Technical Specification: WhatsApp JID Systems (PN and LID)

1. Introduction

1.1. Purpose

This document provides a detailed, language-agnostic specification for the Jabber Identifier (JID) system used in WhatsApp. It covers the legacy Phone Number (PN) based identification system and the new privacy-preserving Lightweight Identity (LID) system.

1.2. Scope

The specification details the structure, function, and interoperability of both PN and LID JIDs. It is intended to serve as a foundational document for developers implementing or interacting with the WhatsApp protocol, ensuring a clear understanding of user identification and routing. A key focus is on maintaining retrocompatibility between clients that are LID-aware and those that are not.

1.3. Terminology

- **JID:** Jabber Identifier. The unique address for an entity on the network (e.g., a user, a group).
- **PN:** Phone Number. The legacy identifier for a user, based on their MSISDN.
- **LID:** Lightweight Identity. The new, privacy-preserving identifier that is not publicly tied to a user's phone number.
- **Client:** A WhatsApp application instance running on a device.
- **Stanza:** An XML element that is the fundamental unit of communication in XMPP (e.g., <message>, <iq>).

2. JID Fundamentals

A WhatsApp JID follows the standard XMPP format, which can be broken down into three main parts: local-part@domain-part/resource-part.

- **Local Part (User):** Uniquely identifies the user or entity within a domain. This is the core component that differs between the PN and LID systems.
- **Domain Part (Server):** Specifies the server or service handling the JID. Examples include s.whatsapp.net for users, g.us for groups.
- **Resource Part (Device):** An optional component that identifies a specific device or session for a user. In the multi-device context, this is crucial for routing messages to the correct client.

Standard Format: user@server

Multi-Device Format: user:device@server or user.device@server (legacy)

3. The Legacy System: Phone Number (PN) JID

The original identification system is directly based on the user's phone number.

3.1. Structure

- **User:** The user's international phone number (MSISDN), excluding any non-numeric characters like +, -, or spaces.
- **Server:** s.whatsapp.net for standard user-to-user communication.
- **Device:** For the primary device, the device ID was often omitted or specified as 0. For companion devices, a positive integer identifies the specific device.

3.2. Examples

- **User JID:** 5511999998888@s.whatsapp.net
- **Primary Device (Legacy):** 5511999998888.0@s.whatsapp.net
- **Companion Device:** 5511999998888:5@s.whatsapp.net

3.3. Other Common Servers

- **g.us:** Group chats.
- **broadcast:** Broadcast lists.

3.4. Limitations

The primary drawback of the PN system is privacy. A user's phone number is exposed to anyone they interact with, as it is embedded in the protocol-level JID.

4. The New System: Lightweight Identity (LID) JID

The LID system was introduced to enhance user privacy by decoupling the protocol-level identifier from the user's phone number.

4.1. Motivation

By using a non-reversible numerical identifier, WhatsApp can route messages without revealing the phone numbers of the sender or recipient to other parties in the conversation (especially in groups).

4.2. Structure

- **User:** A unique, large integer assigned to the user account. This number is not algorithmically derived from the phone number.
- **Server:** lid.
- **Device:** Same as the PN system; a positive integer identifying a specific companion device.

4.3. Examples

- **LID User JID:** 123456789012345678@lid
- **LID Companion Device:** 123456789012345678:5@lid

4.4. The LID-PN Mapping

A critical component of the LID system is the mapping between a user's permanent PN and their assigned LID.

- **Storage:** This mapping is maintained by WhatsApp servers. Clients should also maintain a local cache of these mappings for known contacts to reduce server lookups and enable offline resolution. The whatsmeow source code reveals a whatsmeow_lid_map table for this purpose (lid TEXT PRIMARY KEY, pn TEXT UNIQUE NOT NULL).
- **Discovery:** Clients learn about LID-PN mappings through various protocol flows:
  - **History Sync:** The history.pdf documentation and whatsmeow code show that history sync payloads contain PhoneNumberToLidMappings.
  - **Message Stanzas:** Incoming messages may include attributes that provide the sender's alternative JID (e.g., a message from a PN JID might include a participant_lid attribute).
  - **Notifications:** LidChangeNotification events are pushed to clients when contacts migrate.

5. Retrocompatibility and Interoperability

To ensure a seamless transition, the system must handle communication between LID-aware and legacy PN-only clients.

5.1. Addressing Mode

Modern message stanzas include an addressing_mode attribute. This attribute explicitly tells the client how to interpret the JIDs in the from, to, and participant attributes.

- addressing_mode="lid": Indicates the JIDs are LIDs.
- addressing_mode="pn": Indicates the JIDs are PNs.

This attribute is essential for a client to know which identifier to use for cryptographic operations (Signal Protocol) and contact resolution.

5.2. Message Routing Logic (Server-Side)

- **PN Client -> LID Client:** The PN client sends a message to a PN JID. The server performs a lookup, finds the corresponding LID for the recipient, and routes the message to the recipient's devices, which are logged in with their LID. The incoming stanza on the LID client will likely have attributes mapping the sender's PN to their LID.
- **LID Client -> PN Client:** The LID client sends a message to a LID JID. The server performs a lookup. If the recipient device is a legacy client, the server translates the

sender's LID back to a PN and delivers the stanza as if it came from a PN JID.
- **LID Client -> LID Client:** Communication proceeds using LIDs for both sender and recipient.

5.3. Cryptographic Sessions (Signal Protocol)

To preserve privacy, all end-to-end encryption sessions must be established using the LID-based address, not the PN-based one.

- **Session Key:** The unique identifier for a Signal Protocol session is LID_User:Device_ID (e.g., 123456789012345678:5).
- **Migration:** When a client first learns the LID for a contact, it must migrate any existing Signal sessions. The whatsmeow source code includes a MigratePNToLID function for this purpose. This involves moving session state stored under the PN-based address to be stored under the new LID-based address. Failing to do so would force a new key exchange, causing a "safety number change" notification.

6. Implementation Proposal for Retrocompatibility (Language-Agnostic)

A client implementation must adhere to the following principles to support both systems.

6.1. JID Parsing and Handling

- The JID parsing logic must be robust enough to handle both user@s.whatsapp.net and user@lid formats, correctly identifying the server type.
- The internal JID representation should have a field to distinguish between PN and LID types, or rely on the server domain.

6.2. Local LID-PN Cache

- Implement a persistent key-value store to cache LID-PN mappings.
- Provide functions to query the cache in both directions: GetLIDForPN(pn) and GetPNForLID(lid).
- The cache should be updated whenever new mappings are discovered from message stanzas or history sync.

6.3. Sending Messages

1. **Recipient Resolution:** Before sending a message, resolve the recipient's JID. Prefer using the LID if it is present in the local cache. If not, fall back to the PN.
2. **Stanza Construction:**
    - Set the to attribute of the <message> stanza to the resolved JID (LID if available, otherwise PN).
    - If using a LID, explicitly include the addressing_mode="lid" attribute.
3. **Encryption:** Always use the LID-based address for Signal Protocol encryption if the

recipient's LID is known. This is critical. If only the PN is known, use the PN-based address, but expect to migrate the session once the LID is discovered.

6.4. Receiving Messages

1. **Identify Sender:** When a message is received, parse the from and participant JIDs. Check the addressing_mode attribute to determine if they are LIDs or PNs.
2. **Update Cache:** If the stanza contains mapping attributes (e.g., participant_pn alongside a LID participant), update the local LID-PN cache with this new information.
3. **Contact Resolution:** Use the received JID (PN or LID) to look up the contact in the local address book. The application should be able to resolve a contact from either their PN or LID.
4. **Decryption:** Use the JID provided in the stanza (respecting the addressing_mode) to identify the correct Signal Protocol session for decryption.

6.5. Contact and Session Management

- **Stable ID:** The PN should be treated as the user's stable, primary identifier within the application's data models, as LIDs could potentially change (though this seems unlikely). The LID should be stored as an associated, protocol-level identifier.
- **Session Migration:** Implement a migration path for Signal sessions as described in section 5.3. When a LID is learned for a PN that has an active session, atomically rename the session's key from the PN address to the LID address.