ETSI TR 133 924 V9.0.0 (2010-02)

Technical Report

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Identity management and 3GPP security interworking;
Identity management and Generic Authentication Architecture (GAA) interworking
(3GPP TR 33.924 version 9.0.0 Release 9)

Reference
DTR/TSGS-0333924v900

Keywords
GSM, LTE, SECURITY, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

3GPP SA3 outlined the interworking of the operator controlled GBA with the Liberty Alliance Identity Management. This was sufficient for the time of writing, but now new additional systems are deployed and used. If we want to enable interworking of operator centric identity management, then smooth interworking with those new systems need to be outlined. If this is not done, then a seamless interworking is not possible on global scale and it would be difficult to leverage the existing customer base and security level that operators have.

1 Scope

The objective is to extend the current identity management as outlined in TS 33.220, TS 33.222, TS 29.109 and TR 33.980 with the latest developments on identity management outside of the 3GPP sphere. This will allow a better integration and usage of identity management for services in 3GPP and seamless integration with existing services that are not standardized in 3GPP. This report outlines the interworking of GBA and OpenID.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture".

[3] 3GPP TS 24.109: "Bootstrapping Interface (Ub) and Network Application Function Interface (Ua); Protocol Details".

[4] 3GPP TR 33.980: "Interworking of Liberty Alliance Identity Federation Framework (ID-FF), Identity Web Service Framework (ID-WSF) and the Generic Authentication Architecture (GAA)".

[5] 3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS)".

[6] 3GPP TS 33.223: "Generic Bootstrapping Architecture (GBA) Push Function".

[7] 3GPP TS 29.109: "Generic Authentication Architecture (GAA); Zh and Zn Interfaces based on the Diameter Protocol; Stage 3".

[8] OpenID Foundation "OpenID Authentication 2.0", http://openid.net/.

[9] OpenID Foundation "OpenID Attribute Exchange 1.0", http://openid.net/.

[10] OpenID Foundation "OpenID Provider Authentication Policy Extension 1.0", http://openid.net/.

[11] OASIS Reed, D.; McAlpin, D.: "Extensible Resource Identifier (XRI) Syntax v2.0"; http://www.oasis-open.org/

[12] OpenID Foundation, http://openid.net/foundation/

[13] OpenID Site Directory, http://openiddirectory.com/

[14] 3GPP TS 29.259: "Key Establishment between a UICC hosting device and a remote device".

[15] IETF RFC 4006, "Diameter Credit Control", http://tools.ietf.org/html/rfc4006

[16] W3C, "HTML 4.01 Specification", http://www.w3.org/TR/html401/

[17] IETF RFC 2617 (1999): "HTTP Authentication: Basic and Digest Access Authentication".

[18] IETF RFC 3761 (2004): "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1]. The GAA / GBA specific definitions are originated from [2] and the OpenID definitions are originated from [8]. In case of conflict [2] and [8] take precedence.

Attribute: An attribute is used in the OpenID Attribute Exchange service extension [9]. This extension provides a mechanism for moving identity related information between sites. An attribute is associated with a Subject Identifier. An attribute has a type identifier and a value. An attribute type identifier is a URI. An attribute value can be any kind of data.

Bootstrapping Server Function (BSF): A Bootstrapping Server Function (BSF) is hosted in a network element under the control of an MNO. BSF, HSS/HLR, and UEs participate in GBA in which a shared secret is established between the network and a UE by running a bootstrapping procedure. The shared secret can be used between NAFs and UEs, for example, for authentication purposes.

GBA User Security Settings: GUSS contains the BSF specific information element and the set of all application-specific USSs.

Identifier: An Identifier in OpenID is either an "http" or "https" URL, or an XRI [11]. OpenID [8] defines various kinds of identifiers depending on the context.

Network Application Function (NAF): A NAF is hosted in a network element. GBA may be used between NAFs and UEs for authentication purposes, and for securing the communication path between the UE and the NAF.

OpenID Provider (OP): An OpenID Provider (OP) is an OpenID Authentication Server on which a Relying Party relies for an assertion that the end user controls an Identifier.

OP Endpoint URL: The URL which accepts OpenID Authentication protocol messages, obtained by performing discovery on the User-Supplied identifier. This value must be an absolute HTTP or HTTPS URL.

Relying Party (RP): A Relying Party is a web application that wants a proof that the end user controls an Identifier.

User Supplied Identifier: An Identifier that was presented by the end user to the RP, or selected by the user at the OpenID Provider. During the initiation phase of the protocol, an end user may enter either their own Identifier or an OP Identifier. If an OP Identifier is used, the OP may then assist the end user in selecting an Identifier to share with the RP.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

AKA	Authentication and Key Agreement Protocol
AV	Authentication Vector
BSF	Bootstrapping Server Function
IdP	Identity Provider
GAA	Generic Authentication Architecture
GBA	Generic Bootstrapping Architecture
GPI	GBA Push Information
GUSS	GBA User Security Settings
HLR	Home Location Register
HSS	Home Subscriber Server
ME	Mobile Equipment

MNO	Mobile Network Operator
NAF	Network Application Function
OP	OpenID Provider
PAPE	Provider Authentication Policy Extension
RP	Relying Party
SLF	Subscriber Locator Function
SP	Service Provider
UE	User Equipment
USS	User Security Settings

4 Interworking of OpenID and GAA

4.1 Introduction

In this chapter we outline how the Generic Authentication Architecture, in particular, the Generic Bootstrapping Architecture (GBA) as specified in [2] can be utilized by the OpenID protocol as specified by [8]. The focus will lie on the impact to the network nodes of the Mobile Network Operator (MNO) and the service providers, the supported protocols and the message flows.

The general setting for the GBA – OpenID interworking is quite similar to the GBA – Liberty Alliance Interworking as outlined in [4]. The major difference is that OpenID as currently specified is more lightweight than the Liberty Alliance Web Service and Identity Federation Framework and therefore many well-known service providers have chosen OpenID for their identity management solution.

4.2 Architectural Descriptions

4.2.1 Generic Bootstrapping Architecture

In this section, we give a brief overview of the Generic Bootstrapping Architecture as described in TS 33.220 [2]. GBA enables automatic provisioning of shared keys between a User Equipment (UE) and an application server (Network Application Function (NAF)). The Home Subscriber Server (HSS) contains the long-term subscriber key. The Bootstrapping Server Function (BSF) is a GBA specific network function that resides in the home MNO of the user. It facilitates the use of AKA to bootstrap a new GBA master session key named Ks. The Subscriber Locator Function (SLF) is queried by the BSF in conjunction with the Zh interface operation to get the name of the HSS containing the required subscriber specific data.

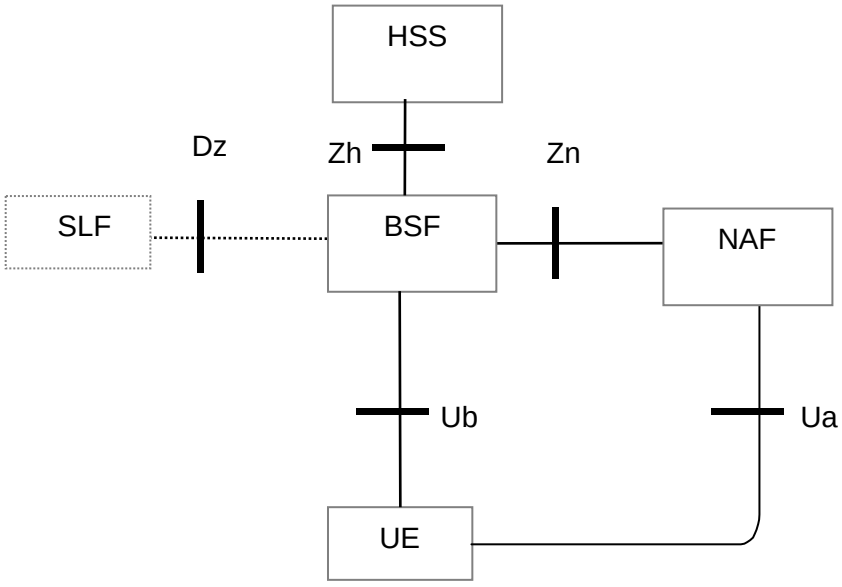

Figure 4.2.1 Simple Network Architecture for GBA

The basic message flow is as follows (for all details, parameter, formats, key derivation and error cases see [2] TS 33.220 section 4.5.2, which takes also precedence in case of conflict).

1. The UE sends a bootstrapping HTTP request towards the BSF over Ub reference point.
2. The BSF determines the IMPI based on the request. If there are several HSS the BSF may consult the SLF to determine the name of the correct HSS for this user.

   The BSF retrieves the complete set of GBA user security settings (GUSS) and one Authentication Vector AV over the reference point Zh from the HSS. The BSF might use a local copy of the GUSS (for details see [2]).

   The GUSS is optional to support in [2], but if GBA_U is intended to be used by the NAF, then the GUSS must be used.

   In the case that no HSS with Zh reference point is deployed, the BSF retrieves the Authentication Vector over the reference point Zh' from either an HLR or an HSS with Zh' reference point support.
3. Then BSF forwards the RAND and AUTN to the UE in the 401 message. This is to demand the UE to authenticate itself.
4. The UE checks AUTN to verify that the challenge is from an authorised network; the UE also calculates CK, IK and RES. This will result in session keys IK and CK in UE.
5. The UE sends another HTTP request, containing the Digest AKA response (calculated using RES), to the BSF.
6. The BSF authenticates the UE by verifying the Digest AKA response.
7. The BSF generates the master key material Ks by concatenating CK and IK. The BSF generates also the application specific keys Ks_(ext/int)_NAF. The bootstrapping transaction identifier B-TID value shall be also generated.
8. The BSF shall send a 200 OK message, including a B-TID, to the UE to indicate the success of the authentication. In addition, in the 200 OK message, the BSF shall supply the lifetime of the key Ks. The key material Ks is generated in UE.
9. Both the UE and the BSF shall use the Ks to derive the application specific key material Ks_NAF during the procedures as specified in clause 4.5.3 of [2]. Ks_NAF shall be used for securing the reference point Ua, in our case the authentication to the OP. If GBA_U is used, then the Ks_ext_NAF should be used for securing the Ua reference point.

   The UE and the BSF shall store the key Ks with the associated B-TID for further use, until the lifetime of Ks has expired, or until the key Ks is updated or until the deletion conditions are satisfied (see 4.4.11 in [2]).

GBA is used by many services like MBMS (Multimedia Broadcast Multicast Service), Enhanced MBMS, Access Network Discovery and Service Function (ANDSF), Open Mobile Alliance (OMA) XML Document Management, Presence Security etc. GBA can use USIM with GBA_U aware UICC application, ISIM or SIM cards. GBA and OpenID are independent of each other and the message flow above is given to introduce how GBA works and to ease the understanding for the interworking.

4.2.2 OpenID Architecture

The OpenID protocol [9] is specified by the OpenID Foundation [12]. It utilizes URL based Identifier.

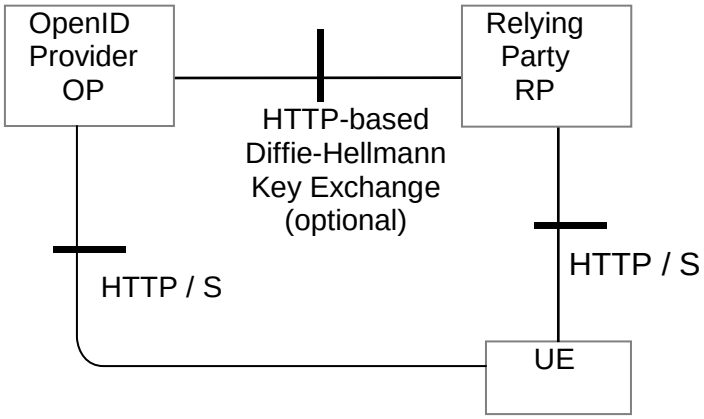

Figure 4.2.2 Simple OpenID Network Architecture

OpenID is HTTP POST and REPLY based. The basic message flow is as follows (for details and message parameters see OpenID Authentication 2.0 [8], which also takes precedence in case of conflict):

1. The browser in the UE sends a User-Supplied Identifier to the Relying Party.

2. The User-Supplied Identifier is normalized as described in Appendix A1 of [8]. The RP retrieves the address of the OP and performs a discovery of the OP Endpoint URL (based on the User-Supplied Identifier) that the end user wishes to use for authentication.

3. The RP and the OP may then establish a shared secret (called association) using the Diffie-Hellman Key Exchange Protocol [8]. The purpose of this shared secret is that the OP may sign subsequent messages and the RP can verify those messages.

NOTE1: This association is an optional feature in [8] and not required for interworking purposes.

4. The RP redirects the UE"s browser to the OP with an OpenID Authentication Request as defined in [8].

5. The OP establishes whether the end user is authorized to perform OpenID Authentication and wishes to do so.

NOTE2: [8] does not specify the manner and protocol used for authentication. In clause 4.4 we will outline how GBA based authentication using TS 33.222 [5] fits in here.

6. The OP redirects the UE"s browser back to the RP with either an assertion that authentication is approved or a message that authentication failed.

7. The RP verifies the information received from the OP. The user is now authenticated.

NOTE3: This Technical Report describes the interworking with OpenID version 2.0. If OpenID Authentication version 1.0 is used, then it should be noted that this specification was quickly replaced by OpenID Authentication version 1.1. OpenID version 1.1 made it clearer what was mandatory and what was optional to support. OpenID Authentication version 2.0 was a larger step forward. It allows usage of XRI to discover the OP, which was not possible in OpenID Authentication version 1.1. It has been reported that some OpenID Authentication version 1.1 implementations support XRIs. OpenID Authentication version 2.0 mandates the usage of the opened.ns field in the HTTP requests. If the value is absent or set to "http//openid.net/signon/1.1" or "http//openid.net/signon/1.0" instead of "http//specs.openid.net/auth/2.0", then the message should be interpreted using OpenID Authentication 1.1 Compatibility mode.

4.3 NAF and OpenID-IdP Co-location

The straightforward approach to combine the GBA and the OpenID Architectures is to co-locate the NAF functionality with the OP. The NAF functionality may be added in form of a library to the OpenID server.

NOTE: Other co-location approaches may require that the OP server has to support Diameter based reference point. The OPs as today are not telecommunication network nodes and hence it can not be assumed that they support the Diameter and underlying protocols just for the purpose of interworking with GBA.

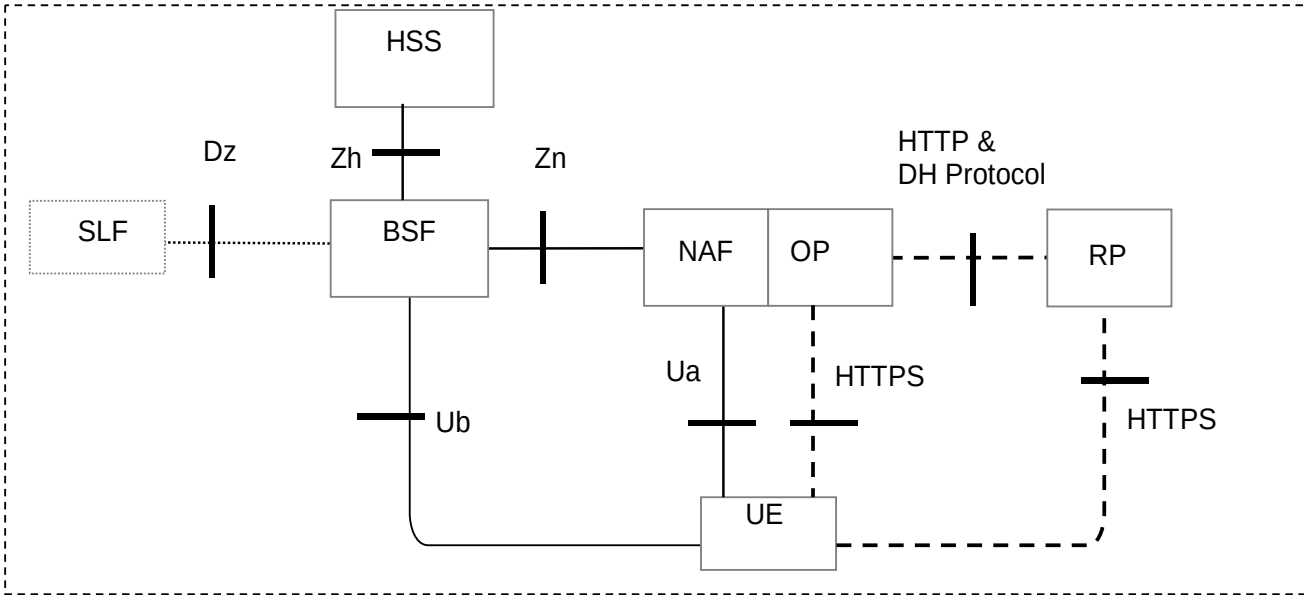

Figure 4.3.1 Simple OpenID Network Architecture

The dashed lines indicate the interfaces which originate from the OpenID architecture and the full lines are from the GBA Architecture. The UE utilizes the browser to communicate with the OP. The Ua protocol should be based on TS 24.109 [3] and TS 33.222 [5] section 5.3 and be supported by OP/NAF and UE. The OP/NAF and UE may support the variants of [5] clause 5.4 and 5.5. Even if the figure above shows two interfaces between the NAF/OP and the UE, there is actually only one, since the Ua is the application protocol, which in this case is the OpenID HTTPS protocol.

TS 33.222 [5] only allows HTTPS, the terminal uses the same connection to the NAF/OP server i.e. Ua and the HTTPS to the OP are in the same tunnel. Hence only HTTPS can be used HTTPS to the OP. Due to re-directs this implies that the connection to the RP should also be HTTPS based.

In the split terminal scenario, the browser and the entity performing the authentication do not reside in the same physical instance. The former UE is here split into a Browsing Agent (BA) that for example may reside in a PC and an Authenticating Agent (AA) that resides in the ME.

As above, the dashed lines indicate the interfaces which originate from the OpenID architecture and the full lines are from the GBA Architecture. The BA utilizes the browser to communicate with the OP. Depending on the scenario the BA and the AA may communicate by a local interface. The Ua protocol should be based on TS 24.109 [3] and TS 33.222 [5] section 5.3 and be supported by OP/NAF and AA with UICC. The OP/NAF and AA may support the variants of [5] clause 5.4 and 5.5.

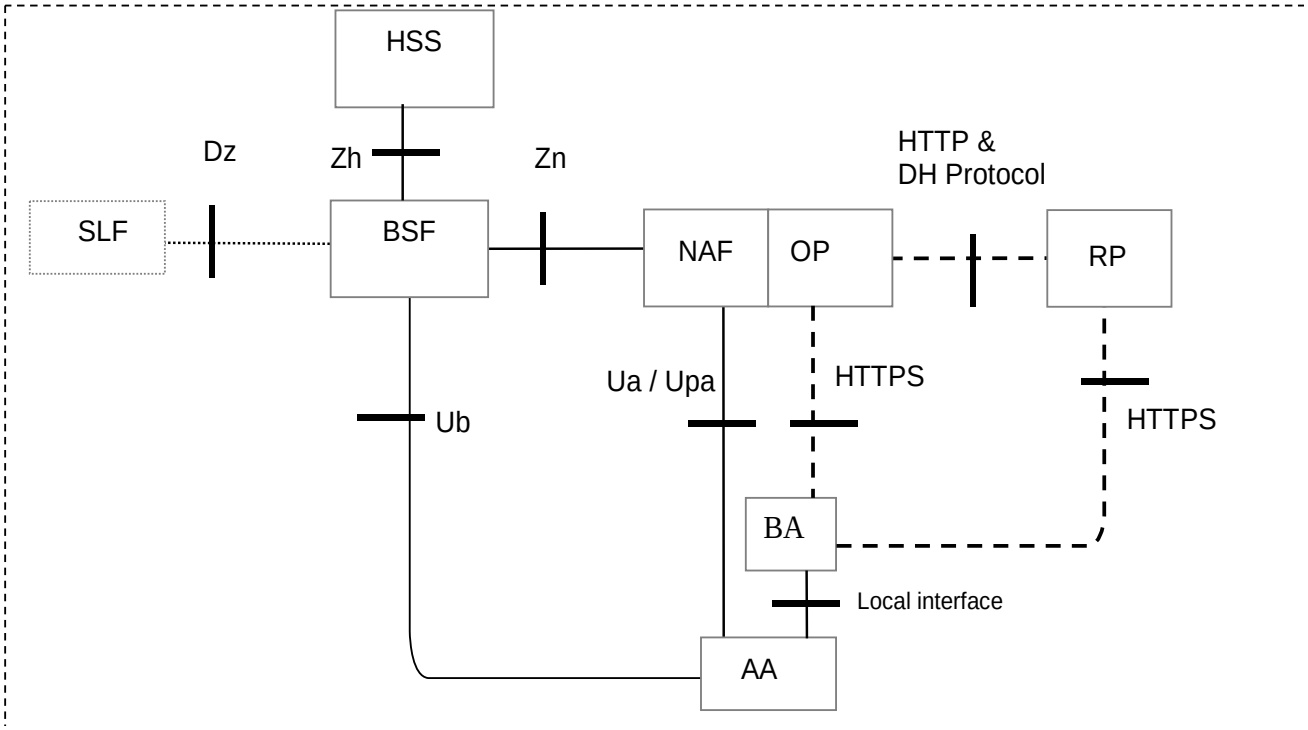

Figure 4.3.2 OpenID Network Architecture in split terminal scenario

4.4 Message Flow for Interworking Scenario

4.4.1 Message Flow for direct GBA Interworking Scenario

In the following a message flow is defined to allow the interworking of the GBA Architecture and the OpenID Architecture as defined in clause 4.3 and focuses on the case where the browser resides in the ME: The case, where the browser does not reside in the ME, will be outlined in the next clause.

1. The browser in the ME sends a User-Supplied Identifier to the Relying Party.

2. The User-Supplied Identifier is normalized as described in Appendix A.1 of [8]. The RP retrieves the address of the OP and performs a discovery of the OP Endpoint URL (based on the User-Supplied Identifier) that the end user wishes to use for authentication.

3. The RP and the OP may then establish a shared secret (called association), such as using the Diffie-Hellman Key Exchange Protocol. The purpose of this shared secret is that the OP can sign subsequent messages and the RP can easily verify those messages. The protocol between the OP and the RP lies outside of the 3GPP specifications, but for security reasons it is recommended to secure the RP OP interface properly

NOTE1: This association using Diffie Hellman is an optional feature in [8] and not required for interworking purposes. If the OP and RP do not both reside under the control of the same MNO, the usage of this option seems strongly advisable.

4. The RP redirects the ME"s browser to the OP with an OpenID Authentication Request as defined in chapter 9 in [8]. The RP inserts into the openid.claimed_id and into the opernid.identity fields the user supplied identifier of step 1.

5. The UE sends a HTTP GET request to the OP

6. The NAF initiates the ME authentication and responds with a HTTPS response code 401 'Unauthorized', which contains a WWW Authenticate header carrying a challenge requesting the UE to use Digest Authentication with GBA as specified in TS 33.222 [5] with server side certificates.

7. If no valid Ks is available, then the UE bootstraps with the BSF as described in TS 33.220 [2], which results in the possession of the UE of a valid Ks. From this the UE can derive the application specific (OpenID specific) Ks_(ext/int)_NAF key(s).

8. The ME generates a HTTP GET request to the NAF. The HTTP request carries an authorization header containing the B-TID received from the BSF.

NOTE2: If GBA push is used, the B-TID is not received from the BSF, but part of the GPI contains the P-TID which is used instead of the B-TID.

9. Using the B-TID and NAF_ID the NAF retrieves the shared application specific NAF key and optionally the USS (if GBA_U i.e. Ks_int/ext_NAF are used then the GUSS must be supported) from the BSF over the web service based Zn reference point. For details see TS 29.109 [7]. The NAF stores the B-TID, the cryptographic keys and the user supplied identifier to allow matching of the OpenID user session and the GBA session.

Since the OpenID is HTTP(S) based it is recommended that the NAF/OpenID server support for the interworking scenario the Web Service based Zn reference point as specified in [7] TS 29.109. It may support the Diameter based implementation of the Zn reference point.

NOTE3: It was assumed that the OPs are more likely to support web service based reference points then Diameter based reference points.

The USS may contain authorization information, which the NAF then retrieves. The OP establishes whether the end user is authorized to perform OpenID Authentication and wishes to do so based on the authorization information stored locally or in the USS. The USS may thereby act as a central authorization and privacy data store. In particular, it may contain information about the type of information are allowed to be shared with the relaying party. This authorization information may be contributed by the user but also by the operator based on their business relationship with the relaying party.

10. NAF/OP authenticates the user for OpenID using TS 33.222 section 5.3. The NAF redirects the browser to the return OpenID address i.e. the OP redirects the ME"s browser back to the RP with either an assertion that authentication is approved or a message that authentication failed. The response header contains a number of fields defining the authentication assertion which may be protected by the shared secret between OP and RP. The protection is especially important if the OP and the RP do not reside both in the same MNO network. NOTE4: At this point, the interworking diverges slightly from TS 33.222. In TS 33.222 the NAF responds with a 200 OK message.

11. The RP validates the assertion i.e. checks if the authentication was approved. The authenticated identity of the user is provided in the response message towards the RP. If a shared secret (association) was established in step 3, then it is now used to verify the message from the OP: If the validation of the assertion and the verification of the message (if the shared secret was used) are successful, then the user is logged in to the service of the RP.

If an operator deploys GBA Push as specified by [6] and not GBA, then the MNO may wish to establish the shared secret according to [6]. The step 7 would then be replaced by the GBA credential push message, after that the ME and NAF continue as outlined above.

The figure below outlines the message flow just described:

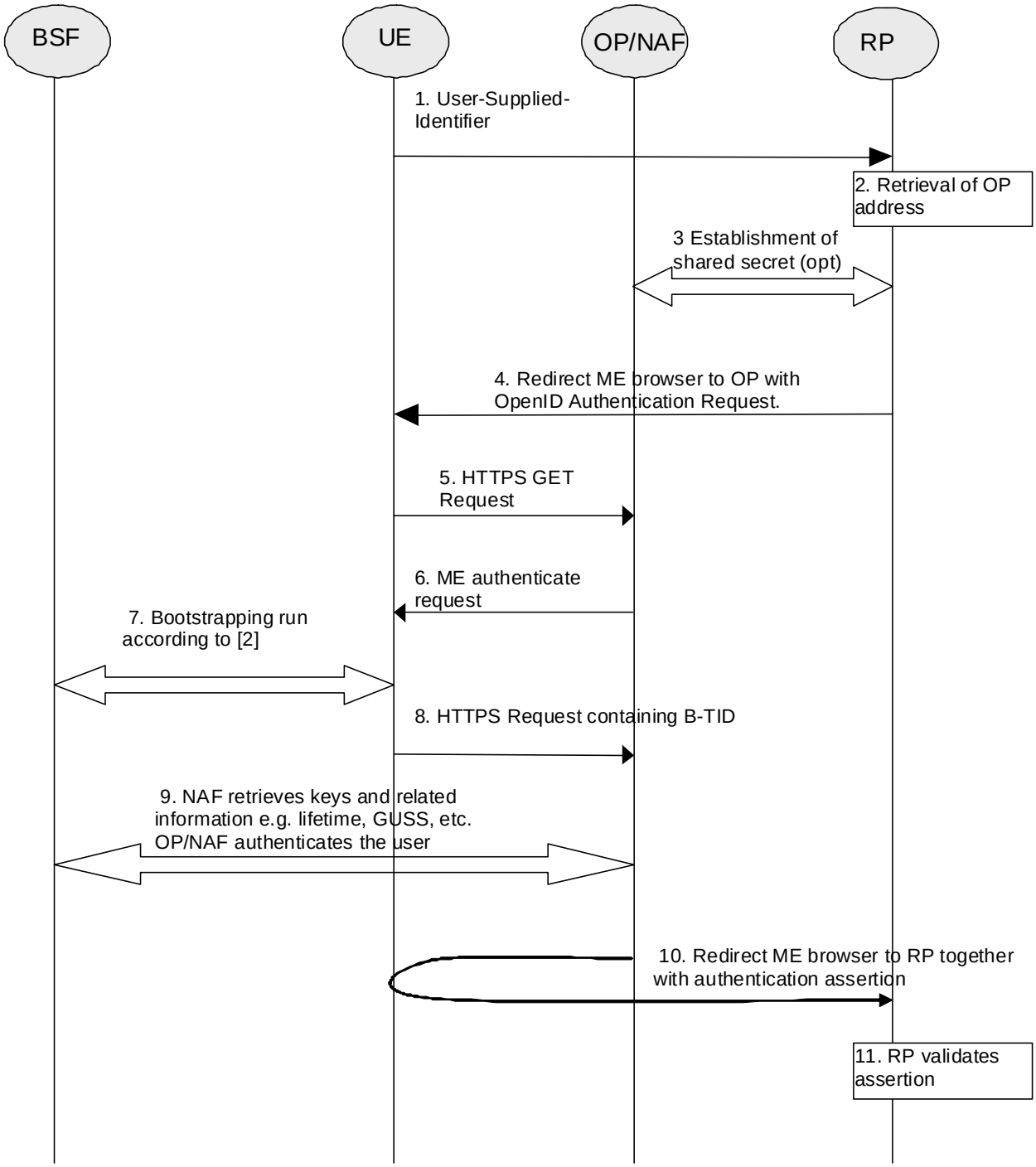

Figure 4.4.1 Interworking message flow for GBA / OpenID

4.4.2 Message Flow for Split Terminal GBA Interworking Scenario

This section will outline the split terminal implementation where the GBA agent (authenticating agent) is not located in the same device as the OpenID user Agent (browsing agent). It will detail 3 scenarios involving an authentication of the authenticating device and for which a successful completion will trigger a success status for the OpenID session on the browsing device.

In the first scenario the GBA session is initiated asynchronously by the server on the authenticating device via a GBA push message.

In a second scenario a GBA session is initiated asynchronously by the server on the authenticating device

In a third scenario, the GBA session is initiated by the authenticating device. This variant includes the approach, where the BA and the AA can utilize a local communication link to exchange a session identifier.

In the following a message flow is defined to allow the interworking of the GBA Architecture and the OpenID Architecture as defined in clause 4.3 and focuses on the scenarios where we have an Authenticating Agent (AA) and a Browsing Agent (BA) that do not reside in the same physical entity (the case where both reside in the same entity can be found in 4.2). The message flow in the 2 first scenarios will involve asynchronous notification of the authenticating Agent (AA). When registering to the OpenID service using a Split UE scenario, then the user has to provide an information of how to contact the AA i.e. phone number and operator, this information is mapped to the User-Supplied-Identifier. If no SplitUE scenario is utilized, then this information is not required.

1. The Browser Agent sends a User-Supplied Identifier to the Relying Party.
2. The User-Supplied Identifier is normalized as described in Appendix A.1 of [8]. The RP retrieves the address of the OpenID Provider (OP) and performs a discovery of the OP Endpoint URL (based on the User-Supplied Identifier) that the end user wishes to use for authentication.
3. The RP and the OP may then establish a shared secret (called association) using the Diffie-Hellman Key Exchange Protocol. The purpose of this shared secret is that the OP can secure subsequent messages and the RP can easily verify those messages.

   NOTE1: This association is an optional feature in [8] and not required for interworking purposes. If the OP and RP do not both reside under the control of the same MNO, the usage of this option seems advisable.

4. The RP redirects the Browsing Agent to the OP with an OpenID Authentication Request as defined in chapter 9 in [8].

Following this redirect operation the three scenarios will be now described in parallel. They correspond to three different ways to implement the split terminal function as follows:

Scenario 1 involves the use of a GBA push challenge which is pushed from the OP/NAF to the AA agent. The high level flow of operations for this scenario is described in Figure 4.4.2-1. Note, that the GBA Push challenge is not an HTTP response.

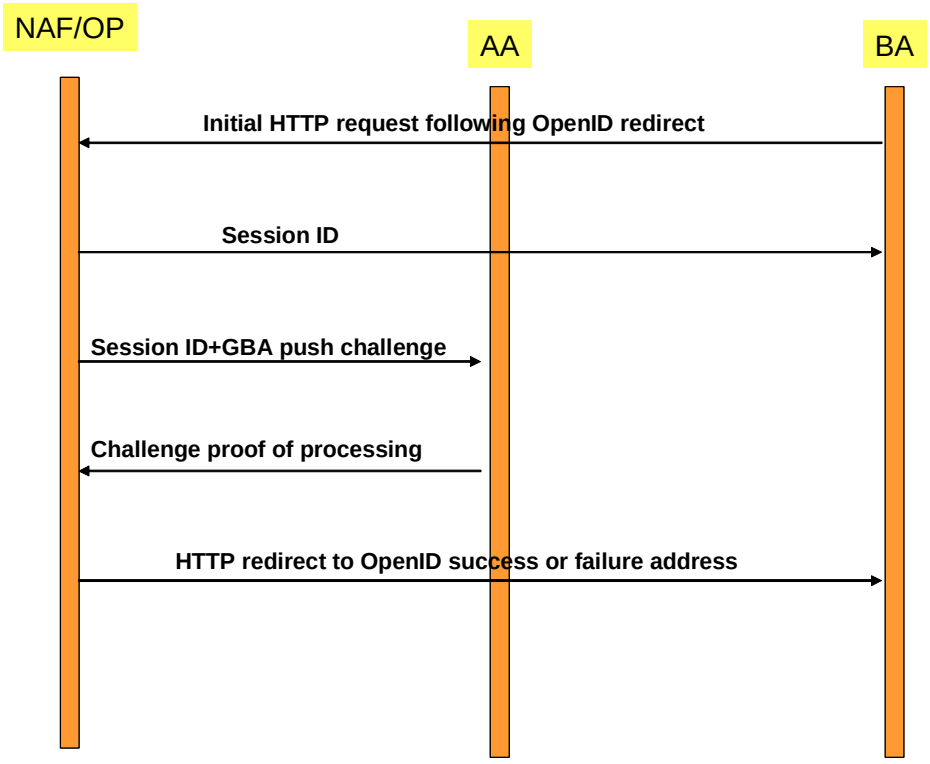

Figure 4.4.2-1: Scenario 1: use of GBA push challenge

- **Scenario 2** involves the use of a push request from the OP/NAF to the AA agent, triggering the AA to initiate a GBA session. The high level flow of operations for this scenario is described in Figure 4.4.2-2

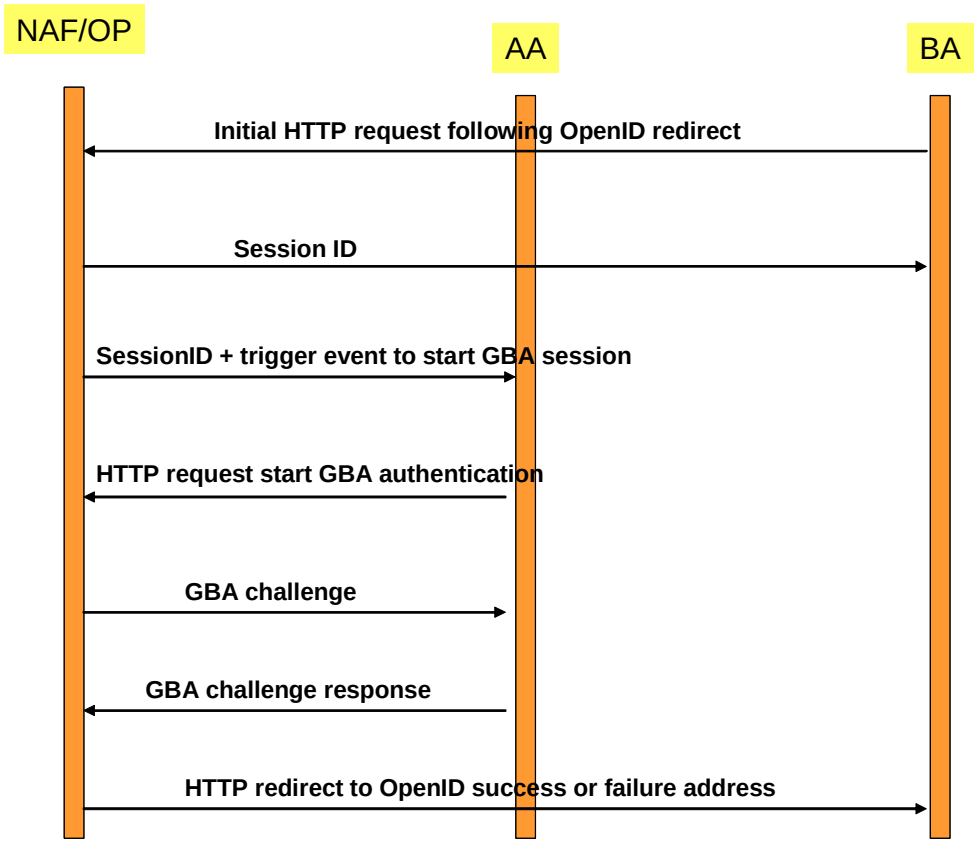

Figure 4.4.2-2: Scenario 2: use of push request to trigger GBA session

Scenario 3 involves local communication between the AA and the BA to share a session ID token generated by the OP/NAF. Following the retrieval of this session id token from the BA, the AA will initiate a GBA session with the NAF, providing the session ID token. Once GBA authentication is completed, the BA will be redirected to the OpenID success or failure URL. The high level flow of operations for this scenario for this scenario is described in Figure 4.4.2-3.

In this scenario, the AA and BA need to be securely connected and authenticated to each other, for example they may use a cable connection or BT Security.

Alternatively, the local communication may utilize GBA based security as outlined in TS 33.259 [14]. The BA would act as the remote device and the AA would take the role of the UICC holding device. If the BA has no valid Ks_local_device available, then the AA and the BA have to obtain the Ks_local_device as described in TS 33.259. This procedure results in the possession of the AA and BA of a valid Ks_local_device. The ME and GBA Agent can communicate in secure channel based using the Ks_local_device key..

NOTE2: The case where the AA sends the Ks_(ext)_NAF through a secure tunnel to the BA and the BA is using the credential is covered by the normal OpenID-GBA interworking. The OP would only exchange messages with the BA for Ks_(ext)_NAF usage and from the OP point of view the BA/AA would be treated then as one entity.

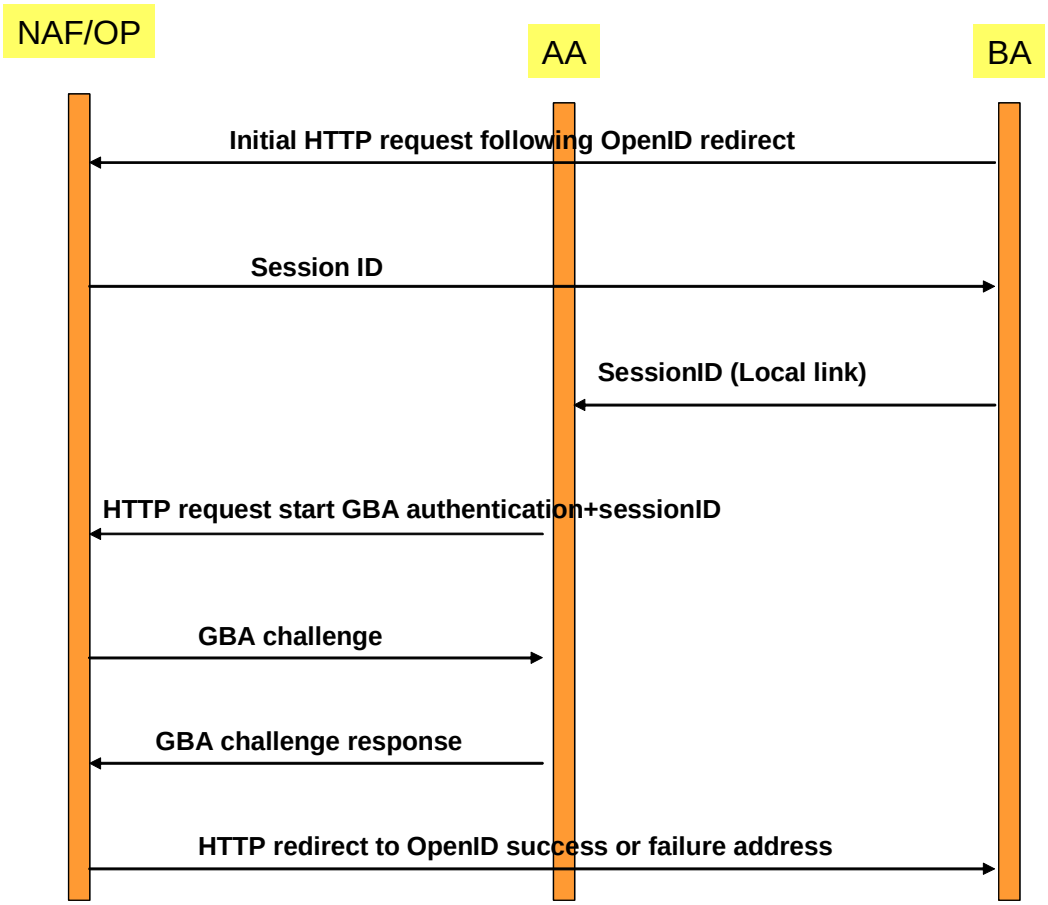

Figure 4.4.2-3: Scenario 3: linking of AA and BA sessions via session ID transferred from AA to BA

5. Following this redirection the BA sends a HTTP GET request to the OP/NAF.

6. The NAF generates an authentication session identifier. The NAF sends a session identifier to the BA.

   NOTE3: Depending on the implemented scheme, there might be different ways to pass the session identifier. One approach to send the session identifier is to use the realm attribute in the WWW-Authenticate header (see RFC 2617 [17]). If the NAF intends to populate the field with further information, then the session identifier should be at the end and separated with a ";".Alternatively, the session ID could be carried in the main body of the response for display by the BA.

   Scenarios 1 and 2 and 3: The NAF identifies the AA associated to the BA. This association has been defined previously, possibly at the time where the user has created his OpenID account and enabled usage of GBA (might be part of the registration procedure). The AA is identified by an endpoint address i.e. MSISDN which is itself dependant of the communication scheme used to push the Authentication request.

   NOTE4: The session identifier might be alphanumeric or a graphic or picture (or reference to one).

7. **Scenario 1:** The NAF/OP initiates a GBA push request to the AA. This push message contains a GPI used to establish a NAF SA. This request contains also the session identifier and the contact address of the OP/NAF Fully Qualified Domain Name (FQDN). Hence, the AA can set up an HTTP Request to the AA.

   Scenario 2: The NAF/OP initiates a push request to the AA. This request is just used to notify the AA to initiate a GBA authentication session with the NAF.

   Scenario 3: The BA pushes the session ID and the NAF contact address to the AA via the local link. If the BA and the AA have an established secure tunnel e.g. using TS 33.259 [14], then this could be utilized to send the session ID and the NAF contact address to the AA.

   NOTE5: The most common approach here is to use SMS for the push message to the AA, but also other Push methods might be used like WAP Push, SIP.

NOTE6: In scenario 2, the GPI may need to be encapsulated in a higher level protocol enabling to carry the additional information (session Identifier and NAF address)

8. **Scenario 2 and 3**: The AA receives the push message either from the BA or from the NAF. Upon reception of this push message, the AA will initiate an GBA bootstrapping run according to TS 33.220 (if no valid shared key is available) and then perform a HTTP based GBA authentication with the OP/NAF according to TS 33.222 as outlined in step 10. To that end, the AA sends then an HTTP GET request to the OP/NAF address contained in the push message.

   Scenario 1 continues in step 9.

9. **Scenario 1 and 2 and 3:** The AA and the BA session have to be mapped. In scenario 1 and 2 the user has to do this. In scenario 3, this may implicitly be done by using a secure connection between AA and BA and transferring the session id.

   NOTE7: The session identifier is used to make the link between the BA session and the AA session. The way this session identifier is processed varies according to the scenario:
   In scenarios 1 and 2, the session identifier may be displayed by both the AA and the BA. The user may visually check that the 2 identifiers displayed match..
   In Scenario 3, the Session identifier provided by the OP/NAF to the BA is presented by the AA to the BA. The link between the 2 sessions can therefore be made at the NAF/OP level without the user being involved in the matching operation.

   NOTE8: The actual methods of linking e.g. PIN, picture comparison is out of scope of this document.

   Scenario 1 continues in step 11.

10. **Scenarios 2 and 3**: The NAF initiates AA authentication by responding with an HTTP response code 401 "Unauthorized" which contains a WWW-Authenticate header carrying a challenge requesting the AA to use Digest authentication with GBA as specified in TS 33.222 [5] with server side certificates. The "realm" attribute starts with the prefix "3GPP-bootstrapping@" or "3GPP-bootstrapping-uicc@".

11. **Scenarios 2 and 3**: If no valid Ks is available, available to the AA, than the AA bootstraps with the BSF as described in TS 33.220 [2]. If a valid Ks key exists, than the AA computes the NAF specific key Ks_(ext/int)_NAF.

    Scenario 1: The Ks is derived as outlined in TS 33.223 [6]. Either way this results in the possession by the UE of a valid Ks. From this the UE can derive the application specific (OpenID specific) Ks_(ext/int)_NAF key(s). The key generation may be protected with a PIN code.

    NOTE9: In scenario 1 and 2, a PIN code or a manual user action is required to prevent risk of unauthorized background usage of the GBA authentication.
    In Scenario 3, the need of setting up a local link between the AA and the BA may result from a manual operation and the use of a manual user operation (PIN or key pressing) may be optional.

    NOTE10: If GBA push is used, than the B-TID is not received from the BSF, but part of the GPI contains the P-TID which is used instead of the B-TID. The P-TID is the GBA Push mirror to the B-TID.

12. **Scenarios 2 and 3**: The AA generates a HTTP GET request to the NAF/OP. The request carries an authorization header carrying the B-TID received from the BSF and a response to the challenge received in step 9 and computed with the (Ks_(ext/int)_NAF.

    Scenario 1: The AA responds to the NAF using the P-TID included in the GPI received from the NAF. The P-TID will point to a specific NAF SA in the NAF.

    NOTE11: In scenario 1, the response from the UE serves the purpose to prove to the NAF that the challenge carried by the GPI has been processed successfully. A higher level protocol not defined here is needed to carry this proof of successful processing.

13. **Scenario 2 and 3**: Using the B-TID, and its NAF-ID, the NAF retrieves the shared key Ks_(ext/int)_NAF and optionally the USS (if GBA_U is used, than the GUSS must be supported) from the BSF using the Zn interface, for details see TS 29.109 [7].

Since the OpenID is HTTP(S) based it is recommended that the NAF/OpenID server support for the interworking scenario the Web Service based Zn reference point as specified in [7] TS 29.109. It may support the Diameter based implementation of the Zn reference point.

Scenario 1: The NAF requests the shared key from the BSF as described in TS 33.223 [6] and TS 29.109.

NOTE12: It is assumed that the OPs are more likely to support web service based reference points then Diameter based reference points.

The USS may contain authorization information, which the NAF then retrieves. The OP establishes whether the end user is authorized to perform OpenID Authentication and wishes to do so based on the authorization information stored locally or in the USS.

14. The NAF/OP authenticates the user for OpenID using TS 33.222 [5] section 5.3. Then the NAF redirects the browser to the return OpenID address i.e. the OP redirects the ME"s browser back to the RP with either an assertion that authentication is approved or a message that authentication failed. The response header contains a number of fields defining the authentication assertion.

NOTE13: At this point, the interworking diverges slightly from TS 33.222. In TS 33.222 the NAF responds with a 200 OK message directly to the UE, here the BA does not reside in the UE.

15. The service provider (RP) checks the assertion (i.e. checks if the authentication was approved) possibly using previously defined shared secrets with the OpenId provider or by direct interrogation of the OpenID provider. Then the user is logged in to the service of the RP.

Figure 4.4.2-4 describes the detailed messages flow for scenario 1 involving the use of GBA push messages.

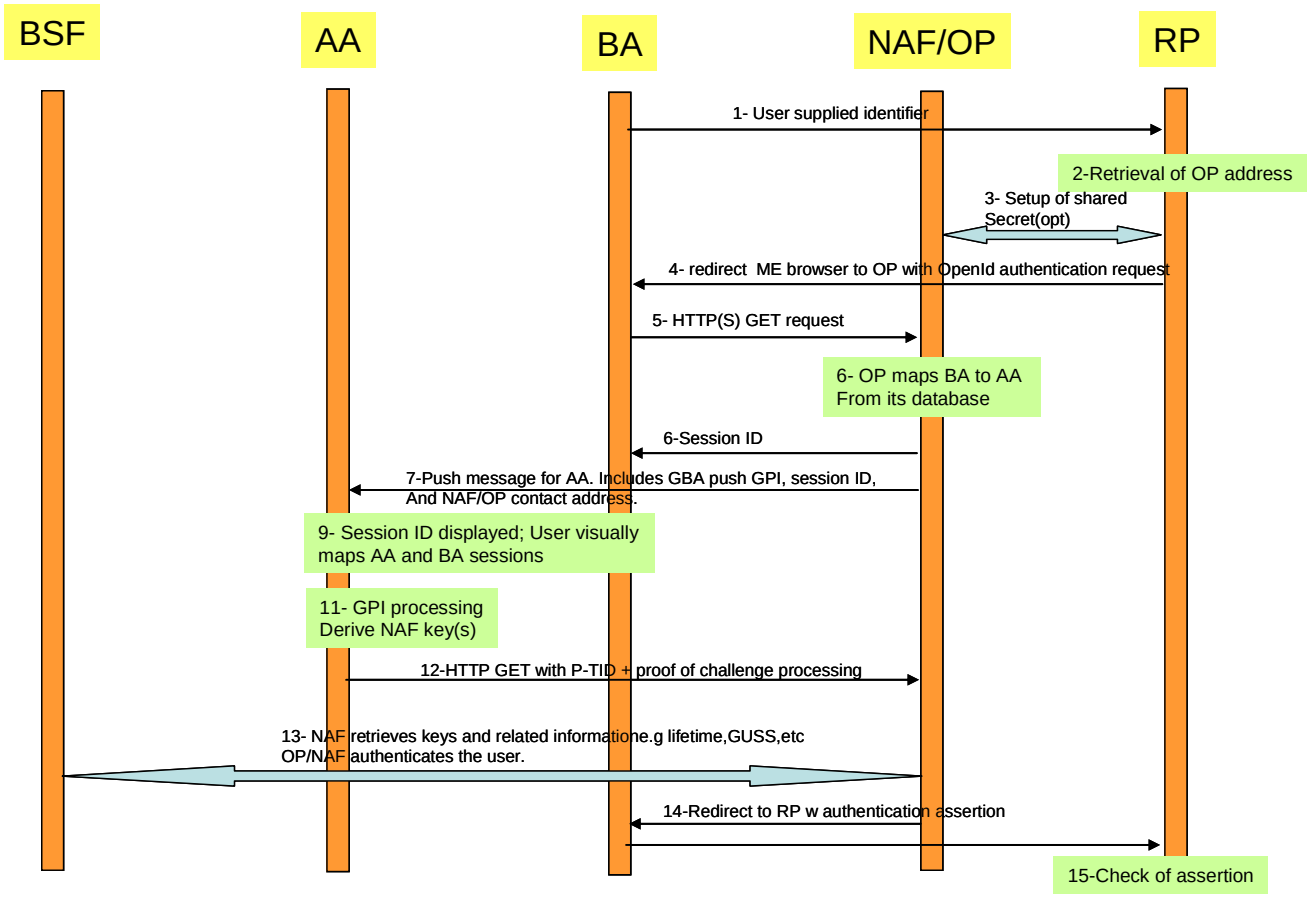

Figure 4.4.2-4: Detailed flow of operations for scenario1 (GBA push)

Figure 4.4.2-5 outlines the message flow based on the usage of TS 33.220 [2] for scenario 2

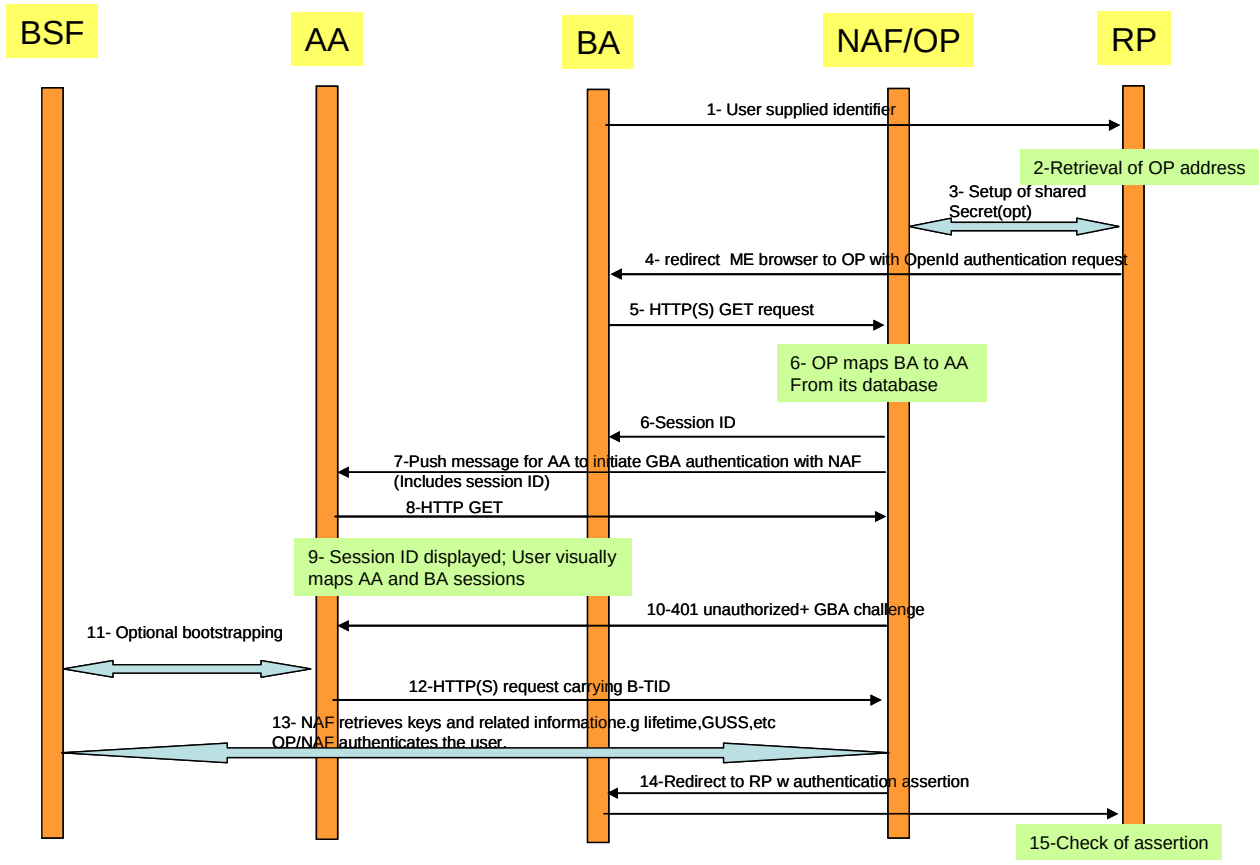

Figure 4.4.2-5: Detailed flow of operations for scenario 2 (push message to trigger GBA authentication)

Figure 4.4.2-6 outlines the message flow based on the usage of TS 33.220 [2] for scenario 3.

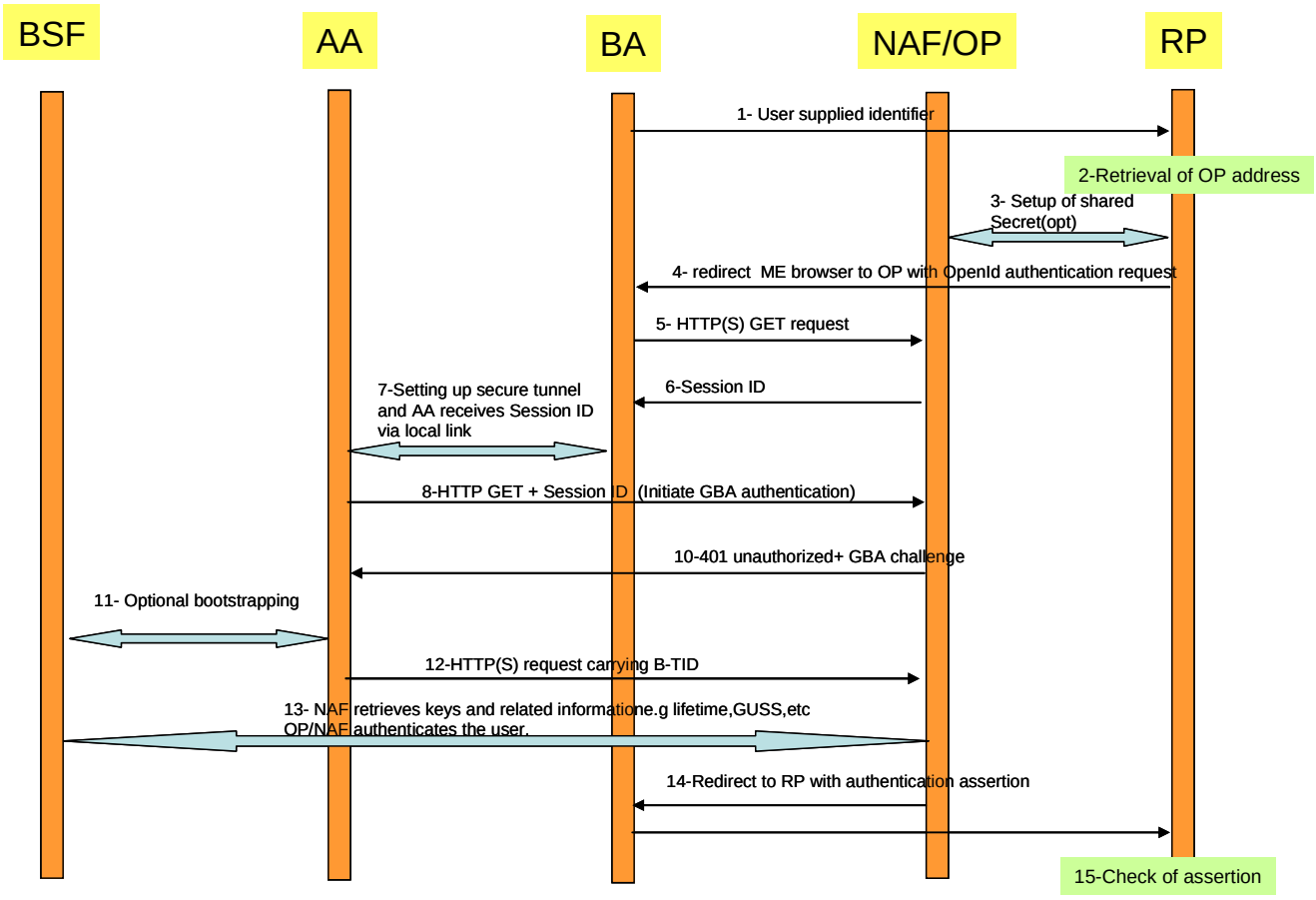

Figure 4.4.2-6: Detailed flow of operations for scenario 3 (local link between AA and BA)

4.5 Mapping of Concepts

4.5.1 Identifiers in OpenID and GBA

In OpenID we have two kinds of identifiers:

- An Identifier in form of a URI or XRI that is used between the OP and RP.

- An User Supplied Identifier that either has already the format of an URL, or if not it is normalized according to rules in [8] and converted into an Identifier.

In GBA we have the following identities:

- Bootstrapping Transaction Identifier (B-TID) which is identifies the generated key(s) and can be used to identify an authenticated GBA user.

- Optional identities are IMPI / IMSI or other kind of identifiers that are stored in the User Security Settings (USS)

The NAF / OP can receive the following identities from the BSF

- IMPI or MSISDN, if the BSF is configured to send one or both of them.

- Other user identifiers, but those need to be stored in the USS [7].

Hence, for the interworking of GBA with OpenID the user should use an MNO specific identifier recognizable by the NAF. It is recommended that when the user registers to an OpenID enabled service, then the browser thereby provides information about the operator domain to ensure that the OpenID discovery procedure can be utilizes successfully. This can be done by using MSISDN or pseudonyms (if supported).

If only the MSISDN is intended to be used as OpenID identifier by the user, then this mapping between OpenID identifier and MSISDN may be done by one of the following means:

First, the mapping may take place at the UE. This may be implemented as part of a browser plugin. The browser provides the discovered OpenID identifier along with the HTTP page when the accesses the relying parties webpage for usage with OpenID.

Alternatively, the RP may perform the MSIDN to OpenID identifier lookup. This step assumes that the user enters the MSISDN into the logon window or alternatively it is entered automatically as a browser feature.

The standard OpenID discovery step should remain unmodified and it requires an OpenID identifier. Hence, the MSISDN identifier has to be translated to the OpenID identifier and the ENUM procedure is utilized for this purpose. A description about ENUM can be found in RFC 3761 [18]

NOTE: The basic idea is convert a telephone number into a string such that it can be used for a DNS lookup and to identify available services. A new service would be the OpenID identifier that could then be retrieved as a DNS resource record.

Registration of this new service is required as shown in the following example taken from RFC 3761 [18]:

```
    $ORIGIN 3.8.0.0.6.9.2.3.6.1.4.4.e164.arpa.
       NAPTR 10 100 "u" "E2U+openid"
"!^.*$!http:exampleuser.livejournal.com!" .
```

Once the OpenID identifier is available to the Relying Party it is normalized, as described in Appendix A.1 of [8]. The RP retrieves the address of the OP and performs a discovery of the OP Endpoint URL (based on the User-Supplied Identifier) that the end user wishes to use for authentication.

If USSes are supported by the HSS, then the OpenID User Supplied Identifier may be stored in the USS.

4.5.2 Association Session Concept

In OpenID an association session is established between a Relying Party (RP) and the OpenID Provider. A shared secret is established, which is used to verify the subsequent protocol messages. The association session is initiated by a direct request from a RP to an OP using the OP Endpoint URL. The establishment of an association session between the RP and the OP Endpoint URL is optional. It allows optimization of the traffic processing between both nodes.

The RP sends an Association Session Request to the OP Endpoint URL. This message contains the parameter defined in [8]. The association type may be of type HMAC-SHA1 or of HMAC-SHA256. The openid:session_type may be set to either no-encryption, DH_SHA1, or DH-SHA256.

The OP answers with an Association Session Response. The assoc_handle is the association handle that is used as a key to refer to this association in the subsequent messages. For optimization purposes the RP may insert the B-TID in this field as a 255 character string. The expires_in field should be populated with the key lifetime of the B-TID, represented in an integer in base 10 ASCII.

4.5.3 Assertions

4.5.3.1 Positive Assertions

If an authorized end user wishes to complete the authentication, then the OP should send a positive assertion to the RP. This message is encoded using Section 17 of [16], send as HTTP Requests (GET or POST) and contain the openid:ns and openid.mode fields. The message may contain the OP-local identifier e.g. B-TID, MSISDN in the opened.identity field. Note that the end user may choose to use an OP-Local Identifier as a claimed identifier, for the GBA-OpenID interworking the B-TID or the MSISDN may be used. The MSISDN might be easier for the user from the usability point of view, but on the other hand the B-TID would provide a better privacy.

If the MSISDN or the B-TID is used in the claimed identifier field, then the same value should be used in the openid.identity field.

4.5.3.2 Negative Assertions

Negative assertions are sent by the OP to the RP according to [8], if the OP is unable to identify the end user or the end user does not or cannot approve the authentication request. This message is encoded using Section 17 of [16], send as HTTP Requests (GET or POST) and contain the openid:ns and openid.mode fields.. One cause might be that the user may with to explicitly terminate an ongoing session with his OpenID Provider or to abort a starting one.

Also, it might be possible that the operator may wish to terminate to support for OpenID usage for a particular user. This could be done by setting the USS by populating the authorization flag in TS 29.109 [7] accordingly, but this may not work for ongoing sessions. Alternatively, an explicit message could be send to the OpenID provider informing him about the termination of the support for OpenID usage for a particular user or cancellation of a particular session or credentials.

4.6 Use of GUSS and USS for OpenID

The USS may contain an authorization flag for the OpenID service. The OP may request the USS from the BSF and retrieve this authorization flag from the received USS as specified in TS 29.109 [7].

Alternatively, the OP may have a local authorization information for individual user"s. Conflicts between those policies should be avoided. If the OP is under MNO control then the MNO should decide where the authorization information is to be stored. If the OP is not under MNO control, the MNO and the OP should agree in their service agreement, where the authorization information should be kept or if they are combined with a logical AND.

NOTE: The MNO can always prevent the usage of GBA by not handing out the NAF keys.

The USS may contain the OpenID User Supplied Identifier in the user identities field. The GUSS or OpenID specific USS may also contain further identity information to be used together with the OpenID Attribute Exchange service extension [9]. This extension provides a mechanism for moving identity related information between sites. This kind of specific extension may be done in a proprietary manner or as part of an interworking customization.

Annex A (informative): Example for Charging via IdP

A service providing node (RP) in the OpenID interworking scenario may not necessarily want to support Diameter or RADIUS and underlying protocols. On the other hand, those services might wish to have the possibility for easy billing via the operators (if the operator offers this possibility). The RP most likely supports web service, hence we give an example how the RP can send the needed information to a suitable operator node (e.g. IdP if hosted by the operator, but that depends on the contractual agreements and the operator network topology) in the body of a web service message. Below are the web service versions of the Credit Control Request and Credit Control Answer messages (taken from [15]).

The receiving node in the operator network needs to extract the values of the web service fields and put them into the diameter fields of the same name. It adds the Diameter header data and the Diameter session id. For the web service based CCA the converse process has to take place. When receiving the CCA, the node, needs to extract the session-Id and determines the corresponding RP (i.e. when doing the conversion web service CCR to CCR conversion, he has to store this pair). The conversion node has also to store the token in connection with the session-id.

The Service-Context-Id in the CCR and web serviced based CCR should contain information about the RP to avoid errors in the sender checking. In the GBA based case the NAF-ID of the protocol conversion node will be placed in the Origin-Host in the CCR and web service CCR. The Service-Parameter-Info element in the web service CCA may be used and contain a token to enable the RP to recognize the corresponding answer to the request later on, this would not be needed for CCA (since we there don"t have an intermediary). In the all the messages the Auth-Application-Id element contains the OpenId identifier for the RP.

The RP in most cases does not have the subscription information. If the node taking care of the conversion has the subscription information i.e. an identity recognizable by the charging responsible network node, then he places it in the Subscription-Id field.

Below is an actual example of a potential web service based CCA and CCR:

```
<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions name="CCService"
                  targetNamespace="urn:3gpp:IdM:CCService:2009-10"
                  xmlns:typens="urn:3gpp:IdM:CCService:2009-10"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">

  <wsdl:types>

    <xsd:schema targetNamespace="urn:3gpp:IdM:CCService:2009-10">

      <!-- Extension element definition -->
      <xsd:complexType name="tExtension">
        <xsd:sequence>
          <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>

      <!-- Request WS Credit Control Request parameter definitions -->
      <xsd:element name="requestWSCCRInfo">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Origin-Host" type="xsd:string"/>
            <xsd:element name="Origin-Realm" type="xsd:base64Binary"/>
            <xsd:element name="Destination-Realm" type="xsd:string"/>
            <xsd:element name="Auth-Application-Id" type="xsd:string"/>
            <xsd:element name="Service-Context-Id" type="xsd:string"/>
            <xsd:element name="CC-Request-Type" type="xsd:integer"/>
            <xsd:element name="CC-Request-Number" type="xsd:integer"/>
            <xsd:element name="Destination-Host" type="xsd:string"/>
```

```
            <xsd:element name="User-Name" type="xsd:string"/>
            <xsd:element name="CC-Sub-Session-Id" type="base64Binary"/>
            <xsd:element name="Acct-Multi-Session-Id" type="xsd:string"/>
            <xsd:element name="Origin-State-Id" type="xsd:integer"/>
            <xsd:element name="Event-Timestamp" type="xsd:dateTime"/>
            <xsd:element name="Subscription-Id" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Service-Identifier" type="xsd:string"/>
            <xsd:element name="Terminatin-Cause" type="xsd:string"/>
            <xsd:element name="Requested-Service-Unit" type="xsd:string"/>
            <xsd:element name="Requested-Action" type="xsd:string"/>
            <xsd:element name="Used-Service-Unit" type="xsd:string"/>
            <xsd:element name="Requested-Service-Unit" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Multiple-Services-Indicator" type="xsd:string"/>
            <xsd:element name="Multiple-Service-Credit-Control" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Service-Parameter-Info" type="xsd:string" minOccurs='0'/>
            <xsd:element name="CC-Correlation-Id" type="xsd:string"/>
            <xsd:element name="User-Equipment-Info" type="xsd:string"/>
            <xsd:element name="Proxy-Info" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Route-Record" type="xsd:string" minOccurs='0'/>
            <xsd:element name="AVP" type="xsd:string" minOccurs='0'/>
```

```
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <!-- Request WS Credit Control Answer parameter definitions -->
      <xsd:element name="requestWSCCAInfo">
        <xsd:complexType>
```

```
          <xsd:sequence>
            <xsd:element name="Result-Code" type="xsd:integer"/>
            <xsd:element name="Origin-Host" type="xsd:string"/>
            <xsd:element name="Origin-Realm" type="xsd:base64Binary"/>
            <xsd:element name="Auth-Application-Id" type="xsd:string"/>
            <xsd:element name="CC-Request-Type" type="xsd:integer"/>
            <xsd:element name="CC-Request-Number" type="xsd:integer"/>
            <xsd:element name="User-Name" type="xsd:string"/>
            <xsd:element name="CC-Session-Failover" type="integer"/>
            <xsd:element name="CC-Sub-Session-Id" type="base64Binary"/>
            <xsd:element name="Acct-Multi-Session-Id" type="xsd:string"/>
            <xsd:element name="Origin-State-Id" type="xsd:integer"/>
            <xsd:element name="Event-Timestamp" type="xsd:dateTime"/>
            <xsd:element name="Granted-Service-Unit" type="xsd:string"/>
            <xsd:element name="Multiple-Service-Credit-Control" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Service-Parameter-Info" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Cost-Information" type="xsd:string"/>
            <xsd:element name="Final-Unit-Indication" type="xsd:string"/>
            <xsd:element name="Check-Balance-Result" type="xsd:integer"/>
            <xsd:element name="Credit-Control-Failure-Handling" type="xsd:integer"/>
            <xsd:element name="Direct-Debiting-Failure-Handling" type="xsd:string"/>
            <xsd:element name="Validity-Time" type="xsd:dateTime"/>
            <xsd:element name="Redirect-Host" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Redirect-Host-Usage" type="xsd:integer"/>
            <xsd:element name="Redirect-Max-Cache-Time" type="xsd:string"/>
            <xsd:element name="Proxy-Info" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Route-Record" type="xsd:string" minOccurs='0'/>
            <xsd:element name="Failed-AVP" type="xsd:string" minOccurs='0'/>
            <xsd:element name="AVP" type="xsd:string" minOccurs='0'/>
```

```
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <!-- Request WS Credit Control Requestinfo fault parameter definitions -->
      <xsd:element name="requestWSCCRInfoFault">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="errorCode" type="xsd:integer"/>
            <xsd:element name="errorText" type="xsd:string" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

    </xsd:schema>

  </wsdl:types>

  <wsdl:message name="requestWSCCRInfoMessage">
    <wsdl:part name="body" element="typens:requestWSCCRInfo"/>
  </wsdl:message>
```

```
  <wsdl:message name="requestWSCCAInfoMessage">
    <wsdl:part name="body" element="typens:requestWSCCAInfo"/>
  </wsdl:message>

  <wsdl:message name="requestWSCCRInfoFaultMessage">
    <wsdl:part name="body" element="typens:requestWSCCRInfoFault"/>
  </wsdl:message>

  <wsdl:portType name="CCServicePortType">
    <wsdl:operation name="requestWSCC">
      <wsdl:input message="typens:requestWSCCRInfoMessage"/>
      <wsdl:output message="typens:requestWSCCAInfoMessage"/>
      <wsdl:fault name="FaultName" message="typens:requestWSCCRInfoFaultMessage"/>
    </wsdl:operation>
  </wsdl:portType>


  <wsdl:binding name="CCServiceBinding" type="typens:CCServicePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="requestWSCCRInfo">
      <soap:operation soapAction="urn:3gpp:IdM:CCServiceAction:2009-10"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="FaultName">
        <soap:fault name="FaultName" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="CCService">
    <wsdl:port name="CCServicePort" binding="typens:CCServiceBinding">
      <!-- add SOAP address location URI below -->
      <soap:address location="http://add.here.uri.to/CCService"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

Editor"s note: Some of the CCA and CCR fields are of type group. It has to be validated, if the WS version can utilize string type for those.

Annex B:
Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2009-09	SA#45	SP-090529	--	--	Presentation to SA for Information	0.2.0	1.0.0
2009-12	SA#46	SP-090827			Presentation to SA for Approval	1.1.0	2.0.0
2009-12	--	--	--	--	Publication of SA-approved version	2.0.0	9.0.0

History

Document history		
V9.0.0	February 2010	Publication