

Esistono tre principali componenti in qualsiasi sistema OpenID: "**consumer**", "**identity provider**", "**user agent**". Questi componenti interagiscono tra loro durante il processo di autenticazione:

- Il consumer, interagisce con l'identity provider e con lo user agent. L'utente finale tenterà di accedere al sito web del consumer tramite OpenID. Durante il processo di autenticazione, il consumer invierà alcuni messaggi al identity provider direttamente, nonché tramite lo user agent con l'aiuto di HTTP redirect messages.
- L'identity provider è il server OpenID che contiene le credenziali dell'utente finale. L'identity provider convaliderà il proprietario del URL per il consumer utilizzando due meccanismi fondamentali che vedremo in seguito.
- L'utente finale interagisce con il consumer e l'identity provider mediante lo user agent.



Durante il processo di autenticazione, il browser agisce come "**middle man**" tra il provider di identità e il sito web del consumatore per alcuni messaggi. In genere, un consumer interagisce con il browser web, cosi come con il provider di identità durante il processo di autenticazione. Tuttavia in alcuni casi, il consumer può utilizzare delle chiavi memorizzate nella cache per autenticare un utente senza alcuna comunicazione diretta con il provider di identità.

3.3 Direct and Indirect Communication

Esistono due metodi di comunicazione base tra le entità diverse in un sistema OpenID:

- comunicazione diretta e
- comunicazione indiretta.

Con il meccanismo della comunicazione diretta, le due entità dialogano direttamente tra loro tramite il protocollo HTTP. Il metodo POST HTTP viene utilizzato per la comunicazione diretta.

Con la comunicazioni indiretta, le due entità dialogano tra loro tramite una terza entità. Questa terza entità è in genere il browser web. La comunicazione indiretta può avvenire tramite HTTP Redirect o tramite HTML Form redirection.

3.4 OpenID Modes of Operation

OpenID ha due modalità principali di operare:

- la modalità Dumb
- la modalità Smart.

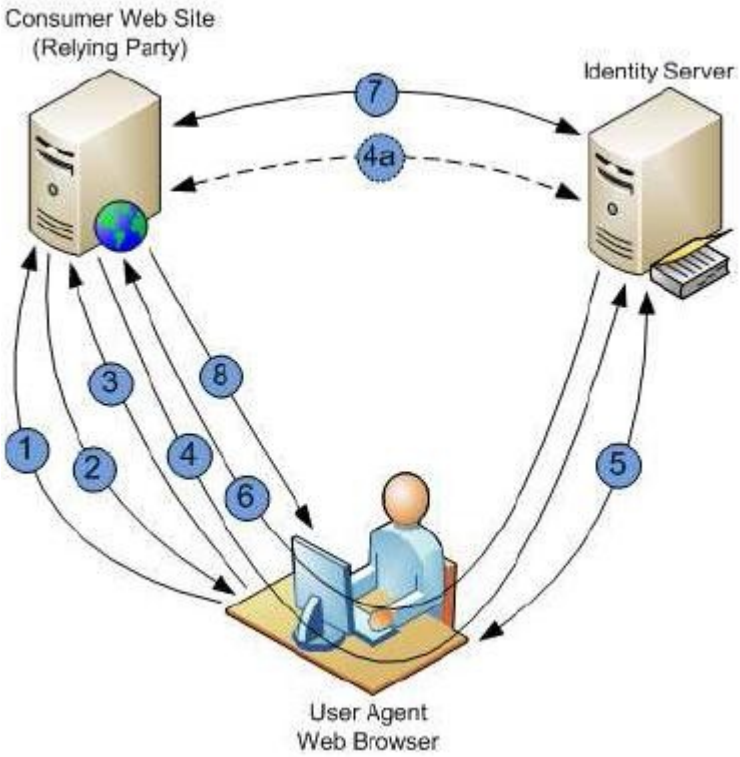
Queste modalità si basano nell'intelligenza del consumer. In modalità **dumb**, come indica il nome, il consumatore non è cosi smart e deve eseguire alcuni passaggi aggiuntivi ogni volta che un utente accede. In modalità **smart**, il consumer mantiene informazioni sullo stato e mette nella cache le chiavi condivise per un utilizzo futuro. In questa sezione, si vedranno informazioni più dettagliate sul funzionamento di queste due modalità.

3.4.1 Dumb Mode Communications Flow

In modalità Dumb, i consumer non mantengono lo stato della connessione, quindi qualsiasi informazione che è stata utilizzata in un precedente login, non potrà essere più utilizzata. Ogni volta che un End User accede a un Dumb Consumer web site, viene ripetuto lo stesso processo. Prendiamo come esempio

- Identity Provider: myopenid
- Relying Party: livejournal

In seguto vediamo step – by – step il processo di autenticazione e login quando viene effettuata la comunicazione fra: il consumer, lo user agent e l'identity provider. La figura sotto da una presentazione grafica di tali passaggi:



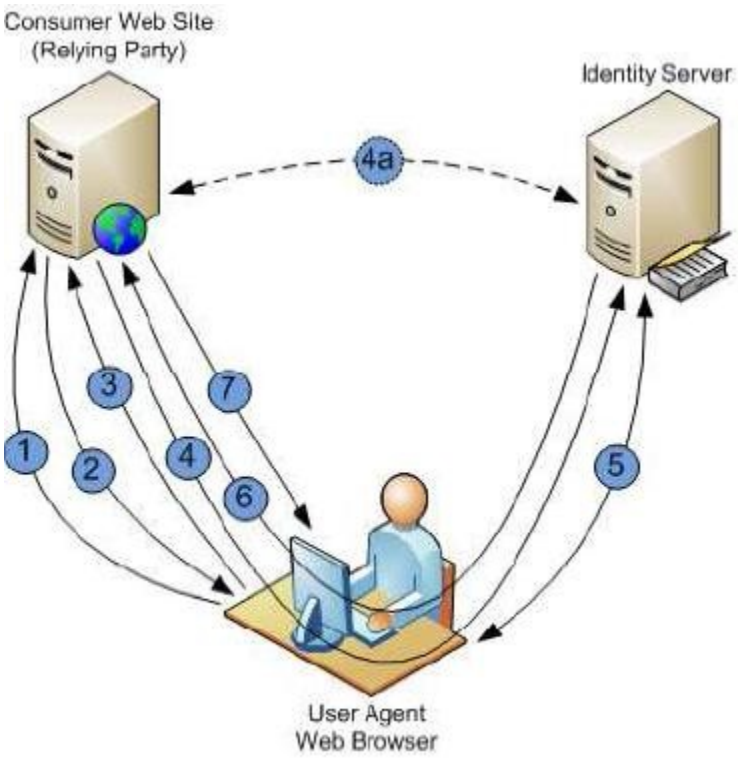
1. accediamo alla pagina web del consumer: http://www.livejournal.com/
2. inseriamo l'**url** dell'identità ed effettuaiamo il login : nome_utente.myopenid.com

   Da notare che non è necessario aggiungere "http:// " all'inizio dell'URL. Le specifiche dell'OpenID richiedono che tutti i consumer dovrebbero essere abbastanza intelligenti a comprendere un URL in formati diversi.
3. Il sito web del consumer (livejournal.com) pulirà l'identifier URL e recupererà le informazioni da tale URL. In OpenID 2.0, un altro protocollo XML, conosciuto come **Yadis**, può essere utilizzato in questa fase di rilevamento del servizio.
4. Dopo il recupero della pagina, il consumer (livejournal.com) analizzerà e determinerà la posizione del provider di identità (OpenID Server). Questa informazione è incorporata all'interno della pagina web HTML. Questo processo di analisi viene anche chiamato "discovery". Dopo l'analisi, il consumer (livejournal.com) reindirizzerà il browser al Identity Provider per ottenere le informazioni necessarie per l'autenticazione. Questo avviene utilizzando il metodo HTTP GET.

   4.a Facoltativamente, il consumatore può stabilire una connessione a questo punto con il provider di identità e condividere una chiave segreta per un'ulteriore comunicazione. Questo è illustrato con una linea tratteggiata nella figura e contrassegnato come passo 4 a.
5. Se l'End User non è già loggato presso l'Identity Provider, l'Identity Provider chiederà all'End User di effettuare il login. Questa parte è esterna al protocollo OpenId ed è lascia all'Identity Provider la facoltà di decidere come autenticare l'End User. Se l'utente è già loggato nell'Identiy Provider questa parte viene saltata.
6. L'identity provider (myopenid.com) ritornerà l'informazione necessaria con la sua firma al Consumer via browser redirect. Questa informazione contenente il risultato dell'autenticazione sia che è andato a buon fine (success) oppure no (failure). Il metodo HTTP GET è utilizzato in questo step. Da notare che c'è una comunicazione indiretta fra l'Identity Provider e il Consumer.
7. Durante l'autenticazione con successo, il consumer (livejournal.com) dovrà stabilire una connessione diretta con l'Identity Provider (myopenid.com), preferibilmente in tramite una sessione SSL sicura. Il Consumer potrà richiedere le informazioni di autenticazione direttamente dall'Identity Provider e confrontarle con le informazioni di asserzione ricevuta tramite User Agent (browser web). Questo è un doppio controllo per la validità dell'affermazione nel caso in qui uno User Agent (o un utente malintenzionato) sta cercando di imbrogliare.
8. Se tale confronto va a buon fine allora l'End User verrà loggato nel sistema, altrimenti il login fallirà.

3.4.2 Smart Mode

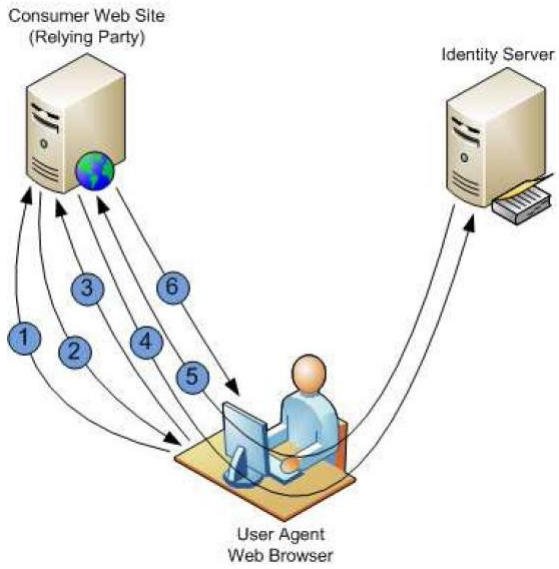
La modalità "Smart" per l'autenticazione è simile alla modalità "Dumb" con l'unica eccezione allo step 7. In questo caso il consumer ha già la chiave segreta (dal passo 4a) e può decriptare e verificare l'affermazione dal step 6 e determinare se l'Identity Provider ha già autenticato l'End User. Da notare che lo step 4a viene effettuato una volta ogni tanto, quando l'End User svuota la cache oppure quando effettua la richiesta di autenticazione la prima volta.



1. L'end user visita la pagina web del counsumer
2. La pagina web del consumer si presenta con una form nella quale l'utente inserisce il suo identity URL e clicka l'invio.
3. Il consumer pulisce l'identifier url e prende la pagina puntata dall'url.
4. Dopo aver preso la pagina dall'url dell'utente, la parsa, e determina la posizione dell'identity provider. Dopo che ha effettuato il parsing della pagina, il consumer, effettuerà un redirect del browser verso l'identity provider per ottenere l'affermazione della validità dell'indetifier che l'utente ha inserito. In questo step, il consumer può inviare una messaggio di “association request” all'identity provider e scambiare con lui una chiave condivisa (step 4.a).
5. Se l'utente non è loggato presso l'identity provider, allora l'identity provider può richiedere all'end user di effettuare l'autenticazione.
6. L'identity provider restituirà le informazioni di affermazione con la firma per il consumatori tramite browser redirect. Tale affermazione rappresenterà sia un successo di autenticazione o il suo fallimento.
7. Dopo un'affermazione di successo, il consumer verifica l'affermazione utilizzando la chiave condivisa nella cache. Se c'è una corrispondenza nel passaggio precedente, l'utente finale verrà login al sito web. Altrimenti il login fallirà.

In molti casi quando si hanno già stabilite le credenziali, il provider di identità non richiederà nuovamente per il login. Ci sono molte tecniche che possano essere utilizzate dal provider di identità per raggiungere questo obiettivo, tra cui active browser session e i cookie. In tale scenario, e supponendo che il consumer ha già memorizzato nella cache la chiave condivisa con l'dentity provider la comunicazione diventerà molto semplice e l'utente finale potrà effettuare il login al sito web del consumer senza alcuna sessione interattiva con il provider di identità.

Ciò è mostrato in figura:



Come si vede in figura le comunicazioni con l'identity provider sono trasparenti all'utente, e tutte le operazioni succedono dietro le scena via browser redirect.

In questo caso, i passaggi saranno i seguenti. Si noti che l'utente finale potrà immediatamente accedere al sito web dopo aver immesso l'URL di identificatore:

1. Visita del sito web del consumer
2. Il sito web presenta la pagina dove si inserisce l'identifier url.
3. Il consumer pulisce l'identifier url e prende la pagina puntata dall'url.
4. Dopo aver preso la pagina dall'url dell'utente, la parsa, e determina la posizione dell'identity provider.
5. L'identity provider restituirà le informazioni di affermazione con la firma per il consumatori tramite browser redirect. Tale affermazione rappresenterà sia un successo di autenticazione o il suo fallimento.
6. Dopo un'affermazione di successo, il consumer verifica l'affermazione utilizzando la chiave condivisa nella cache. Se c'è una corrispondenza nel passaggio precedente, l'utente finale verrà login al sito web. Altrimenti il login fallirà.

3.5 OpenID Identity URL Page

Diamo un occhiata all'informazione presente nella pagina dell'identità OpenId. Questa pagina può essere ospitata preso un qualsiasi server web (non necessariamente al server web dell'identity provider). Basta creare una pagina HTML con le specifiche informazioni e metterla in un sito web. L'url presso questo documento sarà il nostro identificatore. Il documento HTML avrà l'informazione riguardante il server OpenId (il provider) nella sessione HEAD della pagina.

```
<link rel="openid.server" href=https://myopenid.com/ >
```

Si noti che nella riga sopra,"**openid.server**" è una parola chiave e dovrebbe apparire esattamente come indicato sopra. La parte "href" può variare a seconda dell'identity provider che si utilizza. Si noti inoltre che l'URL per il provider di identità possa iniziare con HTTP o HTTPS a seconda della configurazione dell'identity provider.

Vediamo un esempio di una pagina HTML puntata dall'url dell'identità:

```
<html>
	<head>
		<link rel="openid.server" href="http://idp.conformix.com/index.php/serve">
		<link rel="openid.delegate" href="http://idp.conformix.com/?user=test">
	</head>
<body>
	<h3>OpenID Identity Page</h3>
</body>
</html>
```

In questo caso le seguenti linee indicano il percorso del server OpenID che il consumer contatterà per l'autenticazione. Usando questo meccanismo si può separare tale URL dal URL dell'identificatore.

```
<link rel="openid.server" href="http://idp.conformix.com/index.php/serve">
```

La riga seguente mostra l'URL dell'OpenID. Da notare che la parola chiave "***delegate***" viene usata quando l'URL dell'identificatore è ospitato in un'altra macchina diversa dal server.

```
<link rel="openid.delegate" href="http://idp.conformix.com/?user=openidbook">
```

Se il consumer e l'OpenID Server supportano le specificazioni OpenID 2.0, può essere usato anche un documento XRD al posto del documento HTML. Un tipico documento XRD si presenta come segue:

```
<?xml version="1.0" encoding="UTF-8"?>
	<xrds:XRDS
		xmlns:xrds="xri://$xrds"
		xmlns:openid="http://openid.net/xmlns/1.0"
		xmlns="xri://$xrd*($v*2.0)">
<XRD>
	<Service>
		<Type>http://openid.net/signon/1.1</Type>
		<Type>http://openid.net/sreg/1.0</Type>
		<URI>http://idp.conformix.com/index.php/serve</URI>
		<openid:Delegate>
			http://idp.conformix.com/?user=openidbook</openid:Delegate>
	</Service>
</XRD>
</xrds:XRDS>
```

3.6 OpenID Messages

I componenti OpenID scambiano diversi messaggi durante il processo di autenticazione. I formati di questi messaggi sono ben definiti e i Consumers e Identity Providers devono rispettare questi formati per una comunicazione corretta.

Ogni messaggio ha una combinazione di richiesta e risposta e la richiesta e risposta possono avere diversi set di parametri. Vedremo i parametri di richiesta e risposta separatamente.

Come abbiamo visto in precedenza, il Consumer e l'Identity Provider possono usare la comunicazione diretta o indiretta. Per la comunicazione diretta viene usato il metodo "HTTP POST" mentre per quella indiretta "HTTP GET".

Ci sono quattro tipi principali di messaggi usati in un sistema OpenID:

1. *associate* message
2. *checkid_immediate* message
3. *checkid_setup* message
4. *check_authentication* message

3.6.1 The associate Request Message

L'"associate message" è inviato dal Consumer (Relying Party o RP) all'Identity Provider (o OP). Lo scopo principale di questo messaggio è quello di stabilire un "segreto condiviso" tra queste due entità. Il sito web del Consumer può inviare questo messaggio al Identity Provider in qualsiasi momento è richiesto (quando si svuota la cache oppure nella comunicazione smart).

Siccome questa è una comunicazione diretta tra il Consumer e l'Identity Provider, il metodo "HTTP POST" sarà usato per i messaggi di tipo "associate message". Mentre si avvia la richiesta di "associate", il Consumer invierà una serie di parametri insieme alla richiesta. Vediamo una lista di tali parametri che possono essere inviati:

- **openid.ns** : Opzionale, serve per specificare la versione del protocollo OpenID (1.1 – 2.0)
- **openid.mode** : indica il tipo di messaggio e serve per distinguere quale messaggio è stato inviato o ricevuto. Il valore per questo parametro è "associate" per un messaggio di "associate request".
- **openid.assoc_type** : questo parametro viene usato per indicare il tipo di algoritmo utilizzato per la firma del messaggio. OpenID utilizza i seguenti algoritmi per la firma:
  - "HMAC-SHA1"
  - "HMAC-SHA256"
- **openid.session_type** : è utilizzato per indicare il tipo di crittografia utilizzato nel messaggio per crittografare la chiave MAC (Message Authentication Code). Un MAC è breve codice che viene calcolata utilizzando una chiave segreta e il messaggio in input. Un algoritmo MAC prende la chiave privata e il messaggio come input e fornisce il codice MAC come output. Hash Message Authentication Code o HMAC è un tipo di MAC.

  Nei messaggi di OpenID, diversi tipi di crittografia vengono utilizzati per crittografare MAC riportati di seguito:

  - "DH-SHA1" per Diffie-Hellman SHA1

- “DH-SHA256” per Diffie-Hellman SHA256
- Il valore “no-encryption” per inviare il MAC in chiaro, anche se è fortemente raccomandato crittografarlo. Se si usa una sessione di connessione SSL tra il Consumer e l’Identity Provider si può optare a non crittografare il MAC poiche il trasport layer si occupa della crittografia.

- **openid.dh_modulus** - **openid.dh_gen** - **openid.dh_consumer_public**: questi tre parametri vengono usati nel caso in cui viene utilizzato DH-SHA1 o DH-SHA256 per openid.session_type.

Vediamo una tipica richiesta di “**associate Request Message**”

```
openid.mode=associate&openid.assoc_type=HMAC-SHA1&openid.session_type=DHSHA1&
openid.dh_consumer_public=KC6IpA00A6SlCikafFSlrTGql9H8+de6GFi5YLKz4p
yDxUMS5Z8pMOm/Ptr1gFmCcgAXjFbuxS73ZutDTFJYpADoIntFVrah9eaezMcw6SDR24cnFjN
c14xq0zGt3QcRLXaNTRVKfMW8evDAmLCrvEhU5c7B3eqmk+bMMrbQpcE=&openid.dh_modul
us=ANz5OguIOXLsDhmYmsWizjEOHTdxfo2Vcbt2I3MYZuYe91ouJ4mLBX+YkcLiemOcPym2CB
RYHNOyyjmG0mg3BVd9RcLn5S3IHHoXGHblzqdLFEi/368Ygo79JRnxTkXjgmY0rxlJ5bU1zIK
aSDuKdiI+XUkKJX8Fvf8W8vsixYOr&openid.dh_gen=Ag==
```

3.6.2 The associate Response Message

Il messaggio “associate response” viene inviato dall’Identity Provider al Consumer (o RP). Il messaggio può mostrare un'associazione con successo o fallimento. Nel caso in cui l’associazione è andata a buon fine, il messaggio conterrà un “handle” e la vita di quel “handle” in secondi. Nel caso in cui l’associazione è andata male verrà restituito un messaggio di errore. Lista dei parametri:

- **openid.ns**
- **openid.assoc_handle**: è una stringa ASCII con lunghezza massima di 255 caratteri. Tale “association handle” può essere utilizzata nei messaggi successivi. In genere questo “handle” dura per un po' di tempo. Viene utilizzata per determinare quale chiave dovrebbe essere riutilizzata per la crittografia/decrittografia.
- **openid.session_type**: è uguale al caso di request, se l'associazione fallisce per qualsiasi motivo, il valore di questo parametro sarà “unsuccessful response”. Possono esserci diversi motivi per un'associazione non riuscita. Ad esempio, se il Consumer chiede di utilizzare SHA256 e l’Identity Provider può supportare solo SHA1, l'associazione avrà esito negativo.
- **openid.assoc_type:** è uguale al caso di request, “unsuccessful response” nel caso di un fallimento
- **openid.expires:in**: è il tempo in secondi dopo di ché l’associazione scade e il Consumer (RP) deve chiedere una nuova associazione (ttl della handle).
- **openid.mac_key**: parametro che viene usato solo nel caso in cui il valore per “openid.session_type” era “no-encryption”. Il valore per questo parametro è una chiave MAC codificata e in base 64.
- **openid.server_public**: è la chiave pubblica dell’Identity Provider. Questo parametro viene utilizzato se è stato utilizzato l’algoritmo di Diffie-Hellman nella richiesta.
- **openid.enc_mac_key**: è la chiave MAC codificata. Questo parametro viene utilizzato se è stato utilizzato l’algoritmo di Diffie-Hellman nella richiesta.

In caso di insuccesso, vengono restituiti i parametri “error” e “error_code”. La risposta può avere anche altri parametri facoltativi, ma i parametri di errore ed error_code sono importanti.

Esempio:

```
assoc_handle:{HMAC-SHA1}{4607344a}{oDFF0g==}
assoc_type:HMAC-SHA1
dh_server_public:AIPkx6xJ3b1Wnr1olWL7suoZnABDc+lJRR9DeNIBolGXQX3W2e+4udY2
p+dUcF5jKE6uoZuXLVPbimHbndBOYhUDUfkKaAjQtVvONerAjd5RHyt2i2AoYrkjD26traC4j
zg7NukZlmrRjfPRg4q3gwW+EZEXvz+ba9JnQfsXx+iH
enc_mac_key:UtQHBswQimAZAp4s/9sfSQSpuq0=
expires_in:1209600
session_type:DH-SHA1
```

3.6.3 The checkid_setup and checkid_immediate Request Messages

I messaggi **checkid_immediate** e **checkid_setup** vengono utilizzati per ottenere informazioni di affermazione dal server OpenID. Questi messaggi vengono avviati dal Consumer. Per questi messaggi viene utilizzata la comunicazione indiretta, che significa che il consumer utilizzerà il metodo GET HTTP (invece di HTTP POST) per l'invio e ricezione di tali messaggi. Questo significa che questi messaggi passeranno per lo User Agent (il browser web). I seguenti parametri vengono usati con i messaggi “checkid_setup request”.

- **openid.ns**
- **openid.mode:** che ha come valore “checkid_setup”
- **openid.claimed_id:** parametro opzionale visualizza l’identificatore che l’utente dichiara di possedere, ma che non è ancora verificato.
- **openid.identity:** opzionale
- **openid.assoc_handle**: opzionale e se presente indica che è già stata stabilita una associazione fra il Consumer e l’Identity Provider.
- **openid.return_to**: parametro usato per informare il server OpenID dell’URL dove deve reindirizzare il browser dopo aver processato la richiesta. Il server OpenID userà questo URL per inviare la risposta al Consumer.
- **openid.realm**: opzionale è un’URL che il server OpenID usa per identificare il Consumer. Questo parametro può contenere caratteri speciali come ad esempio “*”.

Da notare inoltre che il messaggio di richiesta raggiunge il server OpenID in due passaggi. Nel primo passaggio, il metodo HTTP redirect viene utilizzato dal Consumer per il browser web. Quindi nel passaggio successivo, il browser invia la richiesta GET HTTP al server di OpenID.

Esempio di un messaggio checkid_setup request:

```
GET /index.php/serve?openid.assoc_handle={HMACSHA1}{
46071e25}{Tt8MwQ==}&openid.identity=http://idp.conformix.com/?user=
openidbook&openid.mode=checkid_setup&openid.return_to=http://consumer.con
formix.com:80/finish_auth.php?nonce=nC5sKquX&openid.sreg.optional=email&o
penid.trust_root=http://consumer.conformix.com:80/ HTTP/1.1
```

3.6.4 The checkid_setup and checkid_immediate Response Messages

Una volta ricevuto il messaggio “checked_setup” il server OpenID effettua alcune operazioni ed invia la risposta indietro al Consumer via browser. Opzionalmente il server OpenID può chiedere all’End User di autenticarsi preso questo server per essere sicuro che solo il proprietario dell’Identifier Url può autorizzare l’accesso all’Identifier. Questo step opzionale può presentarsi con una pagina di login, ma come detto in precedenza l’implementazione di questa parte riguarda il server OpenID e non è inclusa nelle specifiche di OpenID.Nel messaggio della risposta, il server OpenID invierà alcuni parametri indietro al Consumer. Alcuni di questi parametri sono opzionali e alcuni necessari:

- **openid.ns**
- **openid.mode:** che avrà come valore “id_res”. Nel caso in cui l’associazione fallisce il valore per questo parametro sarà:
  - “setup_needed” nel caso in cui la richiesta fosse “checkid_immediate”
  - “cancel” nel caso in cui la richiesta fosse “checkid_setup”
- **openid.op_endpoint:** visualizza l’URL del server OpenID
- **openid.claimed_id**
- **openid.identity**
- openid.assoc_handle: rappresenta l'handle che è stato utilizzato per firmare questo messaggio. Il consumer utilizzerà questo handle per finalità di verifica. Questo handle può coincidere con quello che ha inviato il Consumer con il messaggio di richiesta (nel caso esiste già un'associazione). Può essere diverso da quello originale che il Consumer ha inviato con la richiesta se il server OpenID non riconosce l'handle originale. In tal caso, il server deve mantenere un record del nuovo handle di modo che il Consumer possa utilizzarlo a scopo di verifica (utilizzando il messaggio check_authentication discusso tra breve).
- **openid.return_to**: è identico allo stesso parametro inviato dal Consumer con il messaggio di request.
- **openid.response_nonce**: parametro utilizzato per evitare relay attacks, è univoco per ogni messaggio. La lunghezza massima di una **nonce** è di 255 caratteri ed è costituito dal server timestamp e altri caratteri ASCII per renderlo univoco. Il Consumer può respingere come associazione se il timestamp è troppo lontano dal tempo corrente.
- **openid.invalidate_handle**: è utilizzato per indicare se l'handle collegato alla richiesta era valida o no. Se l'handle era valido, questo parametro è facoltativo. In caso contrario, l'handle non valido debba essere collegato con questo parametro. Questo aiuterà il Consumer a rimuovere tale handle dai suoi record.
- **openid.signed**: contiene un elenco di parametri che sono firmati. L'elenco è separato da virgola.

Come con il messaggio di richiesta, il messaggio di risposta raggiunge il sito web dei consumatori in due fasi. Nel primo passaggio, il Server OpenID utilizza il metodo redirect HTTP dal Server OpenID per il browser web. Quindi nel passaggio successivo, il browser invia la richiesta HTTP GET il sito web dei consumatori. Ciò è dimostrato in figura :

Esempio di un messaggio checkid_setup response:

```
GET /finish_auth.php?nonce=nC5sKquX&openid.assoc_handle={HMACSHA1}{
46071e25}{Tt8MwQ==}&openid.identity=http://idp.conformix.com/?user=
openidbook&openid.mode=id_res&openid.return_to=http://consumer.conformix.
com:80/finish_auth.php?nonce=nC5sKquX&openid.sig=nXWc+07GLaSf+RghmGubGPPg
lZc=&openid.signed=mode,identity,return_to,sreg.email&openid.sreg.email=r
r@conformix.com HTTP/1.1
```

3.6.5 The check_authentication Request Message

I messaggi di richiesta e risposta di check_authentication sono uno strumento necessario per verificare l'affermazione ricevuta dall'User Agent (browser web). Questo è necessario per garantire che un utente malintenzionato non invia messaggi affermandosi al posto del server OpenID. Si noti quanto segue per i messaggi di check_authentication:

1. Questo messaggio non verrà inviato se esiste già un'associazione tra il sito web dei consumatori e il server OpenID. L'associazione viene inizialmente stabilita mediante “associate message”.
2. Se viene utilizzata un'associazione esistente, il consumer invierà il parametro “openid.assoc_handle ” nella richiesta e il Server OpenID invierà lo stesso handle nella risposta. In tal caso, il sito web dei consumatori saprebbe che il server OpenID ha accettato di utilizzare l’handle esistente.
3. Se il Server OpenID non accetta di utilizzare l’handle dell’associazione fornito dal sito web dei consumatori, la risposta includerà il parametro “openid.invalidate_handle ” il messaggio di risposta e una diverso “openid.assoc_handle ”. Inoltre il consumer utilizzerà il messaggio “check_authentication” per convalidare l'affermazione.
4. Quando si utilizza la modalità dumb, questo messaggio di risposta verrà utilizzato sempre perché i consumer è stateless e nessun record di qualsiasi precedente handle di associazione esiste.

Si noti che questo messaggio è una comunicazione diretta tra il consumer e il server di OpenID e il metodo POST HTTP viene utilizzato per questa comunicazione. La richiesta ha il parametro “openid.mode ” con un valore “check_authentication” e tutti gli altri parametri che facevano parte del messaggio “associate”. Un tipico messaggio avrà l'aspetto seguente:

Esempio:

```
openid.assoc_handle={HMACSHA1}{
460730e1}{zr1gKg==}&openid.identity=http://idp.conformix.com/?user=
openidbook&openid.invalidate_handle={HMACSHA1}{
46071e25}{Tt8MwQ==}&openid.mode=check_authentication&openid.return_
to=http://consumer.conformix.com:80/finish_auth.php?nonce=mAotRbGM&openid
.sig=4hwwyWbPtSAmP2dYxEC+dq605Os=&openid.signed=mode,identity,return_to,s
reg.email&openid.sreg.email=rr@conformix.com
```

3.6.6 The check_authentication Response Message

In risposta al messaggio “check_authentication”, il Server OpenID invierà un messaggio breve con i seguenti parametri:

- **openid.ns**
- **is_valid:** parametro che ha un valore “true ” o “false ”
- **invalidate_handle:** parametro facoltativo e in caso che il parametro “is_valid” è true, il consumer rimuoverà l'handle dal elenco delle handle salvate.

Un tipico messaggio è il seguente:

```
invalidate_handle:{HMAC-SHA1}{46071e25}{Tt8MwQ==}
is_valid:true
```

Da notare che questo messaggio è molto breve e la risposta include solo l'esito positivo o negativo del controllo di autenticazione.

3.7 How OpenID Works: Some Scenarios

Come sapiamo ormai, OpenID viene utilizzato per autenticare un utente su diversi siti web utilizzando le credenziali archiviate nel server di identità scelto. Esistono diversi scenari di come si svolgerà l'autenticazione OpenID.

3.7.1 Scenario One: First Time Login to a Web Site Using OpenID in Dumb Mode

Quando ci loggiamo su un sito web che supporta OpenID per la prima volta succedono un paio di cose. In seguito vediamo una lista di steps (molto in generale, senza entrare nei dettagli):

1. Inseriamo il nostro OpenID URL nella pagina del consumer cliccando il pulsante Login.
2. Il sito web del consumer identifica la locazione del OpenID server, e può usare Yadis per scoprire i servizi che l'URL offre. Il consumer reindirizza il browser (lo user agent) al server OpenID per ottenere le credenziali.
3. Poiché questa è la prima volta che l'utente va presso questo sito web, il server OpenID non sa se questo sito web è attendibile o no. Così il server OpenID visualizzera una schermata di login (login al server OpenID).
4. L'utente indica questo sito web (del consumer) come sito fidato al server OpenID.
5. Il server OpenID reindirizza il browser nella pagina del consumer.
6. Il consumer controlla l'autenticazione direttamente con il server OpenID e l'autenticazione è finita.

3.7.2 Scenario Two: Login to a Trusted Web Site Using OpenID in Smart Mode

L'utente può visitare regolarmente dei siti web. Se questi siti supportano OpenID, si possono segnare come siti “fidati” preso l'Identity Provider. Ecco i step in questo caso:

1. Inseriamo il nostro OpenID URL nella pagina del consumer cliccando il pulsante Login.
2. Il consumer stabilisce un associazione con l'OpenID server, se una tale associazione non esiste.
3. Il sito web localizza l'OpenID server e reindirizza il browser presso il server per ottenere le credenziali. In questo passo può utilizzare anche Yadis.
4. Il server OpenID sa che questo è un consumer fidato e sa quali parametri passargli. Se l'utente è già autenticato presso il server OpenID, il server invierà al consumer le credenziali richieste. L'utente verrà loggato presso il consumer senza ulteriori passaggi.

3.8 Problems Solved by OpenID

OpenID risolve una serie di questioni riguardante l'Identity Management. Alcuni di questi sono i seguenti:

- gli utenti hanno il controllo su quali dati devono essere condivisi con il sito web del consumer.
- OpenID consente, l'utilizzo delle credenziali attraverso tutti i siti web abilitati per OpenID. Così non è necessario creare singolarmente username e password per ogni sito web.
- Ferma i replay attacks utilizzando la variabile nonce una sola volta. Un consumer può ignorare un'affermazione positiva, esaminando il timestamp nella variabile nonce. Se il timestamp è troppo lontano dal tempo corrente, il consumer può respingerla.