Social Network Web Enabler

Approved Version 1.0 – 29 Mar 2016

Open Mobile Alliance

OMA-ER-SNeW-V1_0-20160329-A

Contents

Figures

Tables

1. Scope

This Enabler Release (ER) document is a combined document that includes requirements, architecture and technical specification of the Social Network Web (SNeW) Enabler.

The SNeW Enabler scope covers the following items:

- the requirements and reference architecture to allow interoperability between clients and servers and server-to-server federation of OMA Compliant SNs, supporting at least features such as:
    - profile discovery;
    - publication and sharing of contents, activities and reactions;
- the interface between a SNEW Client entity and a SNEW Server entity (intra-domain or UNI) that supports the above identified features;
- the interface between SNEW Server entities (inter-domain or NNI) that supports the above identified features;
- a set of Device APIs and Network APIs, to easily integrate OMA Compliant SN with external applications, as well as an appropriate privacy framework to control access to information through these APIs;
- a set of guidelines to reuse existing OMA enablers for providing additional features (e.g. profile search using OMA MSF);

In particular, with respect to interface specification, it is in the scope of this Enabler:

- to consider referencing OStatus-related specifications, in particular for server-to-server interactions;
- to consider referencing OpenSocial Social APIs, in particular for client-server interactions;
- to consider referencing OpenSocial JavaScript APIs as Device APIs;
- to consider reusing OAuth 2.0 and related specifications as privacy framework for APIs;
- to consider reusing OMA Push enabler to support notifications to SNEW Client entities (e.g. for reactions delivery, private message delivery, user status notification, etc).

Connections with External SNs are expected (through gateways implementing proprietary interfaces) but the definition of which External Social Network will be interconnected (and how) is out-of-scope of this activity.

SNeW Enabler will reuse as much as possible existing technologies.

2. References

2.1 Normative References

[ACCTURI] "The 'acct' URI Scheme", P. Saint-Andre, July 2013. Work in progress. URL:http://tools.ietf.org/html/draft-ietf-appsawg-acct-uri-06

[AS-ATOM] "Atom Activity Streams 1.0", M. Atkins, W. Norris, C. Messina, M. Wilkinson, R. Dolin, February 2011. URL:http://activitystrea.ms/specs/atom/1.0/

[AS-BASE-SCHEMA] "Activity Base Schema (Draft)", J. Snell, M. Atkins, D. Recordon, C. Messina, M. Keller, A. Steinberg, R. Dolin, May 2012. URL:http://activitystrea.ms/specs/json/schema/activity-schema.html

[AS-JSON] "Json Activity Streams 1.0", J. Snell, M. Atkins, W. Norris, C. Messina, M. Wilkinson, R. Dolin, May 2011. URL:http://activitystrea.ms/specs/json/1.0/

[AS-JSON-AUDIENCE] "Audience Targeting for JSON Activity Streams", J. Snell, March 2012. URL:http://activitystrea.ms/specs/json/targeting/1.0/

[AS-JSON-REPLIES] "Responses for Activity Streams", J. Snell, May 2012. URL:http://activitystrea.ms/specs/json/replies/1.0/

[ATOM] "The Atom Syndication Format", M. Nottingham et al, December 2005 URL:http://www.ietf.org/rfc/rfc4287.txt

[Autho4API_10] "Authorization Framework for Network APIs", Open Mobile Alliance™, OMA-ER-Autho4API-V1_0, URL:http://www.openmobilealliance.org/

[ENUM-ACCT] "ENUM Service Registration for acct URI", L. Goix, K. Li, May 2013. Work in progress. URL:http://tools.ietf.org/html/draft-goix-appsawg-enum-acct-uri-02

[GEORSS-SIMPLE] GeoRSS-Simple, URL:http://www.georss.org/simple

[HOST-META] "Web Host Metadata", E. Hammer-Lahav, B. Cook, October 2011 URL:http://www.ietf.org/rfc/rfc6415.txt

[MagicSig] Panzer, J., Laurie, B., and D. Balfanz, "Magic Signatures." URL:http://salmon-protocol.googlecode.com/svn/trunk/draft-panzer-magicsig-01.html

[OAUTHLRDD] "Link relation Type Registration for OAuth 2". W. Mills, February 2013. Work in progress. URL:http://tools.ietf.org/html/draft-wmills-oauth-lrdd-07

[OExchange-Spec] "OExchange Technical Specification" URL:http://www.oexchange.org/spec/

[OMA TLS] "OMA TLS Profile", Open Mobile Alliance™, OMA-TS-TLS_Profile-V1_1 URL:http://www.openmobilealliance.org/

[OMAPUSH] "Push Over The Air", Open Mobile Alliance™, OMA-TS-PushOTA-V2_3, URL:http://www.openmobilealliance.org/

[OMAPUSH-MSG] "Push Message", Open Mobile Alliance™, OMA-TS-Push_Message-V2_3, URL:http://www.openmobilealliance.org/

[OMAPUSH-PAP] "Push Access Protocol", Open Mobile Alliance™, OMA-TS-PAP-V2_3, URL:http://www.openmobilealliance.org/

[OMAPUSH-PPG] "Push Proxy Gateway Service", Open Mobile Alliance™, OMA-TS- PPGService –V2_3, URL:http://www.openmobilealliance.org/

[OMAPUSH-REST] "RESTful Network API for OMA Push". Open Mobile Alliance™. OMA-TS-REST_NetAPI_Push-V1_0, URL:http://www.openmobilealliance.org/

[OMAPUSH-SL] "ServiceLoading", WAP Forum™, WAP-168-ServiceLoad-20010731-a, URL:http://www.openmobilealliance.org/

[OMAPUSH-WRAPI] "Web Runtime API (WRAPI) – Push", Open Mobile Alliance™, OMA-TS-WRAPI_Push-V1_0, URL:http://www.openmobilealliance.org/

[OS-Core-API] OpenSocial Core API Server Specification 2.5.1, August 2013 URL:http://opensocial.github.io/spec/2.5.1/Core-API-Server.xml

[OS-Core-Data] OpenSocial Core Data Specification 2.5.1, August 2013 URL:http://opensocial.github.io/spec/2.5.1/Core-Data.xml

[OS-Social-API] OpenSocial Social API Server Specification 2.5.1, August 2013 URL:http://opensocial.github.io/spec/2.5.1/Social-API-Server.xml

[OS-Social-Data] OpenSocial Social Data Specification 2.5.1, August 2013 URL:http://opensocial.github.io/spec/2.5.1/Social-Data.xml

[OS-Spec] OpenSocial Specification 2.5.1, August 2013 URL:http://opensocial.github.io/spec/2.5.1/OpenSocial-Specification.xml

[POCO] Smarr J., "Portable Contacts 1.0 Draft C", August 2008 URL:http://portablecontacts.net/draft-spec.html

[PubSubHubbub] "PubSubHubbub Core 0.3", B. Fitzpatrick, B. Slatkin, M. Atkins, February 2010. URL:http://pubsubhubbub.googlecode.com/svn/trunk/pubsubhubbub-core-0.3.html

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2387] "The MIME Multipart/Related Content-type", E. Levinson, August 1998 URL:http://www.ietf.org/rfc/rfc2387.txt

[RFC2392] "Content-ID and Message-ID Uniform Resource Locators", E. Levinson, August 1998 URL:http://www.ietf.org/rfc/rfc2392.txt

[RFC2616] "Hypertext Transfer Protocol -- HTTP/1.1". Fielding, et al. June 1999. URL:http://www.ietf.org/rfc/rfc2616.txt

[RFC3403] "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database". M. Mealling, October 2002. URL:http://www.ietf.org/rfc/rfc3403.txt

[RFC4627] "The application/json Media Type for JavaScript Object Notation (JSON)", Crockford, D., July 2006. URL:http://www.ietf.org/rfc/rfc4627.txt

[RFC4685] "Atom Threading Extensions", J. Snell, September 2006. URL:http://www.ietf.org/rfc/rfc4685.txt

[RFC5234] **"**Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. January 2008, URL:http://www.ietf.org/rfc/rfc5234.txt

[RFC5724] "URI Scheme for Global System for Mobile Communications (GSM) Short Message Service (SMS)", E. Wilde et. al, January 2010, URL:http://tools.ietf.org/rfc/rfc5724.txt

[RFC5988] "Web Linking". M. Nottingham. October 2010. URL:http://www.ietf.org/rfc/rfc5988.txt

[RFC6116] "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)". S. Bradner, L. Conroy, K. Fujiwara, March 2011 URL:http://www.ietf.org/rfc/rfc6116.txt

[RFC6118] "Update of Legacy IANA Registrations of Enumservices". B. Hoeneisen, A. Mayrhofer, March 2011 URL:http://www.ietf.org/rfc/rfc6118.txt

[RFC6750] "The OAuth 2.0 Authorization Framework: Bearer Token Usage". M. Jones, D. Hardt, October 2012. URL:http://www.ietf.org/rfc/rfc6750.txt

[Salmon-Protocol] Panzer, J., "The Salmon Protocol" URL:http://salmon-protocol.googlecode.com/svn/trunk/draft-panzer-salmon-00.html

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

[TELURI] "The tel URI for Telephone Numbers". H. Schulzrinne, December 2004 URL:http://www.ietf.org/rfc/rfc3966.txt

[WebFingerProtocol] "Webfinger". P. E. Jones, G. Salgueiro, J. Smarr, July 2013. Work in progress. URL:http://tools.ietf.org/html/draft-ietf-appsawg-webfinger-16

[XRD] Extensible Resource Descriptor (XRD) Version 1.0. URL:http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html

[ZIP] "Zip File Format Specification". PKWare Inc.
URL:http://www.pkware.com/documents/casestudies/APPNOTE.TXT

2.2 Informative References

[3GPP 29.061] "Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN)", V6.15.0 (2008-12). 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; (Release 6)
URL:http://www.3gpp.org/ftp/Specs/archive/29_series/29.061/

[ActivityStreams] URL:http://activitystrea.ms/

[EUDataProtection] "How will the data protection reform affect social networks?", European Commission, 2012.
URL:http://ec.europa.eu/justice/data-protection/document/review2012/factsheets/3_en.pdf

[EUOpinionSN] "Opinion 5/2009 on online social networking", European Commission Article 29 Working Party, June 2009. URL:http://ec.europa.eu/justice/policies/privacy/docs/wpdocs/2009/wp163_en.pdf

[EUSNPrinciples] "Safer Social Networking Principles for the EU", European Commission, February 2009.
URL:https://ec.europa.eu/digital-agenda/sites/digital-agenda/files/sn_principles.pdf

[FOAF] The Friend of a Friend (FOAF) project. URL:http://www.foaf-project.org/

[MSRP] "The Message Session Relay Protocol (MSRP)", B. Campbell et al, September 2007,
URL:http://www.ietf.org/rfc/rfc4975.txt

[OSTATUS] OStatus W3C Community Group. URL:http://www.w3.org/community/ostatus/

[OStatus-Spec] Prodromou, E., Vibber, B., Walker, J., Copley, Z., "OStatus 1.0 Draft 2", August 2010
URL:http://www.w3.org/community/ostatus/wiki/images/9/93/OStatus_1.0_Draft_2.pdf

[SWAT0] Social Web Acid Test - Level 0
URL:http://www.w3.org/2005/Incubator/federatedsocialweb/wiki/SWAT0

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Aggregation	This is defined as the ability to connect to External Social Networks using the SN service specific credentials to access information related to friends (e.g. status, contact details, activities) via the exposed interfaces. Aggregation in SNs is related to only Inbound interaction.
Cross-posting	This is defined as the ability to connect to External Social Networks using the SN service specific credentials to share information (e.g. status, media) via the exposed interfaces. Cross-posting in SNs is related to only Outbound interaction.
External Social Network	A social network that is not-OMA Compliant and made available through either proprietary or non-proprietary mechanism and/or interfaces
Federated Social Networks	This is defined by a set of social networks that share some level of trust or set of rules, data formats and protocols, while, each social network retains its own administrative control and structure. The trust or set of rules binding the social networks govern the information that can be shared/searched/exchanged among users who are part of the respective social network.
Inbound/Outbound interactions	Inbound interactions relate to the concept of Aggregation of activities, reactions and media from External Social Networks, making thus possible to allow users that own accounts on multiple External SNs to access aggregated information about their friends (e.g. contact information, activities) on these networks. Outbound interactions, on the opposite, relate to the ability of Cross-posting activities and/or media to multiple External SNs. In this way, users could potentially share their activities over all their External SNs at once.
OMA Compliant Social Network	This is defined as a social network that conforms to the SNeW Enabler, essentially to ensure seamless interoperability and satisfaction of various actors in a social network. OMA Compliant Social Networks are natively Federated Social Networks.

3.3 Abbreviations

API	Application Programming Interface
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
MSISDN	Mobile Subscriber ISDN Number
MSRP	Message Session Relay Protocol
NNI	Network to Network Interface
OMA	Open Mobile Alliance
SN	Social Network
SNEW	Social Network Web
SNS	Social Network Service
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UNI	User to Network Interface

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language

4. Introduction

Thanks to Web 2.0 technologies, the World Wide Web has turned into a social space, moving from document links to people links. Individuals and organizations are now linked and leverage on user-generated content, communities, networking and social interaction.

Whilst the Web is becoming increasingly social, social networking itself is heavily fragmented due to the multitude of disparate services that are popular among users.

The existing hundreds of social network sites available on the Internet are based on centralised isolated systems and works standalone, ignoring other siblings, implementing a “walled garden” approach. Users on one Social Network (SN) cannot (easily) interact with users on another Social Network and people will often have to sign up for an account on multiple SNs to keep in touch with different groups of friends.

Unfortunately, or fortunately, there is no single social graph (or even multiple which interoperate) that is comprehensive and decentralized. Rather, these several disperse social graphs are often operated by a single company, sometimes small and with unproven record. This inconvenience ultimately results in a few very large networks with an inordinate amount of control over peoples’ most personal data and a lack of choice and privacy for users.

In this regard, such walled gardens are data silos where user data can easily be inserted, but only accessed and manipulated via proprietary interfaces for humans and machines. This further prevents users from moving easily from one SN provider to another as their social data cannot be shared across networks.

The SNeW Enabler aims at allowing large-scale deployments and interoperability of SNEW Clients and SNEW Servers in a timely manner, further guaranteeing social network federation so that users can easily communicate with users on other SNs and migrate their data. In particular, this document identifies a coherent subset of functionalities specific to SNeW representing a core specification.

The Figure 1 shows the overall ecosystem related to Social Network Web Enabler. In particular, the user (within the red circle) owns an account on a specific OMA Compliant SN ("his/her SN"), and interacts with users belonging either to his/her SN, or to another OMA Compliant SN, or to an External SN through the gateway functionality. Users access SNeW functionalities through multiple devices, each of them capable of running multiple applications.

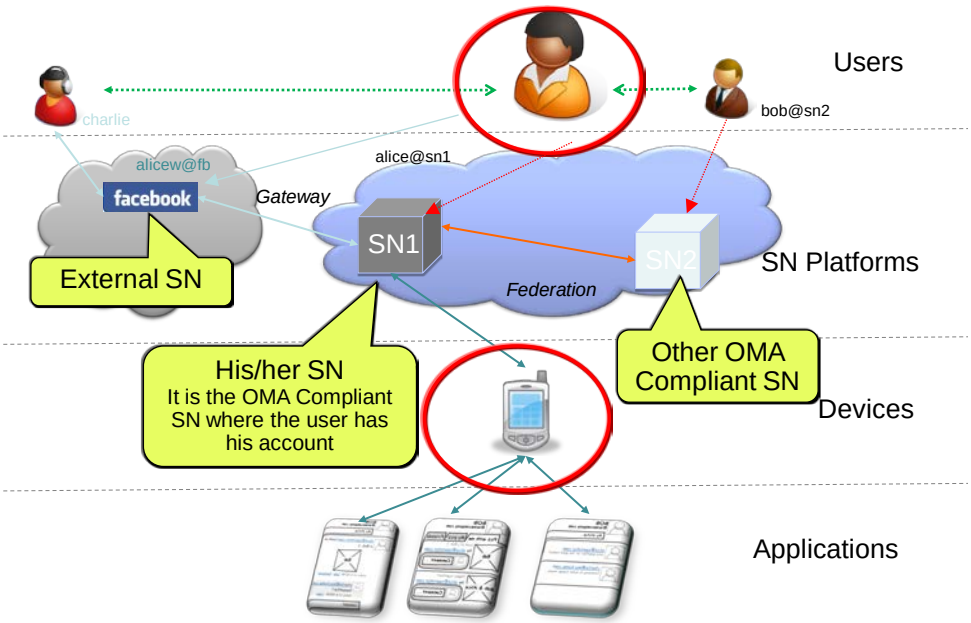

Figure 1: Social Network ecosystem.

4.1 Value Chain

In the ecosystem of mobile social networks, the interoperability will effectively connect all dots together and benefit all stakeholders in the value chain. The aspects of the Social Network Web (SNeW) value chain are provided in Figure 2.

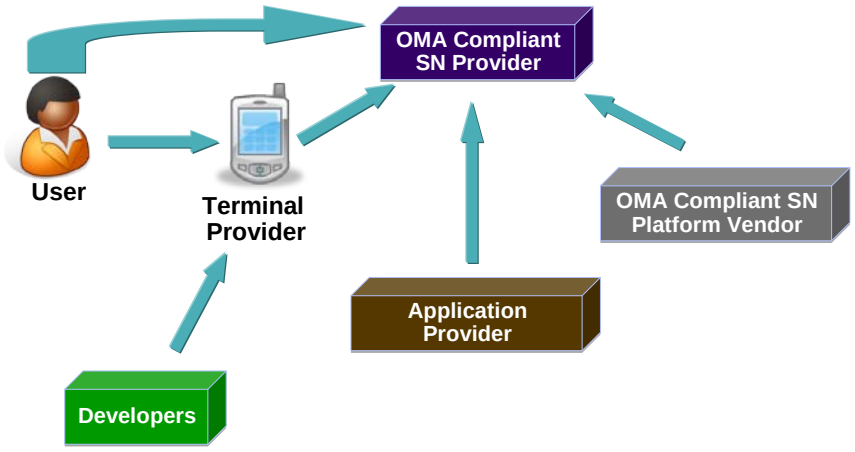

Figure 2: Social Network Web Value Chain

Users (Subscribers / end users) will have more options and freedom to use the mobile social network services which they feel more valuable without having to switch operators or devices. Users and their friends will be able to interact with each other more conveniently across mobile social networks.

OMA Compliant SN providers (e.g. operators) will have more customer loyalty and will be able to retain customers by letting subscribers communicate with friends across different social network services. Furthermore, OMA Compliant SN providers will also be able to increase their customer base and improve ARPU by offering unique, differentiated value of their mobile social network service to subscribers. Platform migration across vendors will be seamless for core features. At same time, they will be able to differentiate their offering by providing rich appealing features within a single social network, whilst all of them offering basic and common interoperable features.

Terminal providers will be able to provide devices that can connect to all social network services irrespective of the operators and Service Providers. Thus, they will benefit from customer loyalty, lowering the development cost, shortening the time to market, and increasing the market share by outreaching the broader market.

Application providers will be able to access end users' social data from the mobile social network, thus enabling them to provide richer services and applications to attract the user and thereby generate revenue.

Developers will also benefit this value chain by providing richer applications or widgets according to the mobile social network APIs. It will be for example possible to arrange an application store combined with mobile social network for expanding the usage of the applications and widgets.

4.2 Version 1.0

The version 1.0 of the SNeW Enabler SHALL cover at least:

- the interoperability between clients and servers and server-to-server federation of OMA Compliant SNs, supporting at least features such as:
    - profile discovery;
    - publication and sharing of contents, activities and reactions;
- a set of Network APIs, to easily integrate OMA Compliant SN with external applications, as well as an appropriate privacy framework to control access to information through these APIs

5. Requirements (Normative)

This section identifies the requirements necessary for the Social Network Web Enabler to support end-to-end interoperability across different devices, networks, service providers and network operators.

The requirements below are related to the Use Cases described in Appendix B.

5.1 High-Level Functional Requirements

This section identifies the high-level functional requirements for the Social Network Web Enabler.

Label	Description	Release
SNEW-HLF-001	The SNeW Enabler SHALL allow a user to perform at least the following activities: • posting a text (e.g. update his status, write a blog post); • upload a multimedia content (e.g. image, video, audio); • share a link (e.g. video link, audio link, website link); • perform check-in (e.g. check-in a place). **Informative Note**: additional domain specific activities (e.g. social TV, social gaming) may be defined/supported together with other OMA enablers.	1.0
SNEW-HLF-002	The SNeW Enabler SHALL allow a user to perform at least the following reactions on existing activities: • like; • share; • comment.	1.0
SNEW-HLF-003	The SNeW Enabler SHALL allow a user to follow/unfollow another user's activities and reactions on his/her SN.	1.0
SNEW-HLF-004	The SNeW Enabler SHALL allow a user to follow/unfollow another user's activities and reactions on another OMA Compliant SN.	1.0
SNEW-HLF-005	The SNeW Enabler SHALL allow a user to perform activities and/or reactions in relation with at least the following entities: • user; • content; • place.	1.0
SNEW-HLF-006	The SNeW Enabler SHOULD allow a user to perform activities and/or reactions in relation with at least the following entities: • group; • event.	1.0
SNEW-HLF-007	The SNeW Enabler SHALL allow a user to be identified through a globally unique identifier across OMA Compliant SNs through the use of at least one of the following form: • MSISDN; • Email-like address (user@domain).	1.0
SNEW-HLF-008	The SNeW Enabler SHALL allow notifying users (”push”-mode) of activities or reactions they are related to (e.g. friend updates, picture tagging notification, comment received, etc), including activities or reactions from users of other OMA Compliant SNs.	1.0
SNEW-HLF-009	The SNeW Enabler SHOULD support interacting with External SNs in Inbound and/or Outbound directions (gateway functionality).	1.0
SNEW-HLF-010	The SNeW Enabler SHALL provide means for users to share links when browsing external websites (e.g. through a “share” button).	1.0

SNEW-HLF-011	The SNeW Enabler SHOULD allow a user to enquire through which users he/she can contact another user within his/her SN. **Informational Note**: User A would like to get acquainted with User B. User A has a friend User 1, who is also a friend of User 2, and User 2 is in the friend list of User 3. User 3 knows User B very well. Thus, we have the following user chain: User A -> User 1 -> User 2 -> User 3 -> User B. With this user chain, User A will be able to be introduced to User B. The internal process of how these users in the chain are identified is out of scope	Future Release
SNEW-HLF-012	The SNeW Enabler SHALL allow a user to exchange private one-shot messages with one or more other users on his/her SN. **Informational Note**: One-shot messages refer to messages sent to another user not related to a conversation session. On the other side, conversation messages enable users to exchange messages in a conversation session.	1.0
SNEW-HLF-013	The SNeW Enabler SHOULD allow a user to exchange private one-shot messages with one or more others users on other OMA Compliant SN.	Future Release

5.2 High-Level Management Requirements

This section identifies the high-level management requirements for the Social Network Web Enabler. The detailed requirements are further identified in the following sections according to the detailed functions identified.

5.2.1 User Management Requirements

Label	Description	Release
SNEW-UMG-001	The SNeW Enabler SHALL allow a user to perform the following actions: • Create an account and obtain an identity on an OMA Compliant SN; • Delete his/her account and all related information (e.g. profile, activities) from his/her SN.	1.0
SNEW-UMG-002	The SNeW Enabler SHALL allow a user to delete specific activities and reactions he/she performed previously.	1.0
SNEW-UMG-003	The SNeW Enabler SHOULD allow a user to update the activities and reactions he/she performed previously (e.g. the user has uploaded an image with a misspelled title and he/she wants to fix the title).	1.0
SNEW-UMG-004	The SNeW Enabler SHOULD allow to search for a user identity within an OMA Compliant SN, e.g. based on email, name, or other attribute.	1.0
SNEW-UMG-005	The SNeW Enabler SHALL support user's profile to be discovered across OMA Compliant SNs based on user identity and according to user's privacy settings.	1.0
SNEW-UMG-006	The SNeW Enabler SHOULD allow a user to block reception of specific or all information/messages: – from specific users – from specific OMA Compliant SNs – from specific External SNs **Informational Note**: the exact set of information/messages to be blocked is FFS	Future Release
SNEW-UMG-007	The SNeW Enabler SHOULD allow a user to export his/her social information (e.g. profile, content, activities) from an OMA Compliant SN and import it into the same or another OMA Compliant SN	1.0

5.2.2 Generic Management Requirements

Label	Description	Release
SNEW-GMG-001	The SNeW Enabler SHOULD allow a user to report abuse/inappropriate activities, reactions, content, users and applications providing a reason for the report.	1.0

5.3 Gateway Requirements

This section identifies the detailed gateway requirements for the Social Network Web Enabler. These requirements are only applicable when the gateway functionality is supported as in SNEW-HLF-009.

Label	Description	Release
SNEW-GWR-001	The SNeW Enabler SHALL allow a user to associate his/her identity on External SNs with his/her SN account.	1.0
SNEW-GWR-002	The SNeW Enabler SHALL allow a user to delete the association between his/her identity on External SNs and his/her SN account.	1.0
SNEW-GWR-003	The SNeW Enabler SHALL support importing/aggregating at least the following information from the External associated SNs, when supported by the External SNs: • user own activities (e.g. status update, link sharing, etc) and reactions (e.g. comments, replies, mentions, etc.) • friends activities and reactions.	1.0
SNEW-GWR-004	The SNeW Enabler SHALL allow a user to select External SNs for Outbound interactions.	1.0
SNEW-GWR-005	The SNeW Enabler SHALL support performing at least the following actions towards the selected External SNs, when supported by the External SNs: • posting user own activities (e.g. status update, link sharing, content upload, etc) and reactions (e.g. comments, replies, mentions, etc.).	1.0

5.4 Device related Requirements

This section identifies the requirements for the device-side of the Social Network Web Enabler related to the support of multiple applications running in a device that interact with SNs. The interactions with External SNs are only applicable when the gateway functionality is supported as in SNEW-HLF-009.

Label	Description	Release
SNEW-DEV-001	The SNeW Enabler SHALL allow different applications running in a device to bidirectionally communicate with OMA Compliant SNs and, provided that the gateway functionality is supported, with External SNs.	1.0
SNEW-DEV-002	The SNeW Enabler SHALL rely on a component on the device, which centralizes communications between device applications and the OMA Compliant SN and (through the gateway functionality) External SNs	1.0
SNEW-DEV-003	The SNeW Enabler SHALL rely on a component on the device to locally cache information uploaded to or downloaded from an OMA Compliant SN / External SNs to make it available to local device applications.	1.0
SNEW-DEV-004	When a device application requests information, the SNeW Enabler SHALL ensure that the component on the device first accesses locally cached information on behalf of the device application, subject to the policies relevant for that specific application, prior to have it retrieved from the network.	1.0
SNEW-DEV-005	When the component on the device receives information from the OMA Compliant SN, the SNeW Enabler SHALL ensure that: – when the information relates to a deleted action, that action is deleted from the local cache so that it is not available anymore to local device applications; – when the information relates to an updated action, that action is updated in the local cache so that its new version becomes available to local device applications.	1.0

SNEW-DEV-006	The SNeW Enabler SHALL rely on a component on the device to queue user's activities and reactions when offline (or inaccessible due to server problems) and process them automatically when back online (or accessible) removing them from the queue when processed, based on user preferences.	1.0
SNEW-DEV-007	The SNeW Enabler SHALL allow the user to manage (e.g. remove, edit) items from the activities and reactions queue on the device.	Future Release

5.5 Device API Requirements

This section identifies the requirements for the Social Network Web Enabler related to Device APIs. The interactions with External SNs are only applicable when the gateway functionality is supported as in SNEW-HLF-009.

Label	Description	Release
SNEW-DAPI-002	The SNeW Enabler SHALL provide a Device API to device applications that allows them to specify which kind of information/notifications it wants to receive and which kind of activities it wants to perform (see SNEW-HLF-001 to SNEW-HLF-004) from/towards each OMA Compliant SN / External SN.	Future Release
SNEW-DAPI-003	The SNeW Enabler SHALL provide a Device API to allow device applications to specify data traffic and time constraints (e.g. frequency of polling or notifications from OMA Compliant SNs and External SNs), subject to policies (e.g. service provider policies, device configuration policies).	Future Release
SNEW-DAPI-004	The SNeW Enabler SHALL ensure that device applications using Device API obtain user authorization for the requested interactions (see SNEW-DAPI-002).	Future Release
SNEW-DAPI-005	The SNeW Enabler SHALL ensure that only authorized device applications are able to use the SNeW Device API.	Future Release
SNEW-DAPI-006	The SNeW Enabler SHALL provide a Device API to device applications that allows them to perform activities and/or receive information/notifications with OMA Compliant SNs and External SNs	Future Release
SNEW-DAPI-007	The SNeW Enabler SHALL allow access to the SNeW Device API to multiple device applications simultaneously.	Future Release
SNEW-DAPI-008	The SNeW Enabler SHALL support callback mechanisms in the SNeW Device API: – to allow device applications to register callbacks, – to callback device applications (based on previous registration). **Informational Note**: Callback events consist of specific notifications/ messages from OMA Compliant SNs, External SNs and local events (e.g. communication failures).	Future Release

5.6 Network API Requirements

This section identifies the requirements for the Social Network Web Enabler related to Network APIs.

Label	Description	Release
SNEW-NAPI-002	The SNeW Enabler SHALL provide a Network API to third-party applications that allows them to specify which kind of information it wants to receive and which kind of activities and/or reactions it wants to perform (see SNEW-HLF-001 to SNEW-HLF-004) from/towards an OMA Compliant SN.	1.0
SNEW-NAPI-003	The SNeW Enabler SHALL ensure third-party applications be authorized before interacting through the SNeW Network API.	1.0

SNEW-NAPI-004	The SNeW Enabler SHALL ensure that third-party applications using the SNeW Network API obtain user authorization for the requested interactions (see SNEW-NAPI-002) with an OMA Compliant SN.	1.0

5.7 Security and Privacy Requirements

This section identifies the requirements for the Social Network Web Enabler related to security and privacy. In particular, such requirements encompass authentication, authorization and privacy mechanisms and settings.

Label	Description	Release
SNEW-SEC-001	The SNeW Enabler SHALL allow a user to define the privacy level of his activities or reactions to control visibility by other users, including at least: • Public: accessible by all users; • Private: accessible only by the user owning the information • Followers: accessible only by the user's followers **Informational Note:** other users can pertain to the same or another OMA Compliant SN.	1.0
SNEW-SEC-002	The SNeW Enabler SHALL allow a user to grant and revoke permissions to external applications (interacting with the SNeW Enabler) to access part or all of his/her own related information, including activities and reactions. **Informational Note:** the exact definition of such permissions is FFS.	1.0
SNEW-SEC-003	The SNeW Enabler SHALL support authentication mechanisms to authenticate requests among the various SNeW components (e.g. client-server, server-to-server), as well as with external applications interacting with the SNeW Enabler.	1.0

6. Architectural Model

6.1 Dependencies

The SNeW Enabler has the following dependencies to other OMA Enablers:

- The OMA Push Enabler as described in [OMAPUSH] and [OMAPUSH-PAP];

6.2 Architectural Diagram

The following diagram illustrates the Functional Components and Interfaces of the SNeW Enabler.

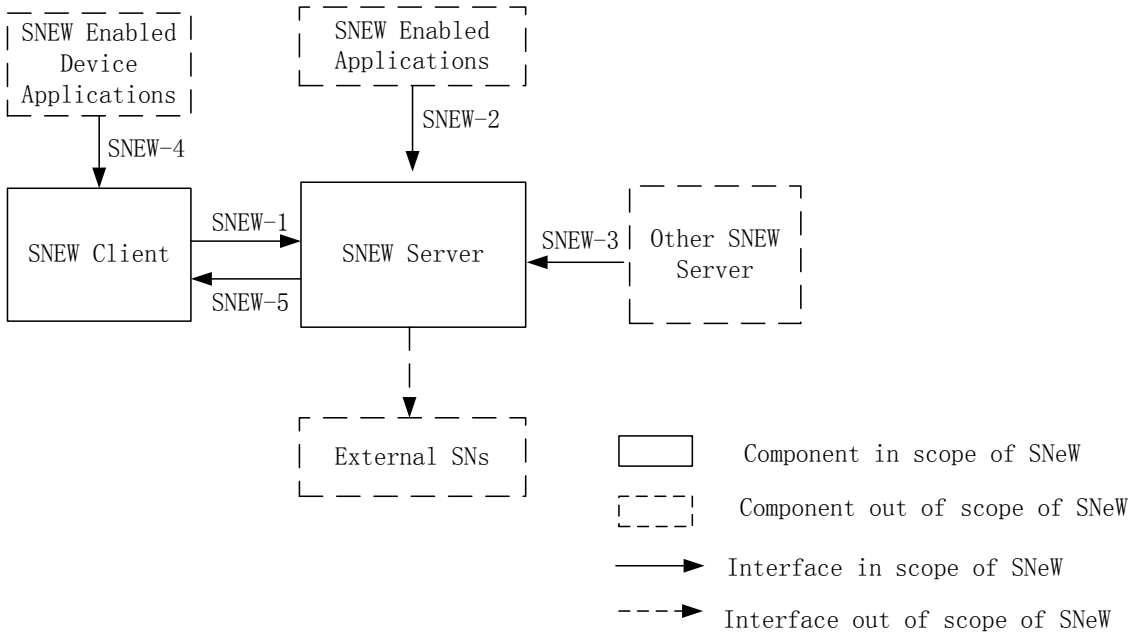

Figure 3: Social Network Web reference model

6.3 Functional Components and Interfaces definition

The SNeW Enabler consists of mandatory components (SNEW Server and SNEW Client) and interfaces exposed by those components. All other depicted components and interfaces are not specified in this document, but are shown for a better understanding of the interactions with the SNeW Enabler.

Both SNEW Server and SNEW Client are mandatory functional components of the SNeW Enabler.

6.3.1 SNeW Enabler Functional Components

6.3.1.1 SNEW Server

The SNEW Server is a SNeW Enabler component resident in the network (outside the device) and is the entry point to the enabler for all the requests coming from an SNEW Client. It represents the central node of an OMA Compliant SN that federates with other OMA Compliant SNs.

It also exposes Network APIs to 3rd party applications (SNEW Enabled Applications) through the SNEW-2 interface in order to exchange social network service information requested by these applications. Through SNEW-2 interface the SNEW

Server is responsible for handling requests from SNEW Enabled Applications related to local users (under his authority). As such, the SNEW Server can reject requests related to users of other SN. However, depending on the nature of the required information (e.g. the information required relates to a local user who has friends on other SNs), the SNEW Server can interact with other federated OMA-Compliant SN or External Social Network.

The SNEW Server exposes SNEW-1, SNEW-2 and SNEW-3. It uses SNEW-5 exposed by SNEW Client and SNEW-3 exposed by other SNEW Servers for federation.

When using the SNEW-5 interface, the SNEW Server acts as Push Initiator when using OMA-PAP or as Push Server when using PUSH-OTA.

The core function is the main functionality of the SNEW Server responsible for providing mobile social network service to the end user. As user social data centre, it provides management features for user accounts and user activity & reactions (including content management), and the related access to this data to third party applications based on user permissions. It provides native federation features with other SNEW Servers, allowing “local” users to interact with users on other OMA Compliant SNs.

The gateway function is an optional functionality responsible for interacting with External SNs. It enables users to interconnect with external SNs on which they already have an account (e.g. Facebook, Twitter) using the proprietary interfaces of such networks.

The gateway function supports inbound interactions to aggregate activities, reactions and media from External networks, and outbound interactions for the ability of cross-posting activities and/or media to multiple External SNs. In this sense the gateway function implements the required protocol & data format translation capabilities in relation with the supported External SNs. It also provides the related management features so that users can associate their “local” account with their account on External SNs.

6.3.1.2 SNEW Client

The SNEW Client is a SNeW Enabler component resident on the device, which enables the user to connect to the SNEW Server of an OMA Compliant SN and interact with users on the same or other OMA Compliant SNs, or on External SNs (if the gateway function is provided by the SNEW Server).

It has the responsibility for interacting bidirectionally with SNEW Server to exchange social network service information, by using SNEW-1 interface exposed by the SNEW Server and by exposing SNEW-5 interface to receive push notifications, acting as a Push Client in case there is no Push Client in the device.

It also exposes Device APIs to 3rd party device applications (SNEW Enabled Device Applications) through the SNEW-4 interface in order to exchange social network service information requested by these applications. The SNEW Client is responsible for providing the appropriate information to each SNEW Enabled Device Application based on the information requested by them and the authorization from the user.

The SNEW Client caches information coming from SNEW Enabled Device Applications, in addition to posting it to the SNEW Server, in order to make it available to any other interested SNEW Enabled Device Application on the same device, without the need to download this piece of information from the SNEW Server later on.

Thus the information provided by the SNEW Client to the SNEW Enabled Device Applications may come either from the SNEW Server or from information locally cached. The decision on whether providing cached information or downloading it from the SNEW Server is up to the SNEW Client, based on policies relevant for that specific application (e.g. requested by the application itself or enforced by service provider policies).

Furthermore, when receiving requests from SNEW Enabled Device Applications to be sent to the SNEW Server, the SNEW Client relies on the cache to temporarily queue information prior to send it to the SNEW Server, for example to defer the request until back online (in case of server or connectivity problems) or for throttling purposes (e.g. regulate bandwidth over the air).

6.3.2 Entities External to the SNeW Enabler (Informative)

6.3.2.1 SNEW Enabled Applications

SNEW Enabled Applications represent 3rd party applications using the SNEW Network APIs exposed by an SNEW Server.

These SNEW Enabled Applications typically require some social network service information for their operation and obtain it by interacting with an SNEW Server. SNEW Enabled Applications choose the target SNEW Server to interact with based on the user on whose behalf they act (e.g. their authoritative SNEW Server), based on the target user(s) of interest or based on local policies.

Examples of SNEW Enabled Applications are web applications running within a user agent, web applications running on an external server or server native applications.

6.3.2.2 SNEW Enabled Device Applications

SNEW Enabled Device Applications represent 3rd party applications residing on the device using the SNEW Device APIs exposed locally by the SNEW Client.

These SNEW Enabled Device Applications typically require some social network service information for their operation and rely on the SNEW Client on the device to interact on behalf of a user with the SNEW Server corresponding to that user. Thus the SNEW Client relieves the various SNEW Enabled Device Applications running in the same device from implementing social network-related functionalities singularly and optimizes their operation.

Examples of SNEW Enabled Device Applications are described in section B.4 and include web applications running within the device's user agent or device native applications.

6.3.2.3 Other SNEW Server

Other SNEW Server represents a peer SNEW Server entity belonging to another OMA-Compliant SN.

6.3.2.4 External Social Networks

See definition.

6.3.3 Interfaces definition

6.3.3.1 SNEW-1

This interface is exposed by the SNEW Server. It is used by the SNEW Client to interact with the OMA Compliant SN for performing some core functionalities related to mobile social networking, covering: user authentication, authorization, user information management (e.g.post and retrieve activities and/or reactions, user account management). Typically, interactions from SNEW Client to SNEW Server over this interface are triggered by SNEW Enabled Device Applications.

6.3.3.2 SNEW-2

This interface is exposed by the SNEW Server to the network-side SNEW Enabled Applications. SNEW-2 covers:

- expose a Network API to access and manage user information (e.g. retrieving user activities and/or reactions)
- share links from external websites,
- allow users to export and/or import their personal information.

6.3.3.3 SNEW-3

This interface is exposed by the SNEW Server to other SNEW Servers for realizing the federation between OMA Compliant SNs. It enables the activity, reactions and media exchange between two different OMA Compliant SNs.

As such, this interface is both exposed, and used, by SNEW Servers.

6.3.3.4 SNEW-4

This interface is exposed by the SNEW Client to SNEW Enabled Device Applications running on the device, according to their specific needs and permissions. As such, SNEW-4 represents a Device API to support SNEW Enabled Device Application access and manage user information with SNEW Client.

Informational Note: The technical definition of SNEW-4 interface is postponed to a future release.

6.3.3.5 SNEW-5

This interface is exposed by the SNEW Client and is used by the SNEW Server to push notification/information (e.g. content, user activities, reactions) to the SNEW Client.

7. Interface Definitions

7.1 SNEW-1

SNEW-1 SHALL follow the REST protocol of the OpenSocial Social API Server Specification [OS-Social-API] with the clarifications provided below.

With specific reference to [OS-Spec] section 4

- SNEW Servers SHALL act as Social API Server and MAY act as OpenSocial Container
- SNEW Clients SHALL implement the Social API Server specification as client to interact with SNEW Servers

In particular, SNEW Servers and SNEW Clients SHALL conform to [OS-Social-API] with the following clarifications:

- SHALL support the creation and deletion of relationships and of a person, and SHOULD support the update of person (relationships) properties, according to [OS-Social-API] section 2.1. See section 7.1.1 for more details about this functionality.
- SHOULD support the Groups Service according to [OS-Social-API] section 2.2. When supported, SNEW Servers and SNEW Clients SHALL support the creation and deletion of groups.
- SHALL support the deletion of activity entries, and SHOULD support the update of activity entries, according to [OS-Social-API] section 2.3. See section 7.1.2 for more details about the support of the ActivityStreams Service, including for uploading content.
- SHALL support the Messages Service according to [OS-Social-API] section 2.7. In particular, SNEW Servers and SNEW Clients SHALL support the retrieval and deletion of messages and SHOULD support the update of messages.

In addition, the following functionalities SHOULD be supported by SNEW Clients and SNEW Servers:

- The abuse reporting functionality and the related management. When supported, this functionality SHALL be implemented as part of the ActivityStreams Service, according to section 9.2.4.1 at least on the following object types: “activity”, “image”, “video”, “person” and “application”.
- The Gateway functionality according to section 7.1.4.
- The setup of subscriptions according to section 7.1.5.

Furthermore, when referring to identifiers over SNEW-1 as OpenSocial “Global-Id”, SNEW Clients & SNEW Servers SHALL support IRIs as a valid Global-Id format (including user identifiers according to section 9.1.1.1), also in order to be consistent with the other protocols and/or formats used in this specifications that rely on URI/IRIs as global identifiers. Support for the Global-Id format defined in [OS-Core-Data] section 2.8 is NOT RECOMMENDED.

SNEW Clients & SNEW Servers SHALL support at least the “@all”, “@followers”, “@following”, “@friends” and “@self” Group-Id values defined in [OS-Social-Data] section 3.8.

Details on response codes and errors over SNEW-1 are further provided in section 7.1.6.

7.1.1 People Service support

7.1.1.1 Generic considerations

The following clarifications apply to the support of [OS-Social-API] section 2.1:

- When retrieving a person and/or a list of persons (section 2.1.1 of [OS-Social-API]):

- each Person object returned SHOULD include the number of “followers”, “following” and “friends” according to [AS-JSON-REPLIES] section 4, and SHOULD include the relationship indication, if any, with the requesting user as per section 9.3.1
- when retrieving the user’s own information, in case the Gateway functionality is supported, the Person object SHOULD include the list of current connections with External SNs for that user under the “accounts” attribute, according to the format and extensions used in section 9.4.1

- When updating a relationship (section 2.1.3 of [OS-Social-API]) the REST-URI-Fragment SHALL contain a Related-User-Id

- When creating a person (section 2.1.2.1 of [OS-Social-API]) using Group-Id value “@self” the “id” and “displayName” field values in the Person object in the request body may indicate respectively a preferred requested user-id and display name subject to server policies.

- When deleting a person (section 2.1.4 of [OS-Social-API]) SNEW Servers and SNEW Clients SHALL support requests with no REST-Request-Payload. In particular:
  - to indicate a specific relationship to be deleted, the REST-URI-Fragment SHALL contain a Related-User-Id. Note that after this deletion the initial user may still have other types of relationships with the same related user.
  - when the REST-URI-Fragment does not contain any other information than the Group-Id, that specific relationship with all contacts under this group SHALL be removed. Note that after this deletion the initial user may still have other types of relationships with the same related users.

See section 9.3 for further details related to the Person objects exchanged over SNEW-1 through this service.

See Appendix G.1.5 for examples related to this service.

7.1.1.2 Profile content upload

When creating or updating a person, some content (e.g. profile picture) may need to be uploaded. This can be easily provided in two steps as follows. At step one, content is uploaded by means out of scope of this specification that provides back a reference to the newly uploaded content as URI. At step two, a Person object is subsequently published on the People Service containing the URI referencing the content already uploaded.

As such, the method of content upload defined in section 7.1.3 SHALL be supported by SNEW Clients and SNEW Servers over the People Service. This method allows uploading one or more content contextually to the creation or update of a profile or relationship information.

7.1.1.3 Account management

SNEW Servers MAY apply restrictions and/or additional security mechanisms based on service provider policies when creating and/or deleting a user account (e.g. disallow, require out-of-band confirmation procedures, require HTTPS, etc). As such, SNEW Clients MUST be prepared to receive responses informing that the request was accepted (e.g. 202 response code) but is not yet completed and typically expect some out-of-band procedures for completion.

Furthermore, when a user requests to delete his account, upon acceptance, SNEW Servers SHALL delete all related user information (including activities and reactions, etc) unless there are legitimate reasons to retain it.

SNEW Servers MAY also apply additional mechanisms to deactivate (i.e. no longer visible to other users) and/or delete user accounts after a defined period of time. If so, SNEW Servers SHOULD notify users in some way before taking these steps.

See Appendix J.3 for some specific user account creation procedures.

7.1.1.4 Search

SNEW Servers SHOULD allow to search for a user based on some attributes. When supported, SNEW Servers SHALL support search at least over users that they manage directly (i.e. “local” users) and at least “displayName” attribute of the Person object as search criteria.

Search requests are based on requests for retrieving a list of persons and SHALL further contain the parameters “filterBy”, “filterOp” and “filterValue” as defined in [OS-Core-API] section 6.2.

SNEW Servers MAY apply restrictions and/or additional security mechanisms to search requests based on service provider policies (e.g. disallow anonymous search requests, hide users that are not searchable, etc). In particular, SNEW Servers SHOULD NOT make users searchable by default, unless these users give explicit consent to do so.

7.1.1.5 Group membership

Joining and leaving groups are considered as creating or deleting relationships with that group, as defined in section 7.1.1.1. In particular, the group identifier SHALL be indicated as Group-Id as specified in [OS-Social-API] is requested for indicating the particular group of interest.

7.1.2 ActivityStreams Service support

The creation and update of an activity and/or a reaction in the form of a ActivityEntry object as REST-Request-Payload together with “application/json” MIME type MUST be supported by SNEW Clients and SNEW Servers as per [OS-Social-API] sections 2.3.2 and 2.3.3, with the clarifications provided in this section.

SNEW Clients & SNEW Servers SHALL comply to section 9.2.3 for defining privacy settings and target audiences of ActivityEntry objects, i.e. to control the visibility of activities and reactions by other users.

In case privacy information is omitted within an activity, the default setting of the activity’s target is applied, based on the service provider policy and/or user preferences.

It is RECOMMENDED that default privacy setting for activities and reactions is restricted only to the users, whom the actor has explicitly approved relationships. For example, depending on the service provider policy, “friends” may require explicit approval of relationship requests whilst “followers” may not, resulting in default activities being accessible to “friends” only.

As specified by [OS-Social-API] section 2.3.2, activity creation relates to metadata associated with an activity, without considering uploading media (or generically content) contextually.

As such, activities related to content upload (e.g. posting a video, attaching a picture to a place) can be easily provided in two steps as follows. At step one, content is uploaded by means out of scope of this specification that provides back a reference to the newly uploaded content as URI. At step two, an ActivityEntry object is subsequently published on the ActivityStreams Service containing the URI referencing the content as “url” parameter in the “object” of that ActivityEntry.

In addition, the method of content upload defined in section 7.1.3 SHALL be supported by SNEW Clients and SNEW Servers. This method allows uploading one or more content (e.g. media such as pictures, video, etc) contextually to an activity, thus allowing the creation of complex activities (e.g. including “targets”).

When updating an activity or a reaction (section 2.3.3 of [OS-Social-API]), the REST-URI-Fragment SHALL further contain an Activity-Id element in the form of an Object-id according to [OS-Core-API] (i.e. "/activitystreams/" User-Id "/@self/" Activity-Id) that references the specific activity entry to be updated.

See Appendix G.1.1 for examples related to the ActivityStreams Service.

7.1.3 Content upload

This method allows content to be uploaded contextually to a JSON object (e.g. ActivityEntry, Person), thus providing the maximum expressiveness in terms of information in the object itself. This method is defined as follows:

- The REST-HTTP-Method, REST-URI-Fragment and REST-Query-Parameters SHALL follow [OS-Social-API] as for the creation or update of objects using the related Service
- The REST-Request-Payload SHALL be encoded as multipart/related as defined by [RFC2387] and SHALL contain at least two body parts. as follows:
    - The JSON object SHALL be included into a body part with Content-Type header set to “application/json”. References to the content parts in the JSON object (e.g. in the “url” attribute in the “object” of an

ActivityEntry, or in the “thumbnailUrl” attribute of a Person) SHALL be in the form of a 'cid:' URI as defined by [RFC2392]

- Content being uploaded SHALL be included into different body part(s). Each of these parts SHALL contain a Content-Type header set to the MIME type of the content being inserted (e.g. “image/jpeg”) in that part, as well as a Content-ID header as per [RFC2392].
- The content type of the request SHALL be set to the “multipart/related” MIME type with “type” attribute set to “application/json”.

– The Return-Object SHALL contain an JSON object as specified in the section of the related Service in [OS-Social-API] corresponding to the newly created resource. This object SHOULD contain the reference URI(s) to the newly uploaded content in lieu of the cid: URI.
The Location header in the 201 response pointing to the URL of the newly created resource can be further used to perform subsequent requests using the related Service, e.g. for update or deletion.

An example of such method is shown in Appendix G.1.2.

7.1.4 Gateway functionality support

When the gateway functionality is supported, SNEW Server and SNEW Client SHALL support following functions:

– Get the list of supported External SNs;

– Connect an External SN account with user account and delete the connection;

7.1.4.1 Gateway endpoint definition

Following the syntax used by [OS-Spec] over SNEW-1, the URI structure of Gateway functionality MUST be:

```
Service-Name = "connections"

Connection-Id = Object-Id / "@all" / "@self"

Connection = see section 9.4.1
```

See Appendix G.1.3 for examples of External SN connection management.

7.1.4.2 Retrieving the list of External SNs connections

```
REST-HTTP-Method = "GET"

REST-URI-Fragment = "/connections/" User-Id "/" Connection-Id

REST-Request-Payload = null

Return-Object = Connection / Collection<Connection>
```

SNEW Client uses the GET method to retrieve information from SNEW Server about connections with External SNs, such as:

– Get the list of all the External SNs potentially supported by the SNEW Server using “@all” as Connection-Id;

– Get all the current connections of the specified user using “@self” as Connection-Id.

– Get a specific connection information by using a specific Connection-Id.

The response payload of successful responses SHOULD contain a connection or a collection of connection objects formatted according to section 9.4.1.

7.1.4.3 Deleting a connection towards an External SN

```
REST-HTTP-Method = "DELETE"
```

```
REST-URI-Fragment = "/connections/" User-Id "/" Connection-Id

REST-Request-Payload = null

Return-Object = null
```

SNEW Client uses the DELETE method to delete the user connections, such as:

- Delete the specific connection by specify the Connection-Id.
- Delete all the current connections for the specified user using “@self” as Connection-Id.

SNEW Servers MUST NOT accept requests for deletion using the “@all” value as Connection-Id.

7.1.4.4 Creating a connection towards an External SN

```
REST-HTTP-Method = "POST"

REST-URI-Fragment = "/connections/" User-Id "/@self"

REST-Request-Payload = Connection
```

SNEW Client uses the POST method to create a new user connection, i.e. to connect an External SN account with user account, including a connection object formatted according to section 9.4.1 that identifies the External SN to be connected (e.g. “domain”).

Upon reception of such request, the SNEW Server SHOULD activate a procedure to allow the user to connect to the specified External SN and create the associated connection. This procedure MAY trigger one or more redirect responses to the original request depending on the External SN. Such responses MUST be detected by the SNEW Client, which SHOULD open a device browser (embedded or default) to the provided Location URI.

Note that in a typical process (such as OAuth2) such URI corresponds to a page on the External SN web site to request user to authenticate and authorize the application to access his data. After the user has given authorization, the External SN notifies back the SNEW Server with an access token.

Once this procedure is concluded, the SNEW Server SHOULD issue a redirect response using 303 response code providing a Location URI in the following format, which MUST be detected by the SNEW Client.

```
"Location: " "snew://connections#" Fragment-Parameters

Fragment-Parameters = "status=" Status-Code ["&connectionId=" Connection-Id
"&domain=" Domain]

Status-Code = "OK" / "ERROR" / "CANCEL"/ "INSUFFICIENT_PERMISSIONS"

Domain = see section 9.4.1
```

If the procedure concluded successfully, the SNEW Server SHALL use “OK” as Status-Code and SHALL include the newly created “connectionId” together with the “domain” of the associated External SN

If the procedure was aborted by the user, the SNEW Server SHALL use the “CANCEL” Status-Code without providing any other information.

If the procedure concluded with the user refusing some required permissions, the SNEW Server SHALL use the “INSUFFICIENT_PERMISSIONS” Status-Code without providing any other information.

In any other case, the SNEW Server SHALL use the “ERROR” Status-Code without providing any other information.

Note that the protocol scheme “snew:” MUST be associated to the SNEW Client on the device to detect the response from browser.

7.1.4.5 Updating an existing connection towards an External SN

```
REST-HTTP-Method = "PUT"

REST-URI-Fragment = "/connections/" User-Id "/" Connection-Id

REST-Request-Payload = Connection
```

SNEW Client uses the PUT method to update an existing user connection, including a Connection object formatted according to section 9.4.1, which "org.oma" object SHALL include a "connectionStatus" property specifying the value of the newly intended status.

Specifically, the "active" value SHALL be used to refresh and/or reactivate an existing connection. In that case the rest of the procedure is identical to the one specified in section 7.1.4.4 for creating connections.

7.1.5 Subscription-mode support

SNEW Clients SHOULD indicate support for subscription-mode in retrieval requests (GET) over SNEW-1 targeting dynamic sets of information (e.g. activities & reactions) to avoid polling of resources, which may be rate-limited by SNEW Servers.

To indicate support for subscription-mode, SNEW Clients SHALL insert an additional Accept: header value "text/event-stream" in such requests.

When providing responses to retrieval requests (GET) over SNEW-1, SNEW Servers SHOULD further indicate support for subscribing to the related information.

To indicate support for subscription-mode in such responses, SNEW Servers:

- SHALL insert at least one Link Header [RFC5988] with rel=hub indicating the URL of the endpoint where to subscribe to this information (subscription endpoint),
- SHOULD insert one Link Header with rel=self referencing the permanent URL of the information to be used as "topic" when subscribing.

In particular the subscription endpoint SHALL act as hub according to [PubSubHubbub] section 6 with the following clarifications:

- SNEW Servers SHALL accept subscription requests containing no "hub.callback" parameter if such requests contain an "Accept" header with value "text/event-stream" and/or an "X-Oma-Push-Accept-Source" header as defined in Appendix H
- When issuing subscription responses, SNEW Servers MAY insert additional headers as defined in Appendix H to indicate the type of Push notifications they support for this subscription.

When indicated by an SNEW Server in the response, an SNEW Client SHOULD subscribe to the related information. In this case, SNEW Clients SHALL act as subscriber according to [PubSubHubbub] section 6 with the following clarifications:

- When creating a subscription request, SNEW Clients:
    - SHALL include a "hub.topic" parameter set to the value provided in the "self" Link header. If no such link was provided by the SNEW Server, the URL of the original retrieval requests SHALL be used.
    - SHALL include an "Accept" header with value "text/event-stream" and/or an "X-Oma-Push-Accept-Source" header as defined in Appendix H.

      Note that the inclusion of both headers may allow seamless switching of notification bearer
    - SHOULD NOT include a "hub.callback" parameter
- In case a Push API Server is locally available on the device according to [OMAPUSH-WRAPI], SNEW Clients SHALL issue the subscription request to the local Push API Server by including the subscription endpoint URL (containing the related "hub.*" parameters in URL-encoded format) as "push-accept-source" parameter.

Otherwise, SNEW Clients SHALL issue the subscription request directly to the endpoint indicated in the “hub” Link header.

– In absence of a local Push API Server, when receiving subscription responses that contain additional headers as defined in Appendix H, SNEW Clients SHALL prepare the related Push notification filters for this subscription. When a Push API Server is locally available on the device it is expected that a similar behaviour be performed by the Push API Server instead.

An example of such mechanism is shown in Appendix G.1.6.

7.1.6 Response codes and errors

Status codes in responses over SNEW-1 SHALL follow the rules and values defined in [RFC2616] section 6.1.1 with the further clarifications provided in [OS-Core-API] sections 2.1.2 and 5.6.

SNEW Servers and SNEW Clients SHALL further support the REST-JSON-Response-Payload defined in [OS-Core-API] section 2.1.2.1.Whilst Return-Objects are typically returned in case of successful responses, this definition allows to complement protocol-level errors (at HTTP level) with application-level errors that can be carried as response payloads for more detailed failures.

7.2 SNEW-2

SNEW-2 interface is exposed by SNEW Servers to SNEW-Enabled Applications and consists of three main functionalities described in the following subsections.

In particular over SNEW-2, SNEW Servers:

– SHALL expose an endpoint to share links from external websites according to section 7.2.1
– SHALL expose a Network API according to section 7.2.2
– SHOULD allow users to export and/or import their personal information according to section 7.2.3

7.2.1 Link sharing from external websites

In order to share links when browsing external websites (e.g. through a “share” button), SNEW Servers SHALL support the OExchange specification [OExchange-Spec] over SNEW-2.

In particular, SNEW Servers SHALL expose at least one OExchange Target endpoint, to be referenced in the related descriptors with the clarification provided in section 8.2.1.

7.2.2 Network API

SNEW Servers SHALL expose a Network API over SNEW-2 interface to access at least user activities and reactions.

In particular, this Network API over SNEW-2 SNEW Servers SHALL conform to section 7.1 with the following clarifications:

– With reference to the ActivityStreams Service, SNEW Servers SHALL support retrieving activity entries and SHOULD support the creation, deletion, and update of activity entries.

– SNEW Servers SHOULD further support the subscription mechanisms defined in section 7.1.5 over SNEW-2 Network API with the exception that they SHALL NOT accept subscription requests containing no “hub.callback” parameter over SNEW-2 interface.

– SNEW Servers MAY NOT expose other functionalities of SNEW-1 over SNEW-2 Network API.

7.2.3 Social data portability

When allowing users to export and/or import their personal information, SNEW Servers SHALL comply to this section and to the procedures defined in section 8.2.8.

This mechanism allows users to import and/or export the social information stored on an SNEW Server (e.g. history of activities, reactions, profile, contacts, multimedia content). This may be used for backup purposes or for migration from a social network to another.

The following packaging format is defined for importing and/or exporting user-related social data:

- The overall user-related data SHALL be packaged as valid Zip archive file format, according to [ZIP] specification
- The package SHALL contain:
  - One social data package descriptor file, located at the root of the package, as specified in section 7.2.3.1.
  - Zero or more arbitrary files located either at the root of the package and/or in arbitrary folders or in locale folders.
- User activities & reactions SHALL be represented into arbitrary files as ActivityEntry JSON objects as per section 9.2.
- User profile and contact information SHALL be represented into arbitrary files as Person JSON objects as per section 9.3.
- References to other files in the package (e.g. media) within JSON objects SHALL rely on the "file:" URI

7.2.3.1 Social data package descriptor

A social data package descriptor is an [XRD] document. Such descriptor SHALL be located at "xrd.xml" file location at the root of the package.

The social data package descriptor MAY be signed according to [XRD] section 5 to guarantee the authenticity and integrity of the document.

The social data package descriptor MUST include a Property element with type http://www.openmobilealliance.org/snew/package/prop/version. This element SHALL assume value "1.0" in this version of the specification. This version identifier should simplify the import task in understanding the exact data formats referenced.

The social data package descriptor SHALL include a Subject element containing the globally unique identifier of the related user, as specified in section 9.1.1.1.

The following link relation types are identified to include references to specific types of user-related information. href attributes of the link elements SHOULD be in the form of "file:" URIs and MAY be in the form of "data:" URIs (e.g. for icons, etc). A social data package descriptor can contain any combination of such links.

Link relation type	*Description*
`http://ns.opensocial.org/2008/opensocial/<OS-URI-Fragment>`	Reference to the related information and data format based on the specific OpenSocial Service, where: `<OS-URI-Fragment> = Service-Name "/@me/" <Group-Id>` (see [OS-Social-API] for Service-Name and Group-Id)
`http://webfinger.net/rel/avatar`	Reference to the user avatar URL

`describedby`	Reference to a description of a user profile. The type attribute indicates the content type of this information (default is “text/html”).
`http://schemas.google.com/g/2010#updates-from`	Reference to the Atom representation of the user’s activity stream

An example of such descriptor is provided in Appendix D.4.

7.3 SNEW-3

To realize federation across SNs, SNEW-3 interface is both exposed and used by SNEW Servers. SNEW-3 further leverages on the same principles and protocols as the ones defined in [OStatus] (see section 7.3.5) as described below.

In particular SNEW Servers SHALL

- expose a .well-known/host-meta descriptor and a user descriptor with the clarifications provided in section 8.2.1.
- handle incoming requests from other SNEW Servers according to section 8.2.5

When issuing requests to other SNEW Servers over SNEW-3 to access remote user data, SNEW Servers SHALL follow the procedures of section 8.2.2 to discover remote user descriptors unless previously cached, and SHALL further:

- follow the procedures of section 8.2.3 to access remote user activities and reactions
- follow the procedures of section 8.2.4 to access remote user information

SNEW Servers SHALL further support the protocols defined in the following sections 7.3.1 to 7.3.4 over SNEW-3.

7.3.1 Salmon support

SNEW Servers SHALL support [Salmon-Protocol] to:

- receive Salmon notifications, by exposing at least one Salmon endpoint to be referenced in the user descriptor according to section 8.2.1.
- send Salmon notifications according to sections 8.2.5 and 8.2.6.

7.3.2 PubSubHubbub support

SNEW Servers SHALL support [PubSubHubbub] to:

- receive notifications from publishers through a PubSubHubbub hub (see section 8.2.5), by exposing at least one PubSubHubbub callback endpoint
- notify subscribers of new updates through a PubSubHubbub hub (see section 8.2.6)

If multiple subscribers under the same SNEW Server subscribe to the same publisher, the SNEW Server SHOULD use the one PubSubHubbub subscription for all of them.

SNEW Servers MAY additionally act as PubSubHubbub hub.

7.3.3 ActivityStreams Atom support

SNEW Servers SHALL expose at least one endpoint according to [AS-ATOM] to be referenced in the user descriptor according to section 8.2.1. This endpoint is used to provide access to user’s activity stream.

In particular, all activity streams exposed by this endpoint:

- SHALL contain a hub relation link to identify a PubSubHubbub hub to establish subscriptions.
- SHALL represent activity entries according to section 9.2
- SHOULD contain a salmon relation link to identify the Salmon endpoint where to receive replies and notifications

7.3.4 Portable Contacts support

SNEW Servers SHALL expose at least one endpoint according to [POCO] to be referenced in the user descriptor according to section 8.2.1.

In particular, the People Service provided by SNEW Server over SNEW-1 (see section 7.1) is a PoCo-compliant endpoint and may be exposed as such over SNEW-3.

7.3.5 Compatibility with OStatus (Informative)

SNEW-3 interface is intended to be wire-compatible with the OStatus 1.0 specification [OStatus-Spec]. Specifically, SNEW Servers are aimed to be both OStatus-compliant "subscriber server" and "publisher server" at the same time.

Although not referencing the OStatus specifications normatively, the SNeW Enabler is wire-aligned with these specifications in order to maximize widespread adoption of a common framework for Social Network federation. It is the intention of the SNeW Enabler to maintain this compatibility in future versions as long as the implications remain compatible with the scope and approach of this enabler.

7.4 SNEW-4

The interface definition of SNEW-4 is postponed to a future release and is not in the scope of SNeW 1.0.

7.5 SNEW-5

The SNEW Server SHALL support either the Push Access Protocol (PAP) [OMAPUSH-PAP] (with the RESTful Network API for OMA Push [OMAPUSH-REST] as an optional PAP binding) or the Push-OTA protocol [OMAPUSH] for point-to-point delivery over the SNEW-5 interface.

SNEW Clients SHALL support direct and indirect Push delivery of SNeW messages over SNEW-5 interface.

Unless otherwise specified, a SNeW message is a Push Content that represents SNeW a notification/information in the context of this enabler related to e.g. persons, user activities, reactions and as such SHALL be represented in JSON format according to [OS-Social-Data].

Whether to use PAP or Push-OTA is a deployment choice. Among other advantages, use of PAP prevents the SNEW Server from needing to implement most of the basic functions of OMA Push Proxy Gateway (PPG), e.g. the various Push-OTA protocols and target client context/capabilities awareness (as needed to select the appropriate transport bearer and protocol). Conversely, for high-volume search services limited to Push delivery over a single specific bearer (e.g. WAP1 Push over SMS), use of Push-OTA is fairly simple and avoids dependency upon a Push Proxy Gateway (PPG).

SNeW messages may not be directly deliverable over connectionless bearers such as WAP Push over SMS (limited to about 512 bytes) or SIP Push via the SIP MESSAGE method (limited to about 1300 bytes), as compared to WAP Push over HTTP or SIP Push via the INVITE/MSRP method (both of which support essentially unlimited content size). The alternative to direct delivery is "indirect delivery", which involves the delivery of a Push notification message carrying a content location URL, from which the client retrieves the response.

When using PAP for direct SNeW messages delivery, the SNEW Server SHALL submit the SNeW messages using the MIME content type "application/json", and SHALL support at least the "URI" target client address scheme defined in [OMAPUSH-PPG] to identify users according to section 9.1.1.1. Additional target address schemes (e.g. PLMN, USER) MAY further be supported.

When directly delivering SNeW messages via Push-OTA, the SNEW Server SHALL send the SNeW message encapsulated into a "message/vnd.oma.push" media type as described in [OMAPUSH-MSG].

To deliver SNEW-5 messages indirectly, the SNEW Server SHALL use the ServiceLoading (SL) [OMAPUSH-SL] content type, via either PAP or Push-OTA as applicable, and include a URL from which the SNEW Client can retrieve the actual SNeW message. SNEW Clients SHALL support indirect delivery of SNeW messages, triggered by reception of ServiceLoading notifications.

Note that the "slc" MIME type for the Push SL events would be delivered as "sl" (i.e. in plain format) with headers, and the "SNeW messages" in their unwrapped form.

Push Clients in SNeW supporting terminals SHALL support routing of Push messages with the Push Application ID header "X-Wap-Application-Id: x-oma-application:snew.ua" to the SNEW Client. If there is no Push Client in the device, the SNEW Client SHALL implement the necessary Push Client functions for the supported Push-OTA protocol variants per [OMAPUSH].

7.5.1 SNEW-5 Interface Bindings

The table below gives an overview of how SNEW-5 messages are bound to the OMA Push-OTA protocol.

Message	SNEW Client ←→ SNEW Server	HTTP Method	Push-OTA Binding	PAP Binding
SNEW-5: Information	←		Push Message (Information)	Push Message (Information)
OR				
SNEW-5: Information Notification	←		ServiceLoading (Information URL)	Push Message (ServiceLoading (Information URL))
	→	HTTP GET		
SNEW-5: Information	←	HTTP Response (including 200 OK of the underlying method)		

Table 1: SNEW-5 interface bindings to OMA Push

7.5.2 SIP Push OTA binding

In addition to the general SIP Push requirements of [OMAPUSH], the additional procedures defined below apply for SNeW:

- at Push Servers (i.e. either the SNEW Server acting as Push Server, or the PPG via which the OMA Push enabler is leveraged for SNeW);
- at Push Clients (i.e. either the SNEW Client acting as Push Client, or the Push Client via which the OMA Push enabler is leveraged for SNeW).

The Push Resource Identifier for SNeW SHALL be "snew.ua". Use of this value applies in all procedures which involve setting or processing of Push Resource Identifiers.

8. Functional Component Procedures

8.1 Procedures at SNEW Client

Procedures at SNEW Client heavily rely on a “local cache” and a “pending request queue” for handling requests and responses in an optimized manner, in particular to limit interactions over the air.

The “local cache” is used to allow internal sharing of information between SNEW Enabled Device Applications when applicable, and can be considered as the latest up-to-date information on the device received from SNEW Server and/or sent by SNEW Client.

The “pending request queue” is used to store “pending” requests (e.g. to post an activity, retrieve a user’s stream, etc) before their actual transfer from SNEW Client to SNEW Server, e.g. for throttling purposes or due to the absence of connectivity. As such, the information contained in the pending request queue is typically different than the one in the local cache.

8.1.1 Issuing requests to SNEW Server

In absence of connectivity and/or based on configuration and policies (e.g. throttling mechanisms), the SNEW Client SHALL store the pending requests in a local queue until normal conditions are resumed, to avoid discarding of information.

As soon as normal conditions are resumed, the SNEW Client SHOULD issue the queued requests in chronological order as this could influence the user experience. When applicable, the SNEW Client may optimize this queue to avoid conflict and/or duplication of requests (e.g. retrieve twice the same user’s stream of activities and reactions).

When issuing a request on SNEW-1 towards SNEW Server, the SNEW Client SHALL support authentication & authorization according to the procedures defined in section 10. The SNEW Client MAY further specify cache control directives and/or cache validators in the request.

Requests on SNEW-1 SHALL follow the rules defined in section 7.1.

8.1.2 Handling responses & notifications from SNEW Server

When receiving responses from SNEW Server that correspond to requests for information over SNEW-1, or when receiving notifications from SNEW Servers over SNEW-5, the SNEW Client SHALL update the local cache for each information received according to the following procedures, subject to the mechanisms defined in [RFC2616] section 13:

- when the received information refers to an information (e.g. activity, person) not yet in the local cache, insert it into the local cache;
- when the received information refers to an information already existing in the local cache, updated it in the local cache with the newly received information;
- when the received information refers to the deletion of an information already existing in the local cache, delete it from the local cache.

The SNEW Client SHALL further forward the retrieved information to the requesting application(s) on the device. Depending on the conditions when the related requests were sent (e.g. requests were queued due to absence of connectivity), a single response may correspond to requests from multiple applications.

8.1.3 Handling requests from SNEW Enabled Device Applications

Upon reception of a request from SNEW Enabled Device Applications on SNEW-4, the SNEW Client SHALL follow the following procedures.

If the request aims at retrieving information (e.g. retrieve the stream of a user’s activities and/or reactions), the SNEW Client SHALL first check the information cached locally for this information. Depending on policies relevant for that specific application (e.g. requested by the application itself or enforced by service provider policies), if the requested information is

not present in the cache, or if such information does not match the relevant policies (e.g. deprecated information), the SNEW Client SHALL issue the associated request to the SNEW Server according to the procedures defined in 8.1.1 to retrieve that information. Otherwise, if the requested information is already present in the cache and considered valid, the SNEW Client SHALL NOT issue the associated request to the SNEW Server and SHALL return the related information cached locally. This procedure optimizes the transfer of information over the air, whilst allowing to provide valid information to SNEW Enabled Device Applications, also in absence of connectivity.

If the request aims at providing information, either in the form of a creation or an update (e.g. posting an activity), the SNEW Client SHALL issue the request to the SNEW Server according to the procedures defined in 8.1.1 to create or update that information, and in addition, SHALL insert the related information in the local cache, or update it if already present, subject to a successful response to the operation being received from the SNEW Server. The SNEW Client MAY specify a duration of validity of this information in the local cache.

If the request aims at deleting information (e.g. deleting a picture previously posted), the SNEW Client SHALL issue the request to the SNEW Server according to the procedures defined in 8.1.1 to delete that information, and in addition, SHALL delete the related information from the local cache, if present, subject to a successful response to the operation being received from the SNEW Server.

8.2 Procedures at SNEW Server

8.2.1 Exposing descriptors

SNEW-2 and SNEW-3 interfaces require the exposure of several descriptors by SNEW Servers.

In particular, the SNEW Server SHALL:

1. Provide a .well-known/host-meta descriptor according to [HOST-META] at its root domain address, both over HTTP and HTTPS, containing at least:
    - one http://oexchange.org/spec/0.8/rel/resident-target relation link to a OExchange Target XRD descriptor for Target discovery (see bullet 3)
    - one oauth2-authorize and one oauth2-token relation links as defined in [OAUTHLRDD] for the discovery of OAuth2 endpoints exposed according to section10.2.
    - one http://ns.opensocial.org/2008/opensocial relation link to the base URI common to all OpenSocial service endpoints exposed over SNEW-1 and SNEW-2 interfaces. This base URI includes the root domain URI and the REST-Base-Path, whilst the REST-URI-Fragment will vary based on the specific service as per [OS-Social-API].

    In addition, this descriptor SHOULD contain:

    - one lrdd relation link template to a XRD descriptor for user profile discovery (see bullet 2)
    - one relation link type icon for graphical representation of host icon image, and the value of the link is a image file URL (see D.1) . Suggested image file formats are icon, png, or jpeg.
2. Provide a Webfinger descriptor according to [WebFingerProtocol] descriptor that describes user accounts containing at least
    - for access to user's activity stream
        - one http://schemas.google.com/g/2010#updates-from relation link of type "application/atom+xml" that contains the URL of the related SNEW Server's endpoint
    - for reactions exchange
        - one salmon relation link according to [Salmon-Protocol] that contains the URL of the SNEW Server's endpoint that receives reactions related to that user (e.g. comments, replies, mentions, etc)
        - one http://salmon-protocol.org/ns/magic-key property according to [MagicSig] that contains the user's magic key

- for access to profile and contact information
  - one http://portablecontacts.net/spec/1.0 relation link according to [POCO] that contains the Base URL of the Portable Contacts API endpoint in order to access user profile information

    Note: [POCO] support is natively provided by the People Service of the OpenSocial Social API Server Specification according to [OS-Social-API] section 3.
  - one http://ns.opensocial.org/2008/opensocial/people relation link that contains the URL of the OpenSocial People Service endpoint related to that specific user account in order to access user contact information

In addition, this descriptor MAY contain other information such as:

- one http://webfinger.net/rel/avatar that contains the URL of a thumbnail picture representing the user's avatar.
- one http://webfinger.net/rel/profile-page link of type "text/html" that contains the URL of the user profile web page
- one describedby link of type "application/rdf+xml" that contains the URL of the user profile description according to [FOAF]

Furthermore, the SNEW Server SHOULD provide an equivalent descriptor in [XRD] format over HTTPS, to be referenced as lrdd link relation type in the host-meta descriptor in order to be accessible by OStatus-compliant SNs.

3. Provide a OExchange Target XRD descriptor according to [OExchange-Spec]

Examples of such descriptors are provided in Appendix D.

8.2.2 Discovering user descriptor at remote server

SNEW-3 interface relies on the discovery of user information across SNs.

This discovery process can be activated upon reception of any SNEW-1 request (from an SNEW Client) asking to interact with a user belonging to another SNEW-compliant SN. In particular, this process SHALL be triggered when receiving requests to follow remote users or when receiving reactions (e.g. comments, replies, mentions) that relate to activities of remote users, unless the equivalent data is already locally available to the SNEW Server.

The goal of this process is to retrieve a descriptor describing how to find a user's public metadata from his/her user identifier.

In particular, the SNEW Server SHALL perform a Webfinger query according to [WebFingerProtocol]. SNEW Servers MAY cache the information provided in retrieved descriptors and/or issue conditional requests at a later stage to check for updates of this information, subject to the mechanisms defined in [RFC 2616] section 13.

To maintain compatibility with OStatus-compliant SNs in case the WebFinger endpoint does not exist at the remote server, SNEW Servers MAY perform an alternative discovery process as described in Appendix K.

After this process, the SNEW Server MAY read additional content from the retrieved descriptor depending on the action requested by the SNEW Client.

8.2.3 Retrieving user activities and/or reactions at remote server

In order to retrieve the activities and/or reactions of a remote user over SNEW-3 interface, the SNEW Server SHALL first perform user descriptor discovery according to section 8.2.2, unless previously cached.

Once retrieved and validated the user descriptor, the SNEW Server SHALL read the http://schemas.google.com/g/2010#updates-from relation link of type "application/atom+xml" representing the URL of the user's activity stream feed.

If the SNEW Server is not yet subscribed to this feed:

1. It SHALL send a request to that URL and validate the retrieved feed as valid ActivityStreams Atom feed according to [AS-ATOM] and [ATOM] with the clarifications provided in section 9.2.

2. If the feed contains a hub relation link, the SNEW Server MAY subscribe to that feed according to section 7.3.2 and [PubSubHubbub] section 6

The end to end example flow of discovering user descriptor and subscribing to the user's activity stream feed is given in Appendix I.1.

8.2.4 Retrieving user information at remote server

In order to retrieve the user information of a remote user over SNEW-3 interface, the SNEW Server SHALL first perform user descriptor discovery according to section 8.2.2, unless previously cached.

Once retrieved and validated the user descriptor, the SNEW Server SHALL read the http://ns.opensocial.org/2008/opensocial/people relation link representing the URL of the user's OpenSocial People Service endpoint. If such relation link is not present in the descriptor, the SNEW Server MAY read the http://portablecontacts.net/spec/1.0 relation link as fallback to obtain the same URL.

The SNEW Server SHALL send a request to that endpoint based on the required operation (e.g. retrieve a user profile information, retrieve user relationships, etc).

In particular, if this process is triggered by an incoming request over SNEW-1, the SNEW Server SHALL consider that endpoint as concatenation of the REST-Base-Path and the Service-Name ("/people") and replicate the remaining part of the REST-URI-Fragment. For example, if the incoming request is targeting "http://example.com/snew/v1/people/acct:bob@test.org/@self" and http://ns.opensocial.org/2008/opensocial/people relation link for bob@test.org is "http://test.org/api/os/people", then the corresponding outgoing request will be sent to "http://test.org/api/os/people/acct:bob@test.org/@self".

If the requested information cannot be retrieved, the SNEW Server MAY generate a Person object locally according to section 9.3 inserting the target user identifier as "id" property. Other properties, such as "displayName " or "thumbnailUrl", to be included in this object may be generated locally and/or derived from other information available from user descriptor (e.g. the http://webfinger.net/rel/avatar link relation if available). This behaviour may be particularly useful when the user descriptor provides a salmon link relation, which *de facto* indicates that the user exists and can be followed, although a direct access to user information is not available.

8.2.5 Handling requests from remote SNEW Servers

Depending on the semantics of the request received on SNEW-3, the SNEW Server may act differently, e.g. by performing local processing (e.g. updating its internal database) and/or by issuing one or more requests on SNEW-2, SNEW-3 or SNEW-5 interfaces (e.g. an SNEW Enabled application or another SNEW Client are subscribed to receive notifications when a specific user updates her status).

The following table illustrates a mapping between incoming SNEW-3 requests and outgoing requests.

Request semantics	SNEW-3 request	Outgoing interface	Outgoing request(s)	Notes
Notification about remote user reactions	Salmon callback	SNEW-3	Salmon notification(s)	Only applicable to propagate reactions towards remote users

		SNEW-5	See section 8.2.7	Applicable towards subscribed SNEW Clients and/or applications
Notification about remote user activity	PubSubHubbub callback	SNEW-5	See section 8.2.7	Applicable towards subscribed SNEW Clients and/or applications

In case the request needs to be forwarded to one or more recipients, the SNEW Server SHALL follow the address resolution procedures defined in section 9.1.1.2.

As per [POCO] specifications, support for incoming requests to the OpenSocial People Service on SNEW-3 (indicated by http://ns.opensocial.org/2008/opensocial/people rel type in the SNEW Server's descriptor according to section 8.2.1) MAY be restricted to GET requests only.

The end to end example flow of cross domain activities dispatching is given in Appendix I.2, and the example flow of cross domain reactions in Appendix I.3.

8.2.6 Handling requests from SNEW Clients and/or SNEW Enabled Applications

The SNEW Server SHALL authenticate the user according to one of the procedures defined in section 10.1 upon reception of a request:

- from SNEW Client over SNEW-1,
- from SNEW Enabled Application over SNEW-2 Network API,
- from SNEW Enabled Application (e.g. web application) over SNEW-2 OExchange endpoint.

Depending on the semantics of the request received, the SNEW Server may act differently, e.g. by performing local processing (e.g. updating its internal database) and/or by issuing one or more requests on, SNEW-3 or SNEW-5 interfaces (e.g. an SNEW Client is subscribed to receive notifications when a specific user updates her status).

Furthermore, in case the Gateway functionality is supported by the SNEW Server, the request received may trigger additional requests towards External SNs according to their own mechanisms that are out of scope of this specification. The issuing of such requests may further depend on the information related to the user's connections with External SNs (e.g. access token and account information on the associated External SN) available at the SNEW Server and may be aimed at either retrieving (inbound) or posting (outbound) information. See section 7.1.4 for more information about the Gateway functionality.

The following table illustrates a mapping between incoming requests and outgoing requests.

Request semantics	*Incoming interface*	*Incoming request(s)*	*Outgoing interface*	*Outgoing request(s)*	*Notes*
Retrieving (or subscribing) a user's feed	SNEW-1 / SNEW-2 Network API	OpenSocial ActivityStreams Service / GET	SNEW-3	See section 8.2.3	Only applicable if target user is remote
		Subscription endpoint (see section 7.1.5)	External SN	Out of scope	Only applicable if the Gateway functionality is supported

Publishing an activity or reaction	SNEW-1 / SNEW-2 Network API SNEW-2 OExchange endpoint	OpenSocial ActivityStreams Service / POST OExchange Offer / GET	SNEW-3	PubSubHubbub notification	
				Salmon notification(s)	Only applicable for reactions towards remote users
			SNEW-5	See section 8.2.7	Applicable towards subscribed SNEW Clients and/or applications
			External SN	Out of scope	Only applicable if the Gateway functionality is supported
Retrieving (or subscribing) a user's information	SNEW-1	OpenSocial People Service / GET Subscription endpoint (see section 7.1.5)	SNEW-3	See section 8.2.4	Only applicable if target user is remote
			External SN	Out of scope	Only applicable if the Gateway functionality is supported

In case the request needs to be forwarded to one or more recipients, the SNEW Server SHALL follow the address resolution procedures defined in section 9.1.1.2.

Furthermore, when sending a request to an SNEW Client or to another SNEW Server that contains a reference to the originator's address, the SNEW Server SHALL insert the asserted user identity in the form of a global identifier according to section 9.1.1.1 into the corresponding field in the request. As such, it MAY override the originator's address provided by the originator SNEW Client.

Note that the SNEW Server may need to transcode information between incoming and outgoing interfaces. For example, incoming requests over SNEW-1 to retrieve activities of a remote user need the related response to be transcoded from Atom (received over SNEW-3 after forwarding the request) into Json format (sent back over SNEW-1).

When issuing the corresponding response, the SNEW Server MAY specify cache control directives and/or cache validators to control expiration times and/or conditional requests by clients as per [RFC2616] section 13, and MAY further indicate support for subscription-mode according to section 7.1.5.

Appendix I illustrates example flows of some federation functions across SNs (e.g. cross domain following, activities & reactions dispatching) triggered by SNEW Client over SNEW-1 and propagated over SNEW-3 interface.

8.2.7 Sending notifications to SNEW Clients

The SNEW Server use SNEW-5 interface to deliver asynchronous notifications/information messages to SNEW Clients related to e.g. content, user activities, reactions.

This notification process can be activated towards one or more users upon reception of any SNEW-1 request (from an SNEW Client) or SNEW-3 request (from another SNEW Server). In particular, this process MAY be triggered to notify local users when receiving reactions (e.g. comments, replies, mentions) that relate to activities of local users, or when receiving activities performed by other users that are followed by local users.

When the information to be notified corresponds to an activity or a reaction received from a local user over SNEW-1, or from a remote user on SNEW-3, the SNEW Server SHALL perform the following steps:

- The SNEW Server SHALL convert the information in JSON format according to [OS-Social-Data] unless already formatted as such.
- The SNEW Server SHALL include the Push Application ID header “X-Wap-Application-Id: x-oma-application:snew.ua”.
- The SNEW Server SHALL forward the information as Push Content to the SNEW Client over SNEW-5 according to section 7.5.

8.2.8 Importing a social data package

When importing a social data package (defined in section 7.2.3), SNEW Server SHALL decompress (and/or extract) the social data package descriptor located at “xrd.xml” file location at the root of the package using [ZIP].

Subsequently, the SNEW Server MUST verify the social data package descriptor XRD signature, if present, according to [XRD] section 5.

In case the social data package descriptor is verified, the SNEW Server SHOULD check the http://www.openmobilealliance.org/snew/package/prop/version property element to ease understanding of the data formats used in the package.

If the package version is supported, the SNEW Server SHOULD import the related user data.

9. Data Model

9.1 Identity

9.1.1 User identity

9.1.1.1 Addressing formats

SNeW users are identified through a globally unique identifier. Such identifiers MUST be expressed in the form of a URI when used over SNeW interfaces.

In particular, SNEW Clients and SNEW Servers MUST support the following URI schemes for user addressing:

- "tel:" URI scheme as defined in [TELURI] to represent E164 phone numbers
- "acct:" URI scheme as defined in [ACCTURI].

Note that the userpart of "acct:" URIs SHALL NOT refer to mobile number addresses. In case userpart is in the form of a numeric string, it MUST NOT be understood as a phone number. To this purpose, usage of the "tel:" URI scheme is RECOMMENDED.

Use of other URI schemes (address spaces) MAY be used to interoperate with other systems, but their use and definition is out of the scope of this specification.

Within a single mobile social network service, users MAY identify themselves using only local identifiers (e.g. username). In addition, local identifiers MAY be used to identify a user within the same mobile social network service.

9.1.1.2 Address resolution

For each recipient that appears in a request, the SNEW Server SHALL be able to resolve whether it is authoritative for the recipient (i.e. its phone number or domain is "locally" managed by this SNEW Server) or whether it pertains to another remote SN domain/provider and, if applicable, identify the recipient's SNEW Server.

If a recipient's address corresponds to a "acct:" address, the SNEW Server SHALL check whether the domain-part corresponds to a local domain or a remote domain. If so, the SNEW Server can locate the recipient's SNEW Server based on this domainpart and trigger the discovery process as defined in section 8.2.2.

If a recipient's address corresponds to a "tel:" address, the SNEW Server SHOULD use the DNS-ENUM protocol to identify the recipient's SNEW Server as defined in [RFC6116] and [ENUM-ACCT] (see Appendix F). In the absence of an ENUM-based solution, SNeW service providers MAY use solutions out of scope of this specification, which may include static tables or other lookup methods (e.g. MSISDN-to-IMSI lookup, composition of remote server URL based on recipient's home mobile network operator information).

This "tel:" address resolution process may discover associated recipient's addresses of any type, either belonging to a local SN domain or to a remote SN domain, over which the resolution process may be recursively performed as defined in this section.

If a recipient's address corresponds to another type of address, the SNEW Server MAY use mechanisms out of scope of this specification to resolve this address and identify the recipient's SNEW Server.

9.2 Activity and reaction

The format of an activity and a reaction follows the Activity construct as defined in [ActivityStreams]. Activities represent the first party actions (e.g. posting a content, joining a group, check-in a place), and reactions represent the actions toward the first party activities (e.g. commenting/replying someone else's post, sharing/retweet the activity, like).

Activities and reactions are serialized as follows:

- according to the Activity Streams Service described in [OS-Social-API] section 2.3, activities and reactions SHALL be represented as [AS-JSON] on SNEW-1 interface and on SNEW-2 Network API
- activities and reactions SHALL be represented as [AS-ATOM] on SNEW-3 interface

When the gateway functionality is supported, SNEW Clients & SNEW Servers SHALL comply to section 9.4.2 for representing External SN information in ActivityEntry objects.

9.2.1 Activity verbs and object types

The “activity” object type and the “post” verb SHALL be supported by SNEW Clients & SNEW Servers as per [AS-ATOM] & [AS-JSON]. The activity verbs & object types defined in section 9.2.4 SHALL also be supported.

In addition, the activity verbs & object types defined in [AS-BASE-SCHEMA] SHALL be considered with their original meaning and SHOULD be supported by SNEW Clients and SNEW Servers. Specifically, the following subset SHALL be supported by SNEW Clients & SNEW Servers:

- Verbs: checkin, follow, like, share, stop-following, unlike
- Object types: application, comment, image, note, person, place, video, bookmark

Other activity verbs & object types MAY be supported.

SNEW Clients & SNEW Servers SHOULD further support the base extension properties (e.g. location, tags, etc) defined in [AS-BASE-SCHEMA].

9.2.2 Reactions

Reactions on existing activities SHALL be represented respectively according to [RFC4685] on SNEW-3 and to [AS-JSON-REPLIES] on SNEW-1 interface and on SNEW-2 Network API.

9.2.3 Targeted audience and privacy

The extensions defined in [AS-JSON-AUDIENCE] SHALL be supported by SNEW Clients & SNEW Servers. Such extensions allow users to define the privacy level of their activities and/or reactions to control visibility by other users.

In particular, when referring to “alias”, SNEW Clients & SNEW Servers SHOULD use fully-qualified IRI (e.g. http://example.com/snew/v1/people/tel:+393351234567/@friends) in lieu of simple alias values (e.g. @friends) to avoid ambiguity.

Additional clarifications for usage in this specification with respect to [AS-JSON-AUDIENCE] section 3.1 are provided below:

- The use of “to” and/or “bto” property values only SHALL have the meaning of a restricted activity, thus only be accessible to the recipients explicitly mentioned either in the form of an “id” or an “alias”. In particular, the use of the @me simple alias value as unique value in “to” or “bto” SHALL have the meaning of a private activity not to be disclosed to other users.
- The use of “cc” and/or “bcc” property values (also in conjunction with other properties) SHALL have the meaning of explicit mentions of users, in addition to the default audience of that activity.

9.2.4 SNeW-specific verbs & object types

The following list represents verbs and object types defined within SNeW to be used in activities and/or reactions. These concepts are serialized over Atom and Json according to [AS-ATOM] and [AS-JSON] respectively.

In particular, when serializing object types and verbs as absolute IRI the following base IRI SHALL be used: “http://openmobilealliance.org/snew/schema/1.0/”.

9.2.4.1 Abuse object type

The "abuse" object type represents a primarily prose-based report on an abuse or violation by another object. The definition of abuse as object type allows activities and/or reactions to potentially associate abuses to any other type of object typically as activity target.

This object type SHALL be supported if the abuse reporting functionality is supported (see section 7.1).

The following properties MAY be used when describing an abuse object:

Property	Value	Description
`author`	Activity Object	Activity Object, typically using the "person" object type, responsible for the reporting of the abuse. If the abuse object appears within an activity or reaction, and the `author` is the `actor` for that activity, the `author` property MAY be omitted
`summary`	JSON [RFC4627] String	A description of the reason in the form of a fragment of HTML. Publishers SHOULD include any markup necessary to achieve a similar presentation to that on the publisher's own HTML pages, including any links that the service automatically adds. Processors MAY remove all HTML markup and consider the content to be plain text.
`title`	JSON [RFC4627] String	The title of the abuse report described in a sentence.
`id`	JSON [RFC4627] String	The unique identifier for the abuse object.
`published`	JSON [RFC4627] String	The optional time the abuse object was created in the form of an [RFC3339] "date-time".
`updated`	JSON [RFC4627] String	The optional time the abuse object was last updated in the form of an [RFC3339] "date-time".
`url`	JSON [RFC4627] String	The permanent IRI of the abuse's HTML representation.

9.2.5 Other activity-related information

Location information, tags, and other additional information MAY be included in activities and reactions as long as they do not break compatibility with [ActivityStreams].

In particular, format of location information and tags SHALL follow the formats defined in [AS-BASE-SCHEMA] when serialized over Json, and respectively the [GEORSS-SIMPLE] “point” format and the [ATOM] “category” format when serialized over Atom.

9.3 Person

According to [OS-Social-API] and [PoCo], when providing person details such as personal & contact information, Person objects provided by SNEW Clients and SNEW Servers SHALL include at least “id” and “displayName” attributes for each person, and SHOULD include “thumbnailUrl”.

In addition, as per Service Provider policy and/or upon explicit request by using the "fields" request parameter over SNEW-1 (see section 7.1.1), Person objects MAY include other information defined in [OS-Social-Data] such as "profileUrl", "status" (to indicate the user's latest activity and/or reaction), etc.

Furthermore, attributes related to sensitive personal data (e.g. "ethnicity", "politicalViews", "religion", "sexualOrientation") SHALL NOT be included in Person objects unless explicit previous consent of the related person.

9.3.1 Relationship indication

The "relationships" attribute defined in [OS-Social-Data] is used within a Person object to indicate the type of relationship existing (if applicable) between the requesting user and the target user described by the Person object. In particular, this specification defines the following reserved values for this attribute:

- "following" when the target user is following the requesting user
- "followers" when the target user is a follower of the requesting user
- "friends" when a mutual relationship has been established between both users

9.4 External SN

When the gateway functionality is supported, SNEW Clients & SNEW Servers SHALL support the data models defined in this section.

9.4.1 Connection

SNEW Clients & SNEW Servers SHALL represent information related to a connection to an External SN as an Account object as defined in [OS-Social-Data] section 3.1 with the following clarifications:

- The Account object SHALL contain at least the "domain" property
- The Account object SHOULD contain an "org.oma" property as an object defined in section 9.4.1.1

When a connection refers to user-specific information (user account on External SN):

- the Account object SHOULD further contain at least one of the "username" and "userId" properties to identify the user on that External SN
- The "org.oma" object in the Account object SHALL contain a "connectionId" property

9.4.1.1 SNeW-specific information

An object containing the following properties is defined in this specification to represent information about a connection. The "siteName" and "siteIcon" properties SHOULD be provided.

Name	Type	Description
connectionId	Object-Id	Id of the link relationship of the user account and the External SN user account.
siteName	string	The name of External SN e.g. Twitter.
siteIcon	URI	The public URL of the icon identifying the External SN. e.g. "http://example.com/sites/logo/twitter.png"
profileUrl	URI	User's profile page URL in the External SN (when applicable)
thumbnailUrl	URI	User's avatar URL in the External SN (when applicable)
connectionStatus	string	Status of the link relationship

The following values and related meaning are defined for the connectionStatus property:

`active`	The connection to the External SN is known to be active with valid permissions.
`expired`	The connection to the External SN has expired (e.g. OAuth2 token has expired) and needs to be refreshed before using the gateway functionality with this connection. The SNEW Client SHOULD attempt to update the connection according to section 7.1.4.5 in order to refresh the authorization from the user to that External SN.
`insufficient_permissions`	The connection to the External SN does not have the required grants (e.g. insufficient permissions, user explicitly revoked some permission on the External SN that are required for some gateway functionality). Note that in this state some gateway functionality may still work with this connection. The SNEW Client MAY attempt to update the connection according to section 7.1.4.5 in order to request the user to grant all the required permissions on that External SN.
`inactive`	The connection to the External SN is known to be inactive (e.g. the connection was disconnected by the user on the External SN by revoking all permissions or by deleting the application explicitly)

If the connectionStatus property is not provided, the active value MUST be implied.

9.4.2 Reference in activities and reactions

Activities and reactions on External SNs can be referenced both by their connection (i.e. the user account on the External SN to which they are associated) and their specific identifier on that External SN.

SNEW Clients and SNEW Servers SHALL use an “org.oma” property in Activity objects as an object defined in section 9.4.2.1 to reference the connection(s) associated with these activities and/or reactions.

This property SHALL be inserted:

- By an SNEW Client in the Activity object sent for creating an activity or reaction to indicate one or more External SN target connections where the SNEW Server SHALL attempt cross-posting this activity (outbound interaction)
- By an SNEW Server in the Activity object returned in the response to the successful creation of an activity or reaction to indicate on which connections it was successfully cross-posted (outbound interaction)
- By an SNEW Server in the Activity objects returned in the response to the successful retrieval of activities and/or reactions to indicate from which External SN source connections it was aggregated (inbound interaction)

Furthermore, SNEW Servers SHALL use the “downStreamDuplicates” property defined in [AS-JSON] for Activity objects to reference the specific identifier(s) of the related activity or reaction on the External SN(s).

This property SHOULD be inserted by SNEW Servers:

- In the Activity object returned in the response to the successful creation of an activity or reaction that was cross-posted to one or more External SN(s) by the SNEW Server (outbound interaction)
- In the Activity objects returned in the response to the successful retrieval of activities and/or reactions that were aggregated from External SNs (inbound interaction)

See Appendix G.1.4 for examples related to references to External SNs in the ActivityStreams Service.

9.4.2.1 SNeW-specific information

An object containing the following properties is defined in this specification to reference connections associated with activities and/or reactions.

Name	Type	Description
connections	Array	An array of Object-Id representing the connectionIds associated with this activity/reaction.

10.Security considerations

SNEW Clients and SNEW Servers SHALL support HTTP over Transport Layer Security (TLS) as specified in [OMA TLS] at least over SNEW-1 and SNEW-2 interfaces.

10.1 Authentication

User authentication is expected on SNEW-1 and SNEW-2 interfaces as a prerequisite to authorization, described in section 10.2. In particular, as part of the authorization flows, SNEW Servers MAY decide to authenticate the user based on one of the following procedures:

- username and password;
- authentic network information as described in section 10.1.1.

10.1.1 User authentication based on authentic network information

The SNEW Server MAY choose to authenticate the user based on MSISDN or other authentic network information. This approach may be used for authentication in cases which client-provided authentication is not available or not trusted.

In a first case, the RADIUS mechanisms defined in [3GPP 29.061] may be used to provide the SNEW Server with the network-provided user identifier (or subscriber identifier, e.g. MSISDN or IMSI) to authenticate the user using this information as asserted identity.

In alternative, SNeW service providers MAY use other network-based solutions out of scope of this specification, which MUST rely upon authentic user information. For example, an SNEW Server may handle requests from SNEW Clients behind a front-end proxy, which inserts a subscriber identity indication (e.g. in a HTTP header) based upon network information (e.g. from RADIUS) into the request, and forwards it to the SNEW Server.

10.2 Authorization

SNEW-1 interface (as per section 7.1) and SNEW-2 Network API (as per section 7.2) rely on [OS-Social-API], which is further based on [OS-Core-API].

As such, over these interfaces SNEW Servers and SNEW Clients SHALL conform to [OS-Core-API] section 3 with the following clarifications:

- they MAY NOT support OAuth1.0a as authorization framework besides OAuth2(which SHALL be supported);
- they MAY NOT support the “client credential” grant type on the token endpoint besides the “authorization code” grant type (which SHALL be supported);
- they SHALL implement the refresh token pattern.

When supporting the Gateway functionality, SNEW Servers MAY support authorization through External SN accounts according to section 10.2.2.

Furthermore, SNEW Servers and SNEW Clients MAY support the authorization framework defined in [Autho4API_10].

In any case, SNEW Servers and SNEW Clients SHALL further support at least the scopes defined in section 10.2.1. Additional scope values not defined in this specification MAY also be supported.

SNEW Clients SHALL use “snew://oauth2-callback” as redirection endpoint URI in OAuth2 flows.

Please see Appendix J for more information about the procedures to obtain an OAuth2 token.

In addition, SNEW Clients and SNEW Servers SHALL support authenticated requests using the "Bearer" HTTP authorization scheme in the "Authorization" request header field as defined in [RFC6750].

10.2.1 Scope values

10.2.1.1 Definitions

The following table describes the list of scope values for SNeW.

Scope value	*Description*	*For one-time access token*
oma_rest_snew.all_{apiVersion}	Provide access to all defined operations over SNEW-1 and/or SNEW-2 Network API in this version of the enabler. The {apiVersion} part of this identifier SHALL have value "1.0" in this version of the specification. This scope value is the union of the other scope values listed below.	No
oma_rest_snew.Service-Name_{perms}	Provide access to defined operations over SNEW-1 and/or SNEW-2 Network API based on the specific OpenSocial Service identified by the Service-Name (see [OS-Social-API]). The type of operations allowed depend on the {perms} part value as follows: – "read" allows read-only operations using GET – "manage" allows all types of operations (GET, POST, PUT, DELETE)	No

Table 2: Scope values for SNeW

10.2.1.2 Downscoping

In the case of authorization requests for "oma_rest_snew.all_{apiVersion}" scope (e.g. from a SNEW Enabled Application or from the SNEW Client), the SNEW Server MAY restrict the granted scope to some of the following scope values:

- "oma_rest_snew.people_read"
- "oma_rest_snew.activitystreams_read"

10.2.2 Authorization through External SN

When supporting authorization through External SN, SNEW Servers SHALL support the "provider" request parameter over their OAuth2 Authorization endpoint.

When present, this parameter SHALL contain the domain of an External SN (e.g. "facebook.com", "twitter.com") the user wants to login with in order to perform authorization with the SNEW Server. In that case, the SNEW Server will activate the procedure(s) to allow the user to connect to the specified External SN and use their account as basis for authorization.

This functionality is particularly useful for provisioning accounts based on External SN accounts and to perform subsequent authorization requests via authentication on the External SN. See Appendix J.3 for more information.

11.Release Information

11.1 Supporting File Document Listing

Doc Ref	Permanent Document Reference	Description
Supporting Files		

Table 3: Listing of Supporting Documents in SNeW 1.0 Release

11.2 OMNA Considerations

SNeW 1.0 includes the following OMNA items:

1. Push-Application ID
    a. x-oma-application:snew.ua (*new in SNeW 1.0*)
2. Push Resource Identifier
    a. snew.ua (*new in SNeW 1.0*)

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-ER-SNeW-V1_0-20160329-A	29 Mar 2016	Status changed to Approved by TP TP Ref # OMA-TP-2016-0049-INP_SNeW_V1_0_ERP_for_final_Approval

Appendix B. Use Cases (Informative)

This Appendix provides high-level use cases focused on the users and deployment scenarios point of view, targeting release's requirements.

B.1 Gateway Scenario

B.1.1 Short Description

This scenario aims at enabling users to interconnect with External SNs on which they already have an account (e.g. Facebook, Twitter, others) using the proprietary interfaces of such networks. This scenario introduces the concept of direction in interacting between an OMA Compliant SN and one or more External SNs.

In particular, Inbound interactions relate to the concept of *Aggregation* of activities, reactions and media from External networks, making thus possible to allow users that own accounts on multiple External SNs to access aggregated information about their friends (e.g. contact information, activities) on these networks. Outbound interactions, on the opposite, relate to the ability of *Cross-posting* activities, reactions and/or media to multiple External SNs. In this way, users could potentially share their activities or reactions over all their External SNs at once.

Step 1: User A updates status and gets notification based on comments from User B to the status update:

1. user A associates her identity on External SNs (SN_i with i=1,2, …) with its identity on an OMA Compliant SN using the proprietary procedures required by the External SNs
2. At a later stage, user A wants to update her status
3. user A selects to which External SN her status update will be posted and posts it (at least she selects SN 2) (*Cross-posting*)
4. user B2 is a friend of user A2 (other identity of user A) on External SN 2, and sees user's A2 new status
5. user B2 comments the new status of user A2 on External SN 2 (out-of-scope)
6. comment from user B2 gets notified to user A on her OMA Compliant SN (*Aggregation*)

Step 2: User B updates status and User A gets notification based the update:

7. user B2 updates her status on External SN 2, which is seen by user A2
8. user A gets notified of user B2 status update on her OMA Compliant SN (*Aggregation*)

This scenario aims at defining the requirements needed to enable gateway functionalities in a generic way, not addressing the peculiarities of the single External SN.

Furthermore, the concept of activity in this scenario is very generic. Whilst the scenario itself focuses on a status update, media upload and check-in activities are intended as additional valid examples.

B.1.2 Market benefits

Mobile operators can provide the social network gateway function in a standalone way or in addition to their own mobile social network. As such it can attract more users and provide competitive services. Users would be attracted by a social network which can provide the gateway function for communication with friends in multiple SNs.

B.2 Federated Scenario

B.2.1 Short Description

The federated Social Web community has defined a simple social federation scenario named “Social Web Acid Test - Level 0” [SWAT0] as an integration use case for the federated social web.

This scenario aims at enabling users to interact in a standard way across social networks run by service providers compliant with a set of interoperable specifications.

The generalized flavour of this scenario is copied here for convenience, where “different services” has to be understood as “different services on different Social Networks”:

“

1. *user A takes a photo of B from their phone and posts it*
2. *user A explicitly tags the photo with B*
3. *B gets notified that they are in a photo*
4. *C who follows A gets the photo*
5. *C makes a comment on the photo*
6. *A and B get notified of the comment*

*Where users are on at least 2 (ideally 3) different services […], and the users only need to have *one* account, on the specific service of their choice (requiring the users to have an account on each service that is interacting misses the point of Federation)”*

B.2.2 Market benefits

Social Network Federation enables users to only have *one* account for SN, whilst still giving them the ability to interact with users on different Federated SNs. It is thus possible to enable a Federated SN as the anchor of mobile operator’s service and other services.

The mobile operator can thus operate its own SN thus attracting its own subscribers to an interoperable SN service that allow them to communicate and interact freely with subscribers of other mobile operators.

B.3 Multiple devices support

B.3.1 Short Description

This scenario aims at enabling user using different mobile phones or devices to access his mobile social network services. User would like to access the mobile social network from different devices, such as using PC while at home or using a mobile phone while on the road. A user may also use multiple mobile phones to connect the social network sites. The mobile social network services need to be provided as other mobile services which are independent of the devices. Some of the operations that users can carry out in multiple devices scenario are:

1. user A accesses the mobile social network service by PC.
2. user A writes a new blog post.
3. user B accesses the mobile social network service through his mobile phone.
4. user B follows user A and get the notification to the mobile phone that user A has a new blog post.
5. user A accesses the mobile social network service using his mobile phone.

6. user B switches device, e.g. accesses the mobile social network service using his PC.
7. user A updates his status through his mobile phone.
8. user B get the notification to the PC that user A has updated his status.

B.3.2 Market benefits

The mobile operator would set mobile social network as one major entry in the mobile services if it is device independent.

The subscribers / end users will have more options and freedom to switch mobile phones and enjoy the mobile social network with any device.

The manufacturers will be able to make the mobile phone that can connect to all mobile social network services of all operators. Thus the manufacturers will benefit from lowering the development cost, shortening the time to market, and increasing the market share by outreaching to broader markets.

B.4 Multiple device applications support

B.4.1 Short Description

This scenario aims at enabling multiple applications running in a device to benefit from the interworking with SNs in an optimized fashion. In fact there may be multiple applications in a device that require exchanging information with one or several SNs. Instead of managing this interworking with the SNs on their own, the applications may rely on a specialized entity in the device, i.e. the SNEW Client, which relieves the applications from this task by centralizing the communication with the user's OMA Compliant Social Network, which in turn interworks with other OMA or non-OMA Compliant Social Networks. The centralization of this interworking through the SNEW Client results also in an optimization of the use of air data traffic.

Device applications may have different needs in terms of interworking with SNs, for instance:

- Application 1: Social Network Aggregation Client, that allows a real-time interaction with multiple SNs (A, B and C):
  - Receive push notification of updates from SNs A, B and C
  - Update status and post media to SNs A, B and C
- Application 2: Multimedia Gallery application that stores all pictures from SNs A and C in which the user is tagged
  - Receive push notifications about the user having been tagged in pictures from SNs A and C.
- Application 3, Address Book application, that enriches the contacts with information from SN B
  - On demand access to contact information from SN B

As a first step the different applications have to register to the SNEW Client their needs in terms of interaction with the SNs. The user needs to grant permission to each application to access the requested resources and/or perform the requested operations.

Provided that the user grants access, the applications are able from that point of time to interwork with the SNs as requested, that is, the applications are provided with the information from the SNs that match the filters indicated by the application at the registration phase and are also able to post information towards the selected SNs.

A special case occurs when an application posts towards the SNs a piece of information that another application running in the same device is intended to receive. In this case the SNEW Client may cache this information to make it available to any other interested applications on the same device, in addition to posting it to the network. In this case there is no need to get this piece of information delivered back from the network to the SNEW Client, as it was generated at the device and the SNEW Client has already cached it locally.

B.4.2 Market benefits

Application developers will be able to develop applications that require interaction with SNs more easily, lowering the development cost, shortening the time to market and thus increasing the application portfolio.

The end users will have a wider offer of applications to access the SNs and will get other type of applications enriched with information from the SNs.

Air data traffic will be minimized as every piece of information will be exchanged with the SNs just once even if there are multiple applications on a device that require exchanging that same piece of information with the SNs.

Appendix C. Static Conformance Requirements

The notation used in this appendix is specified in [SCRRULES].

C.1 ERDEF for SNeW 1.0 - Client Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-SNeW-C-001-M	SNEW Client	[SNeW-ER]:MCF
OMA-ERDEF-SNeW-C-002-O	Client gateway functionality	SNeW-GW-C-MCF
OMA-ERDEF-SNeW-C-003-O	Subscription-mode functionality	SNeW-SUB-C-MCF

Table 4: ERDEF for SNeW 1.0 Client-side Requirements

C.2 ERDEF for SNeW 1.0 - Server Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-SNeW-S-001-M	SNEW Server	[SNeW-ER]:MSF
OMA-ERDEF-SNeW-S-002-O	Server gateway functionality	SNeW-GW-S-MSF
OMA-ERDEF-SNeW-S-003-O	Subscription-mode functionality	SNeW-SUB-S-MSF
OMA-ERDEF-SNeW-S-004-O	Social data import/export functionality	SNeW-IE-S-MSF

Table 5: ERDEF for SNeW 1.0 Server-side Requirements

C.3 SCR for SNEW Client

Item	Function	Reference	Requirement
SNEW-C-001-M	Storage of pending requests in a local queue	Section 8.1.1	
SNEW-C-002-O	Send the queued requests in chronological order	Section 8.1.1	
SNEW-C-003-O	Optimize the queue to avoid conflict and/or duplication of requests	Section 8.1.1	
SNEW-C-004-M	Support authentication	Section 8.1.1 and section 10.2.	
SNEW-C-005-M	Update the local cache when receiving responses from SNEW Server	Section 8.1.2	
SNEW-C-006-M	Forward the retrieved information to the requesting application(s) on the device	Section 8.1.2	

Item	Function	Reference	Requirement
SNEW-C-007-M	Upon reception of a request from SNEW Enabled Device Applications aiming at retrieving information, check the information cached locally for this information	Section 8.1.3	
SNEW-C-008-M	If the requested information is not present in the cache, or if such information does not match the relevant policies (e.g. deprecated information), issue the associated request to the SNEW Server	Section 8.1.3	
SNEW-C-009-M	If the requested information is present in the cache and valid, return the related information cached locally and do not issue request to SNEW Server	Section 8.1.3	
SNEW-C-010-M	Upon reception of a request from SNEW Enabled Device Applications aiming at providing information, either in the form of a creation or an update (e.g. posting an activity), issue the request to the SNEW Server and insert the related information in the local cache, or update it if already present	Section 8.1.3	
SNEW-C-011-O	When inserting information in the local cache, or updating it if already present, specify a duration of validity of this information	Section 8.1.3	
SNEW-C-012-M	Upon reception of a request from SNEW Enabled Device Applications aiming at deleting information, issue the request to the SNEW Server and delete the related information from the local cache, if present	Section 8.1.3	
SNEW-C-013-M	Retrieve, create and delete accounts and relationships (People Service)	Section 7.1 and section 7.1.1	
SNEW-C-014-M	When deleting a specific relationship, insert a Related-User-Id in the REST-URI-Fragment	Section 7.1.1.1	
SNEW-C-015-O	Insert a Related-User-Id in the REST-URI-Fragment when updating a specific relationship	Section 7.1.1.1	
SNEW-C-016-O	update accounts and relationships (People Service)	Section 7.1 and section 7.1.1	SNEW-C-015-O
SNEW-C-017-M	Support sending multipart/related content when creating a person	Section 7.1.1.2 and section 7.1.3	

Item	Function	Reference	Requirement
SNEW-C-018-M	Be prepared to receive responses informing that the request was accepted (e.g. 202 response code) but is not yet completed and typically expect some out-of-band procedures for completion	Section 7.1.1.3	
SNEW-C-019-M	Support sending multipart/related content when creating an activity	Section 7.1.2 and section 7.1.3	
SNEW-C-020-O	Support sending search requests using parameters “filterBy”, “filterOp” and “filterValue”	Section 7.1.1.4	
SNEW-C-021-O	Support users joining and leaving groups as People Service	Section 7.1.1.5	
SNEW-C-022-O	Retrieve, create and delete groups (Groups Service)	Section 7.1	
SNEW-C-023-O	Update groups (Groups Service)	Section 7.1	SNEW-C-022-O
SNeW-C-024-M	Retrieve, create and delete activities & reactions (ActivityStreams Service)	Section 7.1	
SNeW-C-025-O	Update activities & reactions	Section 7.1	
SNeW-C-026-O	Support abuse reporting as activity/reaction	Section 7.1 and section 9.2.4.1	
SNeW-C-027-M	Support privacy settings and target audience in activities & reactions	Section 7.1.2 and section 9.2.3	
SNeW-C-028-M	include Activity-Id in REST-URI-Fragment when updating an activity	Section 7.1.2	
SNeW-C-029-M	Retrieve, create and delete messages (Message Service)	Section 7.1	
SNeW-C-030-O	Update messages	Section 7.1	
SNeW-C-031-M	support IRIs as a valid Global-Id format	Section 7.1	
SNeW-C-032-M	Support “@all”, “@following”, “@followers”, “@friends” and “@self” Group-Id values	Section 7.1	
SNeW-C-033-M	support indirect delivery of push content, triggered by reception of ServiceLoading notifications	Section 7.5	[OMAPUSH]:MCF AND [OMAPUSH-SL]:MCF
SNeW-C-034-M	support device-internal routing of Push messages with the Push Application ID “x-oma-application:snew.ua” to the SNEW Client	Section 7.5	[OMAPUSH]:MCF
SNeW-C-035-M	If there is no Push Client in the device, implement the necessary Push Client functions for the supported Push-OTA protocol variants	Section 7.5	[OMAPUSH]:MCF
SNeW-C-036-M	support “snew.ua” as Push Resource Identifier for SIP Push OTA binding (when supported)	Section 7.5	[OMAPUSH]:MCF
SNeW-C-037-M	Support tel: and “acct:” URIs as user identifiers	Section 9.1	

Item	Function	Reference	Requirement
SNeW-C-038-M	Support activities and reactions as ActivityStreams json format	Section 9.2	
SNeW-C-039-M	Support activity streams base schema: – Verbs: checkin, follow, like, share, stop-following, unlike – Object types: application, comment, image, note, person, place, video	Section 9.2.1	
SNeW-C-040-O	represent location according to activity streams base schema extension in json.	Section 9.2.5	
SNeW-C-041-O	Support location information in activity entries	Section 9.2.1	SNeW-C-040-O
SNeW-C-042-O	represent tags according to activity streams base schema extension in json.	Section 9.2.5	
SNeW-C-043-O	Support tags information in activity entries	Section 9.2.1	SNeW-C-042-O
SNeW-C-044-M	include “id” and “displayName” attributes in person information	Section 9.3	
SNeW-C-045-O	include “thumbnailUrl” attributes in person information	Section 9.3	
SNeW-C-046-M	Support TLS	Section 10.1	
SNeW-C-047-M	Support OAuth2 “authorization code” grant type	Section 10.2	
SNeW-C-048-M	Support the Oauth2 refresh token pattern	Section 10.2	
SNeW-C-049-M	Use “snew://oauth2-callback” as redirection endpoint URI	Section 10.2	
SNeW-C-050-M	Support standard scope values	Section 10.2.1.1	
SNeW-C-051-M	Support "Bearer" HTTP authorization scheme in the "Authorization" request header	Section 10.2	

C.4 SCR for SNEW Server

Item	Function	Reference	Requirement
SNEW-S-001-M	Exposure of descriptors	Section 8.2.1	
SNEW-S-002-M	Retrieve a descriptor describing how to find a user's public metadata from that user's email address or email address like identifier, performing a Webfinger query	Section 8.2.2	
SNEW-S-003-O	Cache the information provided in retrieved descriptors	Section 8.2.2	
SNEW-S-004-O	Issue conditional requests at a later stage to check for updates of retrieved descriptors	Section 8.2.2	

Item	Function	Reference	Requirement
SNEW-S-005-O	Read additional content from the retrieved descriptor depending on the action requested by the SNEW Client	Section 8.2.2	
SNEW-S-006-M	Retrieving user activities and/or reactions at remote server and validate the retrieved feed as valid ActivityStreams Atom feed	Section 8.2.3	
SNEW-S-007-O	If the retrieved feed contains a hub relation link, subscribe to that feed	Section 8.2.3	
SNEW-S-008-M	Retrieving user information at remote server sending a request to that endpoint based on the required operation (e.g. retrieve a user profile information, retrieve user relationships, etc).	Section 8.2.4	
SNEW-S-009-M	Handling requests from remote SNEW Servers accordingly to table in Section 8.2.5	Section 8.2.5	
SNEW-S-010-M	Handling requests from SNEW Clients and/or SNEW Enabled Applications accordingly to table in Section 8.2.6	Section 8.2.6	
SNEW-S-011-M	Authenticate the user	Section 8.2.6	
SNEW-S-012-O	Gateway function for interacting with External SNs	Section 8.2.6	
SNEW-S-013-O	Trigger additional requests towards External SNs according to their own mechanisms	Section 8.2.6	SNEW-S-012-O
SNEW-S-014-M	In case a request needs to be forwarded to one or more recipients, follow the address resolution procedures	Section 8.2.6	
SNEW-S-015-M	When sending a request to an SNEW Client or to another SNEW Server that contains a reference to the originator's address, insert the asserted user identity in the form of a global identifier	Section 8.2.6	
SNEW-S-016-O	Override the originator's address provided by the originator SNEW Client	Section 8.2.6	
SNEW-S-017-O	Transcode information between incoming and outgoing interfaces.	Section 8.2.6	

Item	Function	Reference	Requirement
SNEW-S-018-O	Specify cache control directives and/or cache validators to control expiration times and/or conditional requests	Section 8.2.6	
SNEW-S-019-M	Deliver asynchronous notifications/information messages to SNEW Clients, converting the information in JSON format, include the appropriate Push Application ID header, forward the information as Push Content to the SNEW Client	Section 8.2.7	
SNEW-S-020-M	Support receiving multipart/related content when creating a person	Section 7.1.1.2 and section 7.1.3	
SNEW-S-021-M	Support receiving multipart/related content when creating an activity and/or reaction	Section 7.1.2 and section 7.1.3	
SNEW-S-022-O	include the number of “followers”, “following” and “friends” in Person objects	Section 7.1.1.1	
SNEW-S-023-O	Apply restrictions and/or additional security mechanisms based on service provider policies when creating and/or deleting a user account	Section 7.1.1.3	
SNEW-S-024-M	when a user requests to delete his account, upon acceptance, delete all related user information (including activities and reactions, etc) unless there are legitimate reasons to retain it	Section 7.1.1.3	
SNEW-S-025-O	Support understanding parameters “filterBy”, “filterOp” and “filterValue” as search requests	Section 7.1.1.4	
SNEW-S-026-O	Support at least “displayName” as search criteria	Section 7.1.1.4	SNEW-S-025-O
SNEW-S-027-O	Support search requests over local users	Section 7.1.1.4	SNEW-S-026-O
SNEW-S-028-O	Apply restrictions and/or additional security mechanisms to search requests based on service provider policies	Section 7.1.1.4	SNEW-S-027-O
SNEW-S-029-O	Allow users joining and leaving groups as People Service	Section 7.1.1.5	
SNEW-S-030-O	Retrieve, create and delete groups (Groups Service)	Section 7.1	
SNEW-S-031-O	Update groups (Groups Service)	Section 7.1	SNEW-S-030-O
SNeW-S-032-M	Retrieve, create and delete activities & reactions (ActivityStreams Service)	Section 7.1	

Item	Function	Reference	Requirement
SNeW-S-033-O	Update activities & reactions	Section 7.1	
SNeW-S-034-O	Support abuse reporting as activity/reaction	Section 7.1 and section 9.2.3	
SNeW-S-035-M	Support privacy settings and target audience in activities & reactions	Section 7.1.2 and section 9.2.3	
SNeW-S-036-O	default privacy setting for activities and reactions is restricted only to the users, whom the actor has explicitly approved relationships	Section 7.1.2	
SNeW-S-037-M	Support Activity-Id in REST-URI-Fragment when receiving an activity update/deletion	Section 7.1.2	
SNeW-S-038-M	Retrieve, create and delete messages (Message Service)	Section 7.1	
SNeW-S-039-O	Update messages	Section 7.1	
SNeW-S-040-M	support IRIs as a valid Global-Id format	Section 7.1	
SNeW-S-041-M	Support “@all”, “@following” and “@followers”, “@friends” and “@self” Group-Id values	Section 7.1	
SNeW-S-042-M	expose at least one OExchange Target endpoint and reference it in its descriptor	Sections 7.2, 7.2.1 and 8.2.1	
SNeW-S-043-M	support retrieving activity entries over Network API	Section 7.2.2 and section 7.1	
SNeW-S-044-O	support the creation, deletion, and update of activity entries over Network API	Section 7.2.2 and section 7.1	
SNeW-S-045-M	Support Salmon protocol to send/receive notifications and reference the related endpoint(s) in the user descriptor	Section 7.3.1 and 8.2.1	
SNeW-S-046-M	Support PubSubHubbub protocol to send/receive notifications and expose at least one callback endpoint	Section 7.3.2	
SNeW-S-047-O	Act as PubSubHubbub hub	Section 7.3.2	
SNeW-S-048-M	expose at least one ActivityStreams Atom endpoint that contains a hub link and atom activity entries and reference it in the user descriptor	Sections 7.3.3, 8.2.1 and 9.2	
SNeW-S-049-O	Include a link to salmon endpoint in activitystreams atom feeds	Section 7.3.3	
SNeW-S-050-M	expose at least one PoCo endpoint and reference it in the user descriptor	Section 7.3.4 and 8.2.1	
SNeW-S-051-M	Support PAP, PushREST and/or Push-OTA for sending push notification	Section 7.5	[OMAPUSH]:MSF OR [OMAPUSH-PAP]:MSF OR [OMAPUSH-REST]:MSF
SNeW-S-052-M	support direct and indirect Push delivery	Section 7.5	

Item	Function	Reference	Requirement
SNeW-S-053-M	Represent SNEW push content as Json	Section 7.5	
SNeW-S-054-M	support at least the “URI” target client address scheme	Section 7.5	
SNeW-S-055-M	encapsulate push content into a “message/vnd.oma.push” media type for direct delivery over OTA	Section 7.5	[OMAPUSH-MSG]:MSF
SNeW-S-056-M	use the ServiceLoading content type for indirect delivery	Section 7.5	[OMAPUSH-SL]:MSF
SNeW-S-057-M	support “snew.ua” as Push Resource Identifier for SIP Push OTA binding (when supported)	Section 7.5	[OMAPUSH]:MSF
SNeW-S-058-M	Support tel: and “acct:” URIs as user identifiers	Section 9.1.1.1	
SNeW-S-059-M	Resolve user identifiers into local/remote domain	Section 9.1.1.2	
SNeW-S-060-O	Use ENUM for tel: URI resolution	Section 9.1.1.2 and Appendix F	
SNeW-S-061-M	Support activities and reactions as ActivityStreams json and atom format	Section 9.2	
SNeW-S-062-M	Support activity streams base schema: – Verbs: checkin, follow, like, share, stop-following, unlike – Object types: application, comment, image, note, person, place, video	Section 9.2.1	
SNeW-S-063-O	represent location according to activity streams base schema extension in json, and georss-point in atom.	Section 9.2.5	
SNeW-S-064-O	Support location information in activity entries	Section 9.2.1	SNeW-S-063-O
SNeW-S-065-O	represent tags according to activity streams base schema extension in json, and category in atom.	Section 9.2.5	
SNeW-S-066-O	Support tags information in activity entries	Section 9.2.1	SNeW-S-065-O
SNeW-S-067-M	include “id” and “displayName” attributes in person information	Section 9.3	
SNeW-S-068-O	include “thumbnailUrl” attributes in person information	Section 9.3	
SNeW-S-069-M	Support TLS	Section 10.1	
SNeW-S-070-O	Support user authentication based on authentic network information	Section 10.1.1	
SNeW-S-071-M	Support OAuth2 “authorization code” grant type	Section 10.2	

Item	Function	Reference	Requirement
SNeW-S-072-M	Support the Oauth2 refresh token pattern	Section 10.2	
SNeW-S-073-M	Support standard scope values	Section 10.2.1.1	
SNeW-S-074-O	In case of authorization requests for "oma_rest_snew.all_{apiVersion}" scope restrict the granted scope to: • "oma_rest_snew.people_read" • "oma_rest_snew.activitystreams_read"	Section 10.2.1.2	
SNeW-S-075-M	Support "Bearer" HTTP authorization scheme in the "Authorization" request header	Section 10.2	
SNEW-S-076-O	include the relationship indication in Person objects	Section 7.1.1.1	

C.5 SCRs for Gateway functionality

Item	Function	Reference	Requirement
SNeW-GW-C-001-M	Associate protocol scheme "snew:" to the SNEW Client on the device to detect the response from browser	Section 7.1.4.4	
SNeW-GW-C-002-M	Represent connection information in json	Section 9.4.1	
SNeW-GW-C-003-M	include references to connections in activities & reactions when posting	Section 9.4.2	
SNeW-GW-C-004-M	Gateway functionality	Section 7.1 and Section 7.1.4	

Item	Function	Reference	Requirement
SNeW-GW-S-001-M	Reject requests for deletion of connections using the "@all" value as Connection-Id	Section 7.1.4.3	
SNeW-GW-S-002-M	include "connectionId" and "domain" in responses to successful connection creation	Section 7.1.4.4	
SNeW-GW-S-003-M	Represent connection information in json	Section 9.4.1	
SNeW-GW-S-004-M	include references to connections in activities & reactions	Section 9.4.2	
SNeW-GW-S-005-M	Gateway functionality	Section 7.1 and Section 7.1.4	

C.6 SCRs for Subscription-mode functionality

Item	Function	Reference	Requirement
SNeW-SUB-C-001-M	insert Accept: header value "text/event-stream" in retrieval/subscription requests	Section 7.1.5	
SNeW-SUB-C-002-O	issue the subscription request to the local Push API Server	Section 7.1.5	[OMAPUSH-WRAPI]:MCF
SNeW-SUB-C-003-O	issue the subscription request directly to the subscription endpoint and when receiving subscription responses, prepare the related Push notification filters for this subscription	Section 7.1.5 and Appendix H	
SNeW-SUB-C-004-M	include a "hub.topic" and (an "Accept" header with value "text/event-stream" and/or an "X-Oma-Push-Accept-Source" header) in the subscription request	Section 7.1.5 and Appendix H	SNeW-SUB-C-002-O OR SNeW-SUB-C-003-O
SNeW-SUB-C-005-M	Subscription-mode	Section 7.1 and Section 7.1.5	

Item	Function	Reference	Requirement
SNeW-SUB-S-001-O	Accept subscription requests containing no "hub.callback" parameter if such requests contain an "Accept" header with value "text/event-stream" and/or an "X-Oma-Push-Accept-Source" header	Section 7.1.5 and Appendix H	
SNeW-SUB-S-002-O	Reject subscription requests containing no "hub.callback" parameter over Network API	Section 7.2.2 and section 7.1.5	
SNeW-SUB-S-003-O	insert at least one Link Header with rel=hub indicating the URL of the subscription endpoint in retrieval responses	Section 7.1.5	
SNeW-SUB-S-004-O	Subscription-mode over MSN-1	Section 7.1 and Section 7.1.5	SNeW-SUB-S-001-O AND SNeW-SUB-S-003-O
SNeW-SUB-S-005-O	Support subscription-mode over Network API	Section 7.2.2 and section 7.1.5	SNeW-SUB-S-002-O AND SNeW-SUB-S-003-O
SNeW-SUB-S-006-M	Support subscription-mode		SNeW-SUB-S-004-O OR SNeW-SUB-S-005-O OR (SNeW-SUB-S-004-O AND SNeW-SUB-S-005-O)

C.7 SCRs for Social data import/export functionality

Item	Function	Reference	Requirement
SNeW-IE-S-001-M	Support social data package format	Section 7.2	
SNeW-IE-S-002-M	When importing a social data package, decompress (and/or extract) the social data package descriptor	Section 8.2.8	

Item	Function	Reference	Requirement
SNeW-IE-S-003-M	Verify the social data package descriptor XRD signature, if present	Section 8.2.8	
SNeW-IE-S-004-O	Check the http://www.openmobilealliance.org/snew/package/prop/version property element to understand the data formats used in the package, in case the social data package descriptor is verified	Section 8.2.8	
SNeW-IE-S-005-M	Import the user data, if the package version is supported	Section 8.2.8	
SNeW-IE-S-006-M	allow users to import and/or export their social information	Section 7.2 and 7.2.3	

Appendix D. Example XRD descriptors (Informative)

D.1 SNEW Server Host descriptor

The following XRD descriptor is an example of a .well-known/host-meta file exposed by an SNEW Server at its root domain address (e.g. http://www.example.com).

```
<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0‘
                xmlns:hm='http://host-meta.net/xrd/1.0'>
<hm:Host>www.example.com</hm:Host>
<Link rel='lrdd'
    template='http://example.com/describe?uri={uri}‘/>
<Link rel="icon"
          href="http://www.example.com/logo" />
<Link rel=http://oexchange.org/spec/0.8/rel/resident-target
          type="application/xrd+xml"
          href="http://www.example.com/oexchange.xrd" />
<Link rel="oauth2-authorize" href="https://example.com/oauth/authorize" />
<Link rel="oauth2-token" href="https://example.com/oauth/token" />
<Link rel="http://ns.opensocial.org/2008/opensocial" href="http://example.com/snew/v1" />
</XRD>
```

D.2 SNEW Server User descriptor

The following JRD descriptor is an example of a user descriptor exposed by an SNEW Server to enable user profile discovery on SNEW-3 interface via the Webfinger endpoint.

```
{
       "subject" : "acct:joe@example.com",
       "properties" :
       {
           "http://salmon-protocol.org/ns/magic-key" :
"RSA.mVgY8RN6URBTstndvmUUPb4UZTdwvwmddSKE5z_jvKUEK6yk1u3rrC9yN8k6FilGj9K0eeUPe2hf4Pj-5CmHww.AQAB"
       },
       "links" :
       [
         {
           "rel" : "http://portablecontacts.net/spec/1.0",
           "href" : "http://example.com/api/people"
         },
         {
```

```
      "rel" : "http://portablecontacts.net/spec/1.0#me",
      "href" : "http://example.com/api/people/joe"
    },
    {
      "rel" : "http://ns.opensocial.org/2008/opensocial/people",
      "href" : "http://example.com/api/people"
    },
    {
      "rel" : "http://webfinger.net/rel/profile-page",
      "type" : "text/html",
      "href" : "http://example.com/profiles/joe"
    },
    {
      "rel" : "describedby",
      "type" : "text/html",
      "href" : "http://example.com/profiles/joe"
    },
    {
      "rel" : "describedby",
      "type" : "application/rdf+xml",
      "href" : "http://example.com/profiles/joe/foaf"
    },
    {
      "rel" : "http://webfinger.net/rel/avatar",
      "href" : "http://example.com/profiles/joe/photo"
    },
    {
      "rel" : "http://schemas.google.com/g/2010#updates-from",
      "type" : "application/atom+xml",
      "href" : "http://example.com/activities/joe"
    },
    {
      "rel" : "salmon",
      "href" : "http://example.com/salmon/joe"
    }
  ]
```

```
}
```

Below is the equivalent descriptor in XRD format. In particular, this XRD descriptor can be reached following the URL indicated in the 'lrdd' relation link template in the Host descriptor of the previous section ('http://example.com/describe?uri={uri}'), substituting the {uri} part with the actual use account (e.g. acct:joe@example.com)

```
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0">

<Subject>acct:joe@example.com</Subject>

<Property xmlns:mk="http://salmon-protocol.org/ns/magic-key"

          type="http://salmon-protocol.org/ns/magic-key" >

  RSA.mVgY8RN6URBTstndvmUUPb4UZTdwvwmddSKE5z_jvKUEK6yk1u3rrC9yN8k6FilGj9K0eeUPe2hf4Pj-5CmHww.AQAB

</Property>

<Link rel='http://portablecontacts.net/spec/1.0'

          href='http://example.com/api/people' />

<Link rel='http://portablecontacts.net/spec/1.0#me'

          href= 'http://example.com/api/people/joe'/>

<Link rel='http://ns.opensocial.org/2008/opensocial/people'

          href='http://example.com/api/people' />

<Link rel='http://webfinger.net/rel/profile-page'

          type='text/html'

          href='http://example.com/profiles/joe' />

<Link rel='describedby'

          type='text/html'

          href='http://example.com/profiles/joe' />

<Link rel='describedby'

          type='application/rdf+xml'

          href= 'http://example.com/profiles/joe/foaf' />

<Link rel='http://webfinger.net/rel/avatar'

          href='http://example.com/profiles/joe/photo' />

<Link rel='http://schemas.google.com/g/2010#updates-from'

          href='http://example.com/activities/joe'

          type='application/atom+xml'/>

<Link rel="salmon"

          href='http://example.com/salmon/joe"/>

</XRD>
```

D.3 SNEW Server OExchange Target descriptor

The following XRD descriptor is an example of a OExchange Target descriptor exposed by an SNEW Server to enable link sharing when browsing external websites (e.g. through a "share" button) on SNEW-2 interface.

```
<?xml version='1.0' encoding='UTF-8'?>
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0">
    <Subject>http://www.example.com/linkeater</Subject>
    <Property type="http://www.oexchange.org/spec/0.8/prop/vendor">Examples Inc.</Property>
    <Property type="http://www.oexchange.org/spec/0.8/prop/title">A Link-Accepting Service</Property>
    <Property type="http://www.oexchange.org/spec/0.8/prop/name">LinkEater</Property>
    <Property type="http://www.oexchange.org/spec/0.8/prop/prompt">Send to LinkEater</Property>
    <Link rel= "icon"
        href="http://www.example.com/favicon.ico"
        type="image/vnd.microsoft.icon" />
    <Link rel= "http://www.oexchange.org/spec/0.8/rel/offer"
        href="http://www.example.com/linkeater/offer.php"
        type="text/html" />
</XRD>
```

D.4 Social Data Package descriptor

The following XRD descriptor is an example of a social data package descriptor created by an SNEW Server to export the social data related to user joe@example.com.

```
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0">
<Subject>acct:joe@example.com</Subject>
<Property type="http://www.openmobilealliance.org/snew/package/prop/version">1.0</Property>
<Link rel='http://ns.opensocial.org/2008/opensocial/people/@me/@self'
          href='file://self.json'/>
<Link rel='http://ns.opensocial.org/2008/opensocial/people/@me/@friends'
          href='file://friends.json' />
<Link rel='http://ns.opensocial.org/2008/opensocial/people/@me/@followers'
          href='file://followers.json' />
<Link rel='http://ns.opensocial.org/2008/opensocial/people/@me/@following'
          href='file://following.json' />
<Link rel='http://webfinger.net/rel/avatar'
          href='file://images/joe.jpg' />
<Link rel='http://ns.opensocial.org/2008/opensocial/activitystreams/@me/@self'
          href='file://activities/self.json' />
<Link rel='http://schemas.google.com/g/2010#updates-from'
          href='file://activities/self.xml'
          type='application/atom+xml'/>
</XRD>
```

Appendix E. Architectural deployments (Informative)

In this section are described possible architectural realizations of the SNeW Enabler according to the architecture defined in section 6.

This section is informative and illustrate interactions and flows between instances of SNeW functional components as well as with external entities, using both SNeW interfaces and external interfaces.

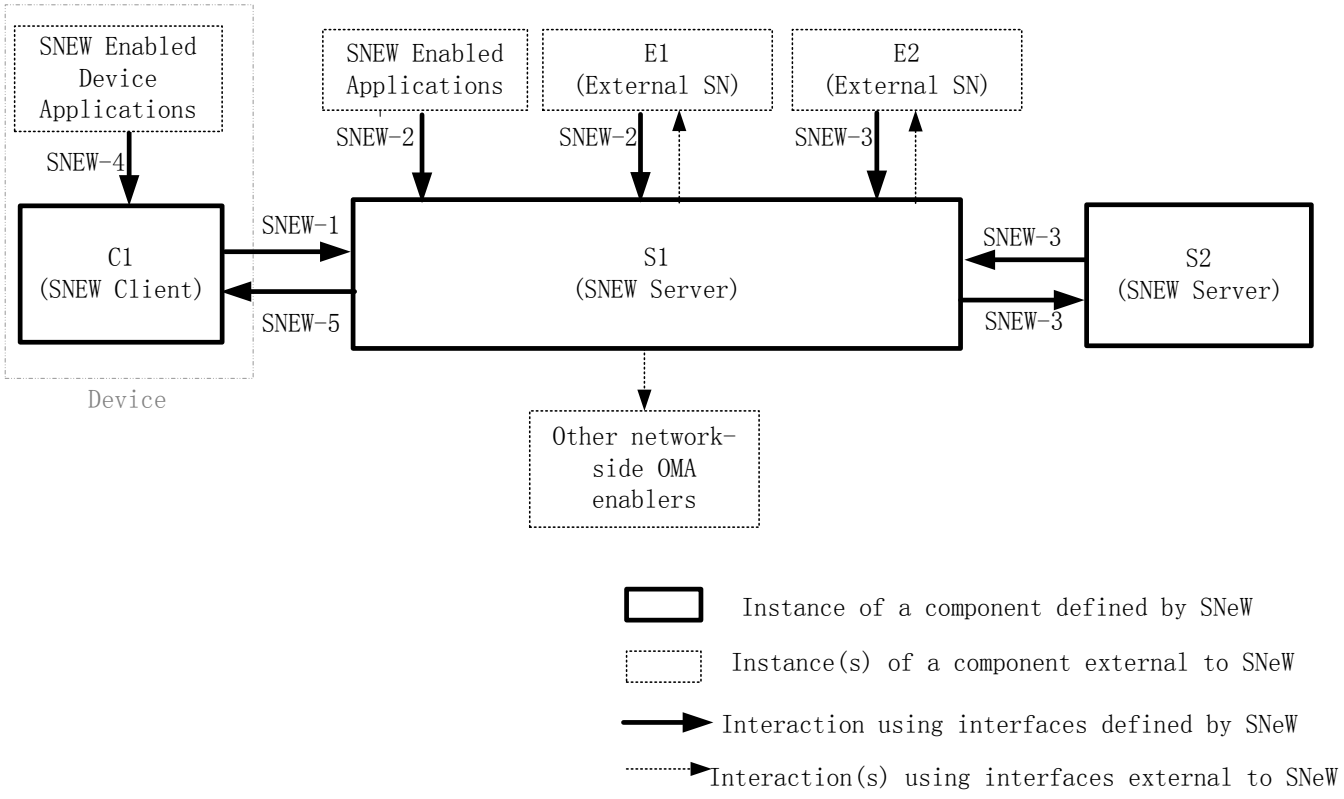

Figure 4: Example of architecture deployment of SNeW Enabler and related flows

Depending on the nature of External SNs, they may rely on SNEW-2 and/or SNEW-3 interface(s) to interact with SNeW Enabler. It is up to Service Providers to define policies to control access to such interfaces from External SNs.

Furthermore, External SNs device applications may be examples of SNEW Enabled Device applications when interacting with SNeW Enabler locally on the device.

Other OMA network-side enablers are external entities with respect to SNeW Enabler that reside on the network and may be accessed in order to provide any relevant enabler functionalities. In particular, internal and/or backend functionalities of SNEW Server may be realized through proprietary interfaces, or through such enablers. Examples of such enablers are CAB, MSF, etc.

In addition, other OMA enablers can reuse SNeW capabilities through either SNEW-2 and/or SNEW-4 interface(s). In that sense, they are considered as examples of SNEW Enabled (Device) applications.

Appendix F. DNS-ENUM recipient MSISDN address resolution (Informative)

For requests asking to interact with a recipient user identified through a tel: URI (e.g. MSISDN address), the SNEW Server willing to use the DNS-ENUM recipient MSISDN address resolution procedure performs as in the following example:

1. The SNEW Server verifies that the recipient address (MSISDN) complies with the E.164 address format and includes the '+' character. In the case of national or local addressing scheme (e.g. only operator code followed by a number), the SNEW Server converts the national or local number to an E.164 address format.

   EXAMPLE: UK phone number 01632 960123

   The UK phone number 01632 960123 (country code 44) is converted to +441632960123

2. The SNEW Server applies the ENUM First Well Known Rule (see [RFC6116] section 3.2) to create the associated FQDN.

   EXAMPLE: 3.2.1.0.6.9.2.3.6.1.4.4.e164.arpa (public top level domain)

3. The SNEW Server requests NAPTR records using this FQDN (see [RFC3403] and [RFC6116] section 3)

   EXAMPLE: The following is an example of NAPTR Resource Records associated with the FQDN derived from the recipient MSISDN address (+441632960123)

   IN NAPTR 100 10 "u" "E2U+sip" "!^.*$!sip:John.Doe@sip.example.com!"

   IN NAPTR 100 11 "u" "E2U+acct" "!^.*$!acct:John.Doe@example.com!"

   IN NAPTR 101 10 "u" "E2U+acct" "!^.*$!acct:Johnny@doe.example.com!"

   IN NAPTR 102 10 "u" "E2U+email:mailto" "!^.*$!mailto:Johnny@doe.example.com!"

   Note: The ENUM service identifiers "E2U+sip" and "E2U+email:mailto" are defined in [RFC6118].

4. If the DNS ENUM service is unavailable, or if no NAPTR record exist for that number, the SNEW Server invokes an appropriate address resolution exception handling procedure (e.g. reporting the error condition to the requesting SNEW entity).

5. Otherwise if NAPTR record(s) exist for that number, the SNEW Server proceeds as follows:

   a. If "acct:" NAPTR record(s) exist for that number

      i. The SNEW Server sorts the " acct:" NAPTR record(s) according to the Order and Preference fields as described in [RFC3403] and [RFC6116]

      ii. The SNEW Server considers the "acct:" URI of the highest precedence "acct:" NAPTR record as the resolved recipient address.

   b. Otherwise, the SNEW Server follows the procedures defined in section 9.1.1.2 and in [ENUM-ACCT] section 6.

   EXAMPLE: The highest precedence "acct:" URI for +441632960123is acct:John.Doe@example.com

Appendix G. Example messages (Informative)

This appendix provides examples of messages over SNEW interfaces.

G.1 Example messages over SNEW-1

G.1.1 Activities and reactions

G.1.1.1 Retrieving activities and/or reactions

The following is an example of a request from an SNEW Client to retrieve all activities and/or reactions related to the user with mobile number +393351234567. In this case the SNEW Client does not include any OAuth2 token in the request and as such will retrieve public information available to anonymous users.

Request

```
GET /snew/v1/activitystreams/tel:+393351234567/@all HTTP/1.1

Host: example.com

Accept: application/json

Content-Length: 0
```

Response

This response provides back the list (as collection) of all the activities and reactions related to that user

```
HTTP/1.1 200 Ok

Content-Length: ...

Content-Type: application/json


{
    "items": [
        {
            "id": "http://example.com/activity/4563",
            "published": "2011-02-10T15:04:55Z",
            "title": "This is John status update",
            "actor": {
                "id": "acct:johndoe@example.com",
                "image": {
                    "url": "http://example.com/johndoe/image"
                },
                "displayName": "John Doe"
            },
            "verb": "post",
            "location": {
                "position": "+48.52+002.20/"
```

```
            },
            "object": {
                "objectType": "note"
                },
            "replies": {
                "totalItems": 2,
                "items": [
                    {
                        "objectType": "comment",
                        "author": {
                            "id": "tel:+393351234567",
                            "image": {
                                "url": "http://example.com/cristinaF/image"
                            },
                            "displayName": "Cristina Ford"
                        },
                        "summary": "What a nice status!",
                        "published": "2011-02-10T15:27:55Z"
                    },
                    {
                        "objectType": "comment",
                        "author": {
                            "id": "acct:bobf@example.org",
                            "image": {
                                "url": "http://example.org/bobf/image"
                            },
                            "displayName": "Bob Ford"
                        },
                        "summary": "Indeed!",
                        "published": "2011-02-10T16:12:55Z"
                    }
                ]
            },
            "likes": {
                "totalItems": 1,
                "items": [
                    {
```

```
                    "objectType": "like",
                    "author": {
                        "id": "acct:bobf@example.org",
                        "image": {
                            "url": "http://example.org/bobf/image"
                        },
                        "displayName": "Bob Ford"
                    },
                    "published": "2011-02-10T16:12:36Z"
                }
            ]
        }
    },
    {
        "id": "http://example.com/activity/4564",
        "published": "2011-02-10T15:25:55Z",
        "title": "This is Alice tweet",
        "actor": {
            "id": "acct:alice@example.com",
            "image": {
                "url": "http://example.com/alice/image"
            },
            "displayName": "Alice Smith"
        },
        "verb": "post",
        "location": {
            "position": "+48.52+002.20/"
        },
        "object": {
            "objectType": "note"
        }
    },
    {
        "id": "http://example.com/activity/4565",
        "published": "2011-02-10T15:25:55Z",
        "actor": {
            "id": "acct:alice@example.com",
```

```
                "image": {
                    "url": "http://example.com/alice/image"
                },
                "displayName": "Alice Smith"
            },
            "object": {
                "objectType": "image",
                "url": "http://example.com/images/2741",
                "location": {
                    "position": "+48.52+002.20/"
                }
            },
            "title": "This is my new photo",
            "verb": "post"
        }
    ]
}
```

G.1.1.2 Posting activities and/or reactions

The following is an example of a request from an SNEW Client to post an activity on behalf of a user.

Request

This requests posts a status update with text “This is my new status update" from user with mobile number +393351234567, posted from location: lat: +48.52, lon: +2.2.

```
POST /snew/v1/activitystreams/@me/@self HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
Content-Type: application/json
Accept: application/json
Content-Length: ...

{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
    },
```

```
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my new status update",
    "verb": "post"
}
```

Response

This response provides back the identifier of the newly created activity object (“1234”), together with its publication time. As per this specification and REST principles a 201 Created response informs that the activity has been successfully created, further including the link to the newly created resource in the Location header.

```
HTTP/1.1 201 Created
Date: Tue, 29 Nov 2011 12:00:00 GMT
Location: http://example.com/snew/v1/activitystreams/tel:+393351234567/@self/1234
Content-Length: ...
Content-Type: application/json

{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my new status update",
    "verb": "post",
    "id": "http://example.com/activities/1234"
}
```

Further request body examples

Checkin

```
{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "place",
        "id": "http://example.com/places/943",
        "location": {
            "position": "+48.52+002.20/"
        }
    },
    "location": {
        "position": "+48.50+002.22/"
    },
    "title": "This is my comment to this new checkin",
    "verb": "checkin"
}
```

Comment an existing activity

```
{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "comment"
    },
    "location": {
        "position": "+44.52+007.70/"
    },
    "title": "This is my comment to the existing post",
    "target": {
        "objectType": "activity",
        "id": "http://example.com/activities/3527"
    },
    "verb": "post"
```

```
}
```

Like an existing activity

```
{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "activity",
        "id": "http://example.com/activities/3527"
    },
    "verb": "like"
}
```

Reporting abuse

```
{
  "published": "2012-02-10T15:04:55Z",
  "actor": {
                 "objectType" : "person",
                 "id": "acct:bob@example.com"
  },
  "verb": "post",
  "object": {
                "objectType": "http://openmobilealliance.org/snew/schema/1.0/abuse",
                "title": "this is offensive"
  },
  "target": {
                "objectType":"activity",
                "id": "http://example.com/activities/1234"
  }
}
```

Restricted status update (to followers only) using “to”

```
{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
```

```
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my new status update for your eyes only",
    "verb": "post",
    "to":[
      {
       "objectType":"group",
       "alias":"http://example.com/snew/v1/people/tel:+393351234567/@followers"
      }
    ]
}
```

Status update with explicit mention using "cc"

```
{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my new status update with Jane",
    "verb": "post",
    "cc":[
      {
       "objectType":"person",
       "id":"acct:jane@example.org"
      }
    ]
}
```

G.1.1.3 Updating activities and/or reactions

The following is an example of a request from an SNEW Client to update an existing activity or reaction on behalf of a user.

Request

This request updates a previous activity with a new title “This is my updated status message" from user with mobile number +393351234567.

```
PUT /snew/v1/activitystreams/tel:+393351234567/@self/1234 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
Content-Type: application/json
Accept: application/json
Content-Length: ...

{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my updated status message",
    "verb": "post",
    "id": "http://example.com/activities/1234"
}
```

Response

This response provides back an acknowledgment that the activity was updated a returns the newly updated Activity object (“1234”).

```
HTTP/1.1 200 Ok
Date: Tue, 29 Dec 2011 12:00:00 GMT
Content-Length: ...
Content-Type: application/json

{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
```

```
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my updated status message",
    "verb": "post",
    "id": "http://example.com/activities/1234"
}
```

G.1.1.4 Deleting activities and/or reactions

The following is an example of a request from an SNEW Client to delete an existing activity or reaction on behalf of a user.

Request

This request deletes a previous activity from user with mobile number +393351234567.

```
DELETE /snew/v1/activitystreams/tel:+393351234567/@self/1234 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
Content-Length: 0
```

Response

This response provides back an acknowledgment that the activity was deleted.

```
HTTP/1.1 200 Ok
Date: Wed, 30 Dec 2011 12:00:00 GMT
Content-Length: 0
```

G.1.2 Content upload

The following is an example of a Photo Upload from an SNEW Client on behalf of user bob@example.com.

Request

This request uploads the activity specified in the “json” part with title “Partying from my favorite Club!”, which consists in a picture targeting a specific place (Bob’s favourite club). The location where the picture was taken is further indicated (which may differ from the place’s location or from Bob’s location at the time of the request).

The picture information in the activity object further contains a reference to the “image1@example.com” part of the request that contains the actual picture content, by the means of the “cid:” URI scheme.

Content-Type and Content-Length are set according to this multipart upload mechanism. Accept header is set to “application/json” as per this specification.

```
POST /snew/v1/activitystreams/acct:bob@example.com/@self HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
Accept: application/json
Content-type: multipart/related;
```

```
          boundary=abcdef012345xyZ;
          type="application/json"
Content-length: <contentLength>

--abcdef012345xyZ
Content-Type: application/json

{
    "actor": {
        "id": "acct:bob@example.com"
    },
    "object": {
        "objectType": "image",
        "url": "cid:image1@example.com",
        "location": {
            "position": "+48.52+002.20/"
        }
    },
    "title": "Partying from my favorite Club!",
    "target": {
        "objectType": "place",
        "id": "http: //example.com/places/5647"
    },
    "verb": "post"
}
--abcdef012345xyZ
Content-ID: <image1@example.com>
Content-Type: image/jpeg

<jpeg image data>
--abcdef012345xyZ--
```

Response

This response provides back the identifier of the newly created activity object (“1234”), together with its publication time. An identifier and URL for the uploaded picture is further provided back in relation to the picture.

As per this specification and REST principles a 201 Created response informs that the activity has been successfully created, further including the link to the newly created resource in the Location header.

```
HTTP/1.1 201 Created

Date: Tue, 29 Nov 2011 12:00:00 GMT

Location: http://example.com/snew/v1/activitystreams/acct:bob@example.com/@self/1234

Content-Length: 580

Content-Type: application/json


{
    "id": "http://example.com/activities/1234",
    "published": "2012-01-10T15:04:55Z",
    "actor": {
        "id": "acct:bob@example.com"
    },
    "object": {
        "objectType": "image",
        "id": "http://example.com/media/1234",
        "url": " http://example.com/bob/media/default/1234.jpg",
        "location": {
            "position": "+48.52+002.20/"
        }
    },
    "title": "Partying from my favorite Club!",
    "target": {
        "objectType": "place",
        "id": "http: //example.com/places/5647"
    },
    "verb": "post"
}
```

G.1.3 User Connection management using Gateway functionality

G.1.3.1 Retrieving the list of supported External SNs

Request

The following is an example of a request from an SNEW Client to retrieve the list of all the supported External SNs from SNEW Server.

```
GET /snew/v1/connections/@me/@all HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

Response

This response provides the list of supported External SNs to SNEW Client.

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "domain": "facebook.com",
        "org.oma":{
                "siteName": "Facebook",
                "siteIcon": "http://example.com/sites/logo/facebook.png"
        }
    },
    {
        "domain": "twitter.com",
        "org.oma":{
                "siteName": "Twitter",
                "siteIcon ": "http://example.com/sites/logo/twitter.png"
        }
    }
]
```

G.1.3.2 Retrieving the list of user's connections

Request

The following is an example of a request from an SNEW Client to retrieve the list of user's connections with External SN from SNEW Server.

```
GET /snew/v1/connections/@me/@self HTTP/1.1

Host: example.com

Authorization: Bearer mF_9.B5f-4.1JqM
```

Response

This response provides the list of user connections to SNEW Client. In this example the connection with Twitter is active, whilst the one with Facebook has expired.

```
HTTP/1.1 200 OK

Content-Type: application/json


[

{

        "domain": "facebook.com",

        "userId": "100003141581416",

        "userName": "Enginsaz Ozturksaz",

                "org.oma":{

                "siteName": "Facebook",

                "siteIcon ": "http://example.com/sites/logo/facebook.png"

                "connectionId": "12323123123123",

                "connectionStatus": "expired",

                "profileUrl": "http://www.facebook.com/100003141581416",

                "thumbnailUrl": "http://profile.ak.fbcdn.net/hprofile-ak-snc4/372458_1273808564_q.jpg"

                }

    },

{

        "domain": "twitter.com",

        "userId": "engin",

        "userName": "Engin Ozturk",

                 "org.oma":{

                "siteName": "Twitter",

                "siteIcon ": "http://example.com/sites/logo/twitter.png"

                                "connectionId": "3434343434",

                "profileUrl": "http://www.twitter.com/engin",
```

```
                "thumbnailUrl": "http://a0.twimg.com/profile_images/1115258974/1.PNG"
                }
    }
]
```

G.1.3.3 Deleting a user connection

Request

The following is an example of a request from an SNEW Client to delete a specific connection with an External SN.

```
DELETE /snew/v1/connections/@me/12323123123123 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

Response

This response acknowledges the successful deletion to SNEW Client.

```
HTTP/1.1 200 OK
```

G.1.3.4 Creating a user connection

Request

The following is an example of a request from an SNEW Client to create a new connection with the External SN "facebook.com".

```
POST /snew/v1/connections/@me/@self HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
Content-Type: application/json


{
    "domain": "facebook.com"
}
```

Response

This response acknowledges the successful creation to SNEW Client, once the association procedure has been completed upon user authorization.

```
HTTP/1.1 303 See other
Location: snew://connections#status=OK&connectionId=AF45Yz&domain=facebook.com
```

G.1.3.5 Updating a user connection

Request

The following is an example of a request from an SNEW Client to update the existing connection with connection id "12323123123123" to reactivate it.

```
PUT /snew/v1/connections/@me/12323123123123 HTTP/1.1

Host: example.com

Content-Type: application/json


{

    "org.oma":{

        "connectionStatus": "active"

    }

}
```

Response

This response acknowledges the successful update to SNEW Client, e.g. once the (re)association procedure has been completed upon user authorization.

```
HTTP/1.1 303 See other

Location: snew://connections#status=OK&connectionId=123231231231 23&domain=facebook.com
```

G.1.4 Cross-posting and aggregation using Gateway functionality

G.1.4.1 Retrieving activities and/or reactions

The following is an example of a request from an SNEW Client to retrieve all activities and/or reactions related to the user currently logged in, including those aggregated from his/her associated External SN connections.

Request

```
GET /snew/v1/activitystreams/@me/@all HTTP/1.1

Host: example.com

Authorization: Bearer mF_9.B5f-4.1JqM

Accept: application/json

Content-Length: 0
```

Response

This response provides back the list (as collection) of all the activities and reactions related to that user.

In this case the first activity corresponds to an activity generated by another local user (John Doe), which was also cross-posted to Facebook & Twitter and associated back as duplicates of the "local" activity, retrieved through connections 123 (linking the current user's Facebook account) and 34 (current user's Twitter account).

The second activity was aggregated from Twitter directly, retrieved only through connection 34 (linking the user's Twitter account).

The third activity is a "local" activity and does not come from External SN aggregation.

```
HTTP/1.1 200 Ok
Content-Length: ...
Content-Type: application/json


{
    "items": [
        {
            "id": "http://example.com/activity/4563",
            "published": "2011-02-10T15:04:55Z",
            "title": "This is John status update",
            "actor": {
                "id": "acct:johndoe@example.com",
                "image": {
                    "url": "http://example.com/johndoe/image"
                },
                "displayName": "John Doe"
            },
            "verb": "post",
            "location": {
                "position": "+48.52+002.20/"
            },
            "object": {
                "objectType": "note"
            },
           "downstreamDuplicates": [
                "http://facebook.com/4563",
                "http://twitter.com/statuses/4124"
           ],
            "org.oma":{
                "connections": [
                   "123",
                   "34"
                ]
            }
        },
        {
            "id": "http://twitter.com/statuses/4564",
```

```
            "published": "2011-02-10T15:25:55Z",
            "title": "This is Alice tweet",
            "actor": {
                "id": "acct:alice@twitter.com",
                "image": {
                    "url": "http://twitter.com/users/alice/photo"
                },
                "displayName": "Alice B. Smith"
            },
            "verb": "post",
            "location": {
                "position": "+48.52+002.20/"
            },
            "object": {
                "objectType": "note"
            },
            "org.oma":{
                "connections": [
                   "34"
                ]
            }
        },
        {
            "id": "http://example.com/activity/4565",
            "published": "2011-02-10T16:25:55Z",
            "actor": {
                "id": "acct:alice@example.com",
                "image": {
                    "url": "http://example.com/alice/image"
                },
                "displayName": "Alice Smith"
            },
            "object": {
                "objectType": "image",
                "url": "http://example.com/images/2741",
                "location": {
                    "position": "+48.00+002.00/"
```

```
                }
            },
            "title": "This is my new photo",
            "verb": "post"
        }
    ]
}
```

G.1.4.2 Cross-posting activities

The following is an example of a request from an SNEW Client to cross-post an activity on behalf of a user towards External SNs.

Request

This requests posts a status update with text “This is my new status update" from user with mobile number +393351234567, posted from location: lat: +48.52, lon: +2.2. This request further asks to cross-post this status update to connections 123 and 34 (e.g. the connections to the Facebook and Twitter accounts of the user)

```
POST /snew/v1/activitystreams/@me/@self HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
Content-Type: application/json
Accept: application/json
Content-Length: ...

{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my new status update",
    "verb": "post",
    "org.oma":{
        "connections": [
             "123",
             "34"
```

```
        ]
    }
}
```

Response

This response provides back the identifier of the newly created activity object ("1234"), together with its publication time. As per this specification and REST principles a 201 Created response informs that the activity has been successfully created, further including the link to the newly created resource in the Location header.

In this case the activity was successfully cross-posted only through connection 123 (e.g. current user's Facebook account), and the identifier of that activity on Facebook is provided as duplicate.

```
HTTP/1.1 201 Created
Date: Tue, 29 Nov 2011 12:00:00 GMT
Location: http://example.com/snew/v1/activitystreams/tel:+393351234567/@self/1234
Content-Length: ...
Content-Type: application/json


{
    "actor": {
        "id": "tel:+393351234567"
    },
    "object": {
        "objectType": "note"
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my new status update",
    "verb": "post",
    "id": "http://example.com/activities/1234",
    "downstreamDuplicates": [
        "http://facebook.com/4563"
    ],
    "org.oma":{
        "connections": [
             "123"
        ]
    }
}
```

G.1.5 People

G.1.5.1 Retrieving user information

The following is an example of a request from an SNEW Client to retrieve the information related to the user currently logged in, including those aggregated from his/her associated External SN connections.

Request

```
GET /snew/v1/people/@me/@self HTTP/1.1

Host: example.com

Authorization: Bearer mF_9.B5f-4.1JqM

Accept: application/json

Content-Length: 0
```

Response

This response provides back the information related to that user.

In this case two connections are returned, corresponding to the Facebook & Twitter accounts that are associated with the current user. The response also provides the number of followers, following and friends of that user.

```
HTTP/1.1 200 Ok

Content-Length: ...

Content-Type: application/json


{
    "id": "tel:+393351234567",
    "displayName" : "Tommy",
    "name": {
        "familyName":"Clinton",
        "givenName":"Tom"
    },
    "thumbnailUrl":" http://example.com/people/1234/image.jpg",
    "accounts" : [
        {
            "domain": "facebook.com",
            "userId": "1416",
            "username": "tclinton",
            "org.oma": {
                "connectionId": "12323123123123 ",
                "siteName": "Facebook",
                "siteIcon": "http://example.com/sites/logo/facebook.png",
                "displayName": "Tom Clinton",
```

```
                "profileUrl": "http://www.facebook.com/1416",
                "thumbnailUrl": "http://www.facebook.com/1234.jpg"
            }
        },
        {
            "domain": "twitter.com",
            "userId": "tommyc",
            "org.oma": {
                "connectionId": "3434343434",
                "siteName": "Twitter",
                "siteIcon": "http://example.com/sites/logo/twitter.png",
                "displayName": "TommyC",
                "profileUrl": "http://www.twitter.com/tommyc",
                "thumbnailUrl": "http://www.twitter.com/11.png"
            }
        }
    ],
    "following": {
        "totalItems": 10
    },
    "followers": {
        "totalItems": 21
    },
    "friends": {
        "totalItems": 4
    }
}
```

G.1.5.2 Searching user information

The following is an example of a request from an SNEW Client to search for a user on the same SN whose name starts with "John".

Request

```
GET /snew/v1/people/@me/@all?filterBy=displayName&filterValue=John HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
Accept: application/json
Content-Length: 0
```

Response

This response provides back the list of users matching this criterion.

```
HTTP/1.1 200 Ok
Content-Length: ...
Content-Type: application/json


{
    "items": [
        {
            "id": "acct:johnny@example.com",
            "displayName" : "Johnny",
            "thumbnailUrl":" http://example.com/people/1234/image.jpg"
        },
        {
            "id": "acct:bobjohn.acme@example.com",
            "displayName" : "Bob John Acme",
            "name": {
                "familyName":"Acme",
                "givenName":"Bob John"
            },
            "thumbnailUrl":" http://example.com/people/2222/image.jpg"
        }
    ]
}
```

G.1.5.3 Retrieving another user information

The following is an example of a request from an SNEW Client to retrieve the information related to another user, further indicating his relationship with the user currently logged in. Note that this request can be issued when discovering address book contacts and that the target user may belong to a remote social network.

Request

```
GET /snew/v1/people/tel:+393351234567/@self HTTP/1.1

Host: example.com

Authorization: Bearer mF_9.B5f-4.1JqM

Accept: application/json

Content-Length: 0
```

Response

This response provides back the information related to that user. In this case the current user is already followed by the target user.

The response also provides the number of followers, following and friends of that user.

```
HTTP/1.1 200 Ok

Content-Length: ...

Content-Type: application/json


{
    "id": "tel:+393351234567",
    "displayName" : "Tommy",
    "name": {
        "familyName" : "Clinton",
        "givenName" : "Tom"
    },
    "thumbnailUrl": "http://example.com/people/1234/image.jpg",
    "relationships": "followers",
    "following": {
        "totalItems": 10
    },
    "followers": {
        "totalItems": 21
    },
    "friends": {
        "totalItems": 4
    }
}
```

G.1.6 Subscription

The following is an example of a subscription from an SNEW Client to all activities related to the user with mobile number +393351234567.

Request-1

This request is a normal retrieval request as in example G.1.1.1, where the SNEW Client further indicates support for subscriptions by adding an Accept header value “text/event-stream”.

```
GET /snew/v1/activitystreams/tel:+393351234567/@all HTTP/1.1
Host: example.com
Accept: application/json
Accept: text/event-stream
Content-Length: 0
```

Response-1

This response provides back the list (as collection) of all the activities related to that user (as in example G.1.1.1) and further includes indications on where to subscribe to the related information as Link: headers.

```
HTTP/1.1 200 Ok
Link: <http://hub.example.com>; rel="hub"
Link: <http://example.com/snew/v1/activitystreams/tel:+393351234567/@all>; rel="self"
Content-Length: ...
Content-Type: application/json

{
    "items": [
        {
            "id": "http://example.com/activity/4563",
            "published": "2011-02-10T15:04:55Z",
            "title": "This is John status update",
            "actor": {
                "id": "acct:johndoe@example.com",
                "image": {
                    "url": "http://example.com/johndoe/image"
                },
                "displayName": "John Doe"
            },
            "verb": "post",
            "location": {
                "position": "+48.52+002.20/"
            },
```

```
            "object": {
                "objectType": "note"
            }
        },
        {
            "id": "http://example.com/activity/4564",
            "published": "2011-02-10T15:25:55Z",
            "title": "This is Alice tweet",
            "actor": {
                "id": "acct:alice@example.com",
                "image": {
                    "url": "http://example.com/alice/image"
                },
                "displayName": "Alice Smith"
            },
            "verb": "post",
            "location": {
                "position": "+48.52+002.20/"
            },
            "object": {
                "objectType": "note"
            }
        }
    ]
}
```

Request-2

This second request is issued by the SNEW Client to initiate the subscription using the hub & topic values indicated in Response-1.

In case a local Push API Server exists on the device, the request is issued indirectly through that Push API Server.

```
GET http://localhost:4035?push-accept-source=
http%3A%2F%2Fhub.example.com%3Fhub.mode%3Dsubscribe%26hub.topic%3Dhttp%3A%2F%2Fexample.com%2Fsnew%2Fv1%2
Factivitystreams%2Ftel%3A%2B93351234567%2F%40all HTTP/1.1

Host: localhost

Accept: text/event-stream
```

Alternatively, this request is issued directly to the hub endpoint.

```
GET /?hub.mode=subscribe&hub.topic=http://example.com/snew/v1/activitystreams/tel:+393351234567/@all
HTTP/1.1

Host: hub.example.com
```

```
Accept: text/event-stream
```

Response-2

This second response represents a long-lived connection that outputs subsequent notifications of activities related to that user whenever applicable. As such, headers are provided first

```
HTTP/1.1 200 Ok

Content-Type: text/event-stream
```

Then notifications of activity entries serialized in JSON are provided as single line text as below.

```
{"id": "http://example.com/activity/4565", "published": "2011-02-10T15:25:55Z", "actor": { "id":
"acct:alice@example.com", "image": { "url": "http://example.com/alice/image"}, "displayName": "Alice
Smith"}, "object": { "objectType": "image", "url": "http://example.com/images/2741", "location": {
"position": "+48.52+002.20/"}}, "title": "This is my new photo", "verb": "post"}
```

Appendix H. Push-specific headers definition

This appendix defines a set of headers to be used for subscriptions over SNEW-1 interface, as defined in section 7.1.5.

H.1 Headers definition

H.1.1 X-Oma-Push-Accept-Source

This header indicates a comma-separated list of acceptable push sources (Push channels) supported by the entity inserting the header. This header is specifically used for negotiating the Push channel(s) to be used between client & server.

Each push source MUST be identified by one of the following form:

- a SMS source addresses in the format “sms:sms-recipient” where sms-recipient is as defined by [RFC5724], which indicates a support for delivery of events from a specific SMS address
- a SIP source addresses in the format “sip:user@domain”, which indicates a support for delivery of events from a specific SIP address
- the OMNA-registered URN “urn:oma:xml:push”, which indicates a support for delivery of any OMA Push message received from the supported OMA Push bearers
- other arbitrary source address values in the form of a URI, enabling the use of other eventing or messaging systems (e.g. OS-specific) or application-specific source addressing

The ABNF [RFC5234] format is:

```
X-Oma-Push-Accept-Source = "X-Oma-Push-Accept-Source" ":" absolute-URI *("," absolute-URI)
      ; absolute-URI is as defined in [RFC3986]
```

H.1.2 X-Oma-Push-Application-Id

This header indicates a comma-separated list of acceptable push application ids.

The ABNF [RFC5234] format is:

```
X-Oma-Push-Application-Id = "X-Oma-Push-Application-Id" ":" app-id *("," app-id)
      ; app-id is as defined in [OMAPUSH-MSG]
```

H.1.3 X-Oma-Push-Server-Address

This header indicates a comma-separated list of acceptable IP addresses or FQDNs of the servers issuing notifications. This header is specifically used for client-side configuration of filters of incoming notifications.

The ABNF [RFC5234] format is:

```
X-Oma-Push-Server-Address = "X-Oma-Push-Server-Address" ":" ppg-specifier *("," ppg-specifier)
      ; ppg-specifier is as defined in [OMAPUSH-PPG]
```

H.2 Headers usage

The above defined headers are intended to be used to negotiate a Push channel to be used for sending notifications from server to client.

The following example illustrates a typical usage of such headers for negotiating the push channels to convey notifications.

Request

When issuing a subscription request, the client inserts the X-Oma-Push-Accept-Source header to indicate the list of Push channels over which it can accept notifications.

```
GET /myapp/subscribe/data HTTP/1.1

Host: example.com

X-Oma-Push-Accept-Source: urn:oma:xml:push, sms:1234, urn:example:push:apns, urn:example:push:c2dm
```

Response

According to the headers received in the response, the client opens one or more push channels and prepare the related filters.

In this case, the client will only start listening for incoming messages over OMA Push and the proprietary “c2dm” technology, only allowing notifications targeting the “myapp” application and sent from the “ppg.example.com” server.

```
HTTP/1.1 200 Ok

X-Oma-Push-Accept-Source: urn:oma:xml:push, urn.example:push:c2dm

X-Oma-Push-Application-Id: myapp

X-Oma-Push-Server-Address: ppg.example.com
```

Appendix I. Cross domain example flows (Informative)

I.1 Cross domain following

The following is an example flow of a user who requests to follow a user belonging to another OMA Compliant SN.

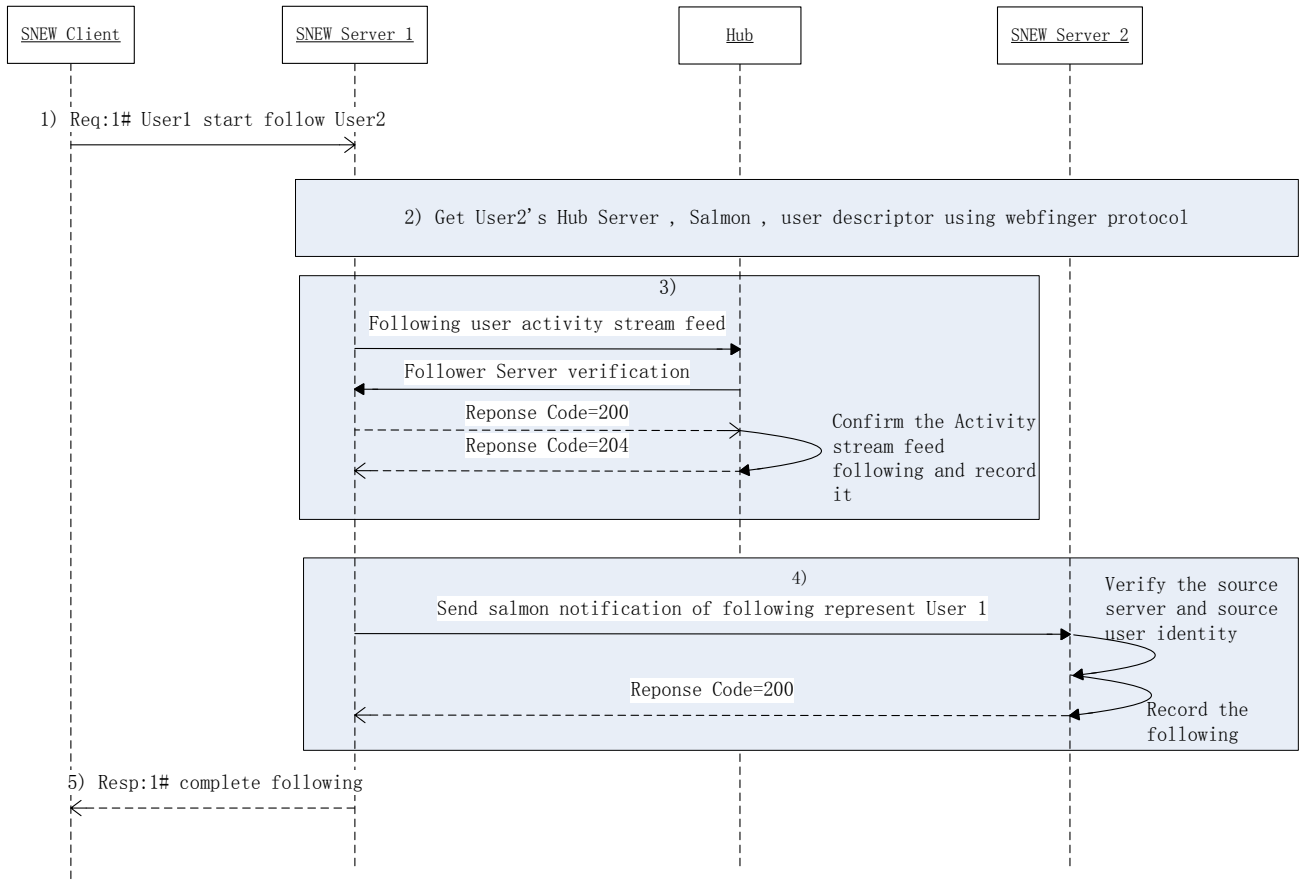

Figure 5: Example flow of cross domain subscription

1) User1 uses the SNEW Client to start follow User2 belonging to another OMA compliant SN through SNEW-1 interface, the request is sent from SNEW Client to SNEW Server 1.
2) SNEW Server 1 as the host server of User1 performs a Webfinger query, getting User2's Hub Server, Salmon and user descriptor
3) Once retrieved and validated the user descriptor, after checking that user activity stream feed hasn't been subscribed and the feed contains a hub relation link, the SNEW Server 1 subscribes to the feed based on [PubSubHubbub].
4) Based on [Salmon-Protocol], SNEW Server 1 sends Salmon notification to SNEW Server 2 about User1 following, SNEW Server 2 verifies the source server and source user identity and records the following information.
5) After successful subscription, SNEW Server 1 sends the success message to SNEW Client.

I.2 Cross domain activities dispatching

The following is an example flow of getting the activities from another OMA Compliant SN. The previous condition is User1 on SNEW Server 1 had successfully subscribed User2's activities (see X.1).

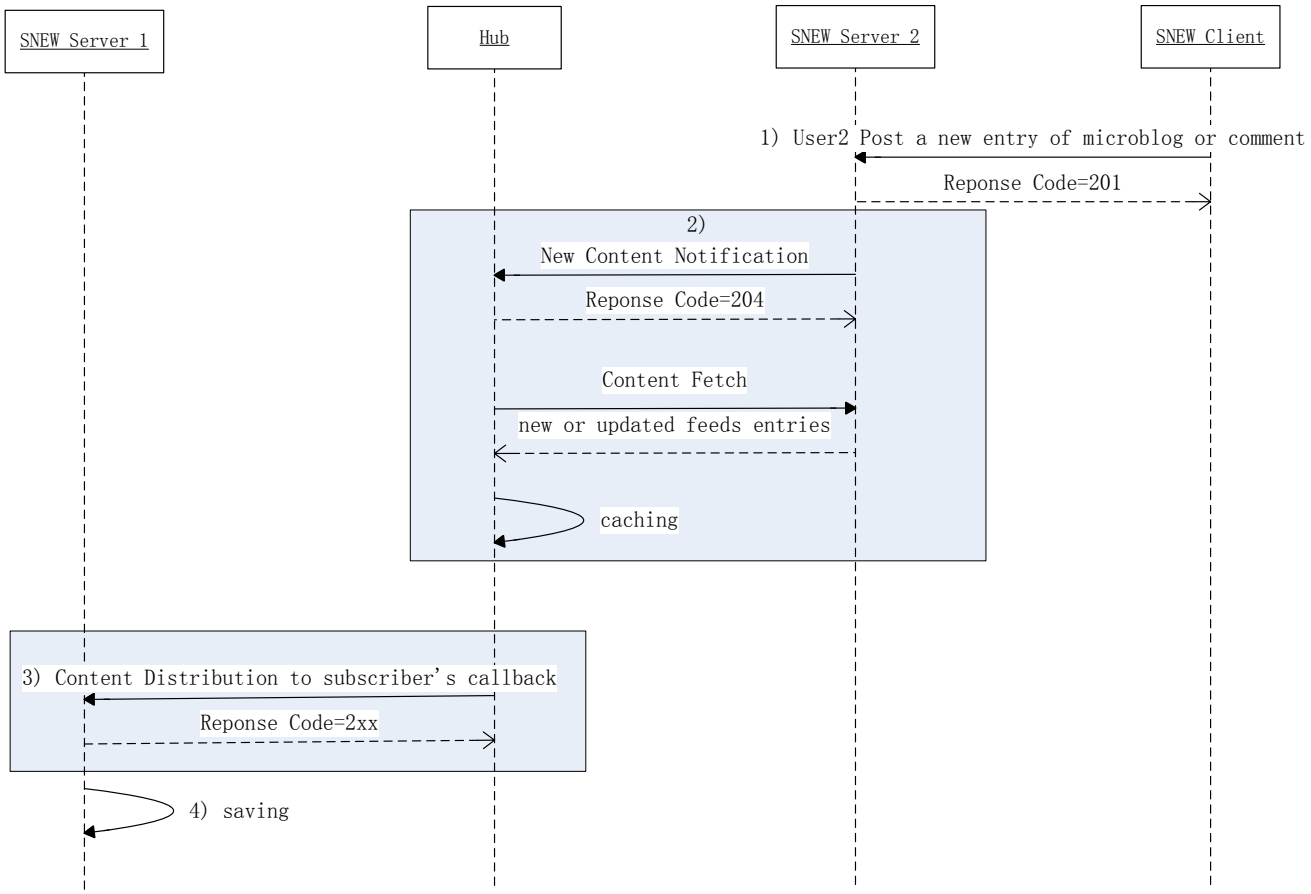

Figure 6: Example flow of cross domain activities dispatching

1) User2 uses SNEW Client to publish an activity in SNEW Server 2, e.g.: post a new entry of microblog.
2) SNEW Server 2 checks the target audience for this activity and identifies that User1 is a follower belonging to another SN domain (SNEW Server 1). As this activity is public it, use PubSubHubbub notification method to send the activity information to the Hub.
3) Since User1 in SNEW Server 1 had subscribed to User2 activity stream feed, SNEW Server 1 receives the notification about User2 from the Hub by PubSubHubbub callback mechanism.
4) SNEW Server 1 saves the receiving activity from User2. Optionally, SNEW Server 1 can notify the SNEW client of User1 that a new activity was published by User2.

I.3 Cross domain reactions

The following is an example flow of a user to comment an activity in another OMA Compliant SN. The previous condition is User1 on SNEW Server 1 received User2's activities in SNEW Server 2 (by push notification or retrieval).

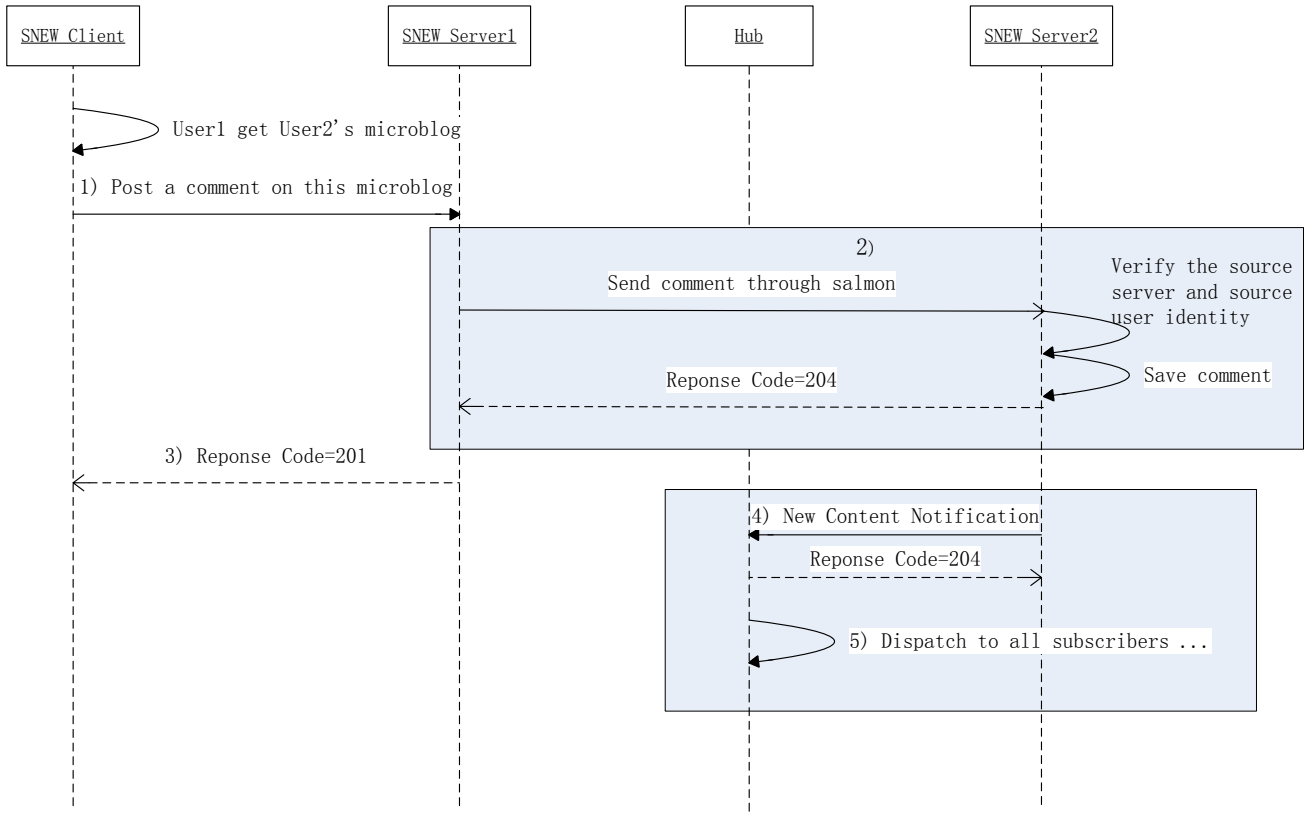

Figure 7: Example flow of cross domain reactions

1) User1 comments on User2's microblog, SNEW Client post the comment to SNEW Server 1.
2) User2 is a remote user in SNEW Server 2 and the comment needs be propagated towards User2's microblog: SNEW Server 1 transcodes the received JSON data to ATOM data; based on [Salmon-Protocol] SNEW Server 1 sends Salmon notification to SNEW Server 2 about the comment, SNEW Server 2 verifies the source server and source user identity and records the comment.
3) The success response is sent back to SNEW Client 1 through SNEW Server 1.
4) Since there may be other subscribers in other SN domain, SNEW Server 2 uses PubSubHubbub notification method to send the reaction information to the Hub.
5) Hub dispatches the information to all subscribers.

Appendix J. Autoconfiguration of mobile devices (Informative)

The following sequence illustrates an example of how a SNEW Client can self-configure on the device to obtain the OAuth2 token needed for SNEW-1 interface, and the related endpoint, on behalf of an application.

J.1 Domain/server discovery

At first start (or as long as no token exists for this application), the SNEW Client needs to identify the authoritative domain and/or SNEW Server for the user account. Whilst this information could be pre-provisioned, this process could be achieved in the following alternative modes:

1. by offering the user a selection of predefined domains/providers,
2. by prompting the user for his/her identity (e.g. user@domain),
3. by automatically identifying the server address based on mobile network operator information, in case of MSISDN-based identification of the user.

In scenario 1, each domain/provider could easily be associated with a predefined server, which can be used as input to the next step.

In scenario 2, note that the domain-only part of the identity is sufficient for the purposes of domain discovery, so it may be queried instead of the entire identity, also for privacy purposes. In this case the domain can be used as input to the next step.

In scenario 3, it is assumed that the network can implement the necessary procedures to resolve the user's MSISDN and that the SNEW Client can access the home operator's Mobile Country Code (MCC) and Mobile Network Code (MNC) of the SIM currently used on the device. The server address to be used as input to the next step will be composed based on this information (e.g. "snew.mnc<MNC>.mcc<MCC>.pub.3gppnetwork.org") and as such will only be routable within the operator's network.

Note that this addressing can be reused for routing of SNeW requests across operators.

J.2 Endpoint discovery and configuration

Once the SNEW Client has identified a domain and/or server address related to the user:

- the SNEW Client performs host-meta discovery to discover three (3) links:
    1. the "oauth2-authorize" link relation type,
    2. the "oauth2-token" link relation type, and
    3. the "http://ns.opensocial.org/2008/opensocial" link relation type

The process (similarly to the one described in section 8.2.2 for SNEW Servers) is as follows:

- If the server address was identified based on network-based information (scenario 3), to ensure that MSISDN resolution procedures (e.g. RADIUS requests, header enrichment, IP-to-CLI, etc) can be applied, the SNEW Client performs an HTTP request to http://<DOMAIN>/.well-known/host-meta at first.
- Otherwise, or if the above step was performed and answered successfully, the SNEW Client performs an HTTPS request to https://<DOMAIN>/.well-known/host-meta

  Note that in case of scenario 3, the server should be able to correlate both the initial HTTP and the second HTTPS requests on the server side. In order to achieve this, the server can provide a cookie in the response to the initial HTTP request (Set-Cookie header) and expects to receive that cookie from the client in the subsequent HTTPS request (Cookie header). From the client perspective, the second request is towards the same Uniform Resource Locator (URL) as the first one with only the protocol change.

- The SNEW Client then opens the device's native browser on the link #1 to start the authorization process for that application according to OAuth2 and obtain an authorization code. This process may ask the user to authenticate himself or may automatically authenticate him e.g. based on mobile network information. Once the user has authorized the application the redirect URI allows to return back from the browser to the SNEW Client.
- Once the authorization code is obtained, the SNEW Client issues the token request to link #2 according to OAuth2 and obtains the related token for that application
- The application can now open regularly and use the SNEW Client to access SNEW-1 interface using base URL #3.

J.3 Account provisioning

J.3.1 Using authentic network information

Once the SNEW Client has obtained base URL #3 according to section J.2, it can attempt an account creation request according to section 7.1.1. Depending on SNEW Server policies this request may be accepted as a self-provisioning request by creating an account for the user identified by authentic network information. In that case the SNEW Server can consider the tel: URI based on the user's MSISDN as "id" of the newly created account, or any other URI that uniquely identifies that user (e.g. based on a hash, etc).

J.3.2 Using External SN account

When the SNEW Server supports the Gateway functionality, it can decide to allow users to create themselves an account based on their External SN account.

In this case, once the SNEW Client has performed host-meta discovery according to section J.2, it can issue a request to the SNEW Server via SNEW-1 interface using base URL #3 to discover the External SN connections supported by the SNEW Server according to section 7.1.4.2, if any.

The SNEW Client then provides the list of supported External SNs to the user, who selects the one to be used for creating an account on the SNEW Server. Based on the selected "domain" of the connection, the SNEW Client inserts this value as "provider" parameter in the "oauth2-authorize" request to link #1 (as per section 10.2.2) to signify the intention of the user to create a local account based on their External SN account. The SNEW Server will activate the procedure(s) to allow the user to connect to the specified External SN and use their account as basis for local account creation on that SNEW Server.

Appendix K. RFC6415-based resource discovery (Informative)

The following illustrates a resource (e.g. user) discovery process similar in principle to WebFinger [WebFingerProtocol], using an alternative solution based on [HOST-META] and the "lrdd" link relation. Such approach is widely used by OStatus-compliant SNs and as such may be of interest for SNEW Servers that intend to federate with such SNs.

The SNEW Server first discovers the Host-Meta descriptor of the domain name part of the user identity, unless previously cached, according to [HOST-META] with the following clarifications:

1. Issue the Host-Meta discovery request over HTTPS. If no response is provided, the SNEW Server can send the same request over HTTP.

2. Verify the Host-Meta signature according to [XRD] section 5, in case the descriptor was not retrieved through HTTPS. If the signature is expected but not found or invalid, the process is aborted.

3. Verify that the Host-Meta *Host* element matches the domain name part of the user's URI, if applicable. If the element is invalid, the process is aborted.

4. Read the lrdd relation link template URL from the Host-Meta and send a request to that URL substituting the {uri} parameter with the user's URI

5. Verify the signature of the newly retrieved descriptor according to [XRD] section 5, in case it was not retrieved through HTTPS. If the signature is expected but not found or invalid, the process is aborted.

6. Verify that the *Subject* element of the newly retrieved descriptor matches the URI of the user. If the element is not found or invalid, the process is aborted.

Appendix L. Addressing EU Data Protection principles (Informative)

This appendix describes how the SNEW Enabler addresses EU principles in terms of data protection of Social Network Services (SNS) [EUOpinionSN] from a technical standpoint.

It is envisioned that SNEW Enabler may be deployed in countries outside of the EU; as such these principles may not apply in those countries, notwithstanding the fact the EC directive also applies to SNS operating outside of the EU that target EU citizens as their customers.

It is also worth distinguishing SNS that provide the actual service from the SNEW Enabler defined in this specification, that provides the technical foundation, obligations and recommendations to operate a SNS. In that sense, this appendix informs about how some of the principles have been considered and addressed from a technical standpoint, whilst others may be up to the actual implementation and/or deployment to be addressed.

Furthermore, it has to be noted that the EU data protection directive is currently under revision and will be replaced by a framework consisting of a regulation (replacing Directive 95/46/EC) setting out a general EU framework for data protection and a directive setting out rules on the protection of personal data processed for the purposes of prevention, detection, investigation or prosecution of criminal offences and related judicial activities (used for fight against terrorism). The reform of the EU data protection rules will result in

- Improving Individuals' ability to control their data
- Improving the means for individuals to exercise their rights
- Reinforcing data security
- Enhancing the accountability of those processing data

L.1 Application of the Data Protection Directive

L.1.1 Default privacy settings

According to [EUOpinionSN] sections 3.1 & 3.2, SNS should offer privacy-friendly default settings for accessing user data, i.e. by restricting access to profile content (in a wide sense thus including profile information, activities and content) and by making users not searchable by default.

In accordance with the EU directive, section 7.1.1.4 of this specification recommends that users are not searchable unless they give explicit consent. Furthermore, section 7.1.2 of this specification also recommends to restrict default privacy of user information "only to the users, whom the actor has explicitly approved relationships."In this case, the concept of "self-selected" contacts has been translated technically into "explicitly approved relationships" as a way of identifying other users that the reference user has explicitly decided to identify as contact.

L.1.2 Sensitive data

According to [EUOpinionSN] section 3.4, SNS may not require any sensitive data from their users and may publish or process sensitive personal data of their users only with their explicit consent.

This specification does not mandate any sensitive data to be provided by users. However, the user profile data model referenced in section 9.3 of this specification does allow to represent such information, if supported by the SNS and provided explicitly by the user. In accordance with the EU directive, this specification further clarifies that access to this information may not be provided without the explicit user consent.

L.1.3 Access by third parties

According to [EUOpinionSN] sections 3.6.2, SNS should provide adequate levels of granularity to APIs accessing user data, i.e. by restricting access to content to the minimum sufficient level.

Section 10.2 of this specification defines a technical authorization mechanism based on OAuth2 with a predefined set of "scopes" that allows users to provide read and/or write access to any specific set of information (e.g. profile, activities, etc).

This level of granularity has been considered an adequate balance between a coarse-grained approach (all-read / all-write) and a fine-grained approach (e.g. controlling access to each single attribute of information)

L.1.4 Retention of data

According to [EUOpinionSN] sections 3.8, SNS must inform the user about the different purposes for which they retain their data after ban, deletion or update (retaining the previous data). In general SNS should delete any user-related information upon user request for account deletion, or automatically after a period of inactivity.

In accordance with the EU directive, section 7.1.1.3 of this specification requires SNEW Servers to delete user-related information upon account deletion (unless for legitimate reasons, e.g. security or legal). This section further mentions the possibility for SNS relying on SNEW to apply procedures to automatically deactivate and/or delete accounts after a defined period of time.

L.1.5 Information on data processing and rights

According to [EUOpinionSN] sections 3.3 & 3.9, SNS should inform the user about the different purposes for which they process personal data, typically via reminders and warnings, as well as on the existence of a “complaint handling office” and the obligation of providing their real identity, if applicable.

Such principles have not been translated technically into this specification as it is expected that SNS provide such awareness through their website and/or application via their own means.

L.1.6 Children and minors

According to [EUOpinionSN] section 4, SNS should take appropriate actions to create awareness and protect the interests of children using their service. It is recommended to implement technologies such as age verification, warning boxes and self-regulation mechanisms.

Such principles have not been translated technically into this specification as it is expected that SNS provide such awareness through their website and/or application via their own means. However, sections 7.1 & 9.2.4.1 of this specification have defined an abuse reporting functionality as an optional functionality that can be used to report content that is inappropriate for children.

L.2 Safer Social Networking Principles

The European Commission has defined a set of principles for SNS [EUSNPrinciples] subject to self-regulation, i.e. for SNS to submit a self-declaration form explaining how they address these principles in their SNS. Assessment are performed periodically to monitor the implementation of these principles.

L.3 Other Principles

The European Commission has launched a data protection reform that will affect SNS [EUDataProtection], which focus mostly of the “right to be forgotten”, “privacy by default” and “complete control over their data” that are somehow addressed in the previous section. However, some other principles are mentioned that would benefit the user experience, such as “data portability”.

L.3.1 Data Portability

In [EUDataProtection] the European Commission has also been advocating for data portability (aka “data transfer”) across providers in the era of Internet services.

In that sense, this specification has defined an archive format for exporting/importing user-related information (including profile, contacts, activities & content) across SNEW-compliant SNS. Such functionality, named “social data portability” is defined in section 7.2.3 and the import procedure is defined in 8.2.8. It is expected that future releases of this specifications will further specify both the export and import functionalities.