WhatsApp Cloud API & MM Lite API Report for WhatsWay

Executive Summary

This report provides comprehensive information about WhatsApp Cloud API and MM Lite API for the WhatsWay project, including the latest version information, API usage guidelines, webhook implementation, and key considerations for building a WhatsApp Marketing PaaS platform.

---

1. Latest Version Information

WhatsApp Cloud API Status

- **Current Status**: WhatsApp Cloud API is now the primary API offering from Meta
- **On-Premises API Sunset**:
  - No new signups allowed after July 1, 2024
  - Complete deprecation on **October 25, 2025**
  - All businesses must migrate to Cloud API before this date
- **Version**: The API follows Meta's Graph API versioning (e.g., v16.0, v17.0, v18.0)
- **Updates**: Automatic monthly updates rolled out by Meta

Key Timeline

- **January 2024**: Meta began the On-Premises API sunset process
- **July 2024**: No new On-Premises API applications accepted
- **October 2025**: Complete shutdown of On-Premises API
- **2025 onwards**: Cloud API becomes the only supported option

---

2. WhatsApp Cloud API Overview

What is Cloud API?

WhatsApp Cloud API is Meta's cloud-hosted version of the WhatsApp Business API. It eliminates the need for businesses to host their own infrastructure while providing all the messaging capabilities of the traditional API.

Key Features

1. **No Infrastructure Required**: Hosted entirely on Meta's servers
2. **Quick Setup**: Can be operational within an hour
3. **Automatic Updates**: New features and updates rolled out automatically
4. **Scalable**: Handles from 100 to 10,000+ messages without infrastructure changes

5. **Free Access**: No cost to access the API (pay only for conversations)

Technical Specifications

- **Message Throughput**: Up to 250 messages per second (combined sending/receiving)
- **Message Types Supported**:
  - Template messages (marketing, utility, authentication)
  - Session messages (24-hour window)
  - Media messages (images, documents, videos, audio)
  - Interactive messages (buttons, lists, product catalogs)
- **Conversation Pricing**: Billed per 24-hour conversation window
- **Multiple Phone Numbers**: Support for multiple WhatsApp numbers per account

API Endpoints

- Base URL: `https://graph.facebook.com/{version}/{phone-number-id}/messages`
- Authentication: Bearer token (Access Token)
- Message Format: JSON payload with template or text message

---

3. MM Lite API (Marketing Messages Lite)

Overview

MM Lite API is Meta's specialized solution for sending marketing messages at scale. It's designed specifically for outbound promotional messaging with enhanced delivery optimization.

Key Differences from Cloud API

Feature	Cloud API	MM Lite API
Purpose	Two-way communication	Outbound marketing only
Message Types	All types	Marketing templates only
Delivery Optimization	Standard	AI-optimized delivery
Setup Complexity	Moderate	Minimal
Metrics	Standard	Enhanced marketing insights
Regional Availability	Global	Limited (expanding)

MM Lite Features

1. **Optimized Delivery**: Up to 9% higher delivery rates for engaged audiences
2. **AI-Powered Timing**: Messages sent when users are most likely to engage
3. **Performance Benchmarks**: Compare your campaigns against regional averages

4. **Advanced Analytics**: Detailed engagement metrics and conversion tracking
5. **Same Templates**: Use existing marketing templates without modification

Current Availability

- **Available Regions** (as of 2025):
  - India
  - Brazil
  - Mexico
  - Indonesia
  - Chile
  - Peru
- **Global Rollout**: Expected by end of Q1 2025
- **US Restriction**: Marketing messages to US users not delivered (error code 131049)

Implementation Requirements

1. Existing WhatsApp Business Account (WABA)
2. Facebook Business Manager admin approval
3. One-time onboarding process
4. Update message endpoint to MM Lite endpoint
5. Configure webhooks for enhanced metrics

---

4. Webhook Implementation Guide

Overview

Webhooks are essential for receiving real-time updates about message status, user responses, and system notifications.

Webhook Setup Process

Step 1: Create Webhook Endpoint

```javascript
```

```
// Express.js webhook endpoint example
app.get('/webhook', (req, res) => {
  // Verification request from WhatsApp
  const mode = req.query['hub.mode'];
  const token = req.query['hub.verify_token'];
  const challenge = req.query['hub.challenge'];

  if (mode && token) {
    if (mode === 'subscribe' && token === process.env.VERIFY_TOKEN) {
      console.log('Webhook verified');
      res.status(200).send(challenge);
    } else {
      res.sendStatus(403);
    }
  }
});

app.post('/webhook', (req, res) => {
  // Handle incoming messages and status updates
  const body = req.body;

  if (body.object === 'whatsapp_business_account') {
    // Process the webhook data
    body.entry.forEach(entry => {
      const changes = entry.changes;
      // Handle messages, status updates, etc.
    });

    res.sendStatus(200);
  } else {
    res.sendStatus(404);
  }
});
```

Step 2: Configure in Meta Developer Dashboard

1. Navigate to your app in Meta Developer Dashboard
2. Go to WhatsApp > Configuration > Webhooks
3. Enter your webhook URL: `https://your-domain.com/webhook`
4. Set a secure Verify Token (store in environment variables)
5. Click "Verify and Save"

Step 3: Subscribe to Webhook Fields

1. In the Webhooks section, click "Manage"
2. Select the events you want to receive:
   - `messages` - Incoming messages
   - `message_status` - Delivery receipts
   - `message_template_status_update` - Template approval status
   - `account_alerts` - Account-level notifications

Webhook Events Structure

Incoming Message Event

```json
{
  "object": "whatsapp_business_account",
  "entry": [{
    "id": "WABA_ID",
    "changes": [{
      "value": {
        "messaging_product": "whatsapp",
        "metadata": {
          "display_phone_number": "15550555555",
          "phone_number_id": "PHONE_NUMBER_ID"
        },
        "messages": [{
          "from": "15551234567",
          "id": "MESSAGE_ID",
          "timestamp": "1234567890",
          "text": {
            "body": "Hello!"
          },
          "type": "text"
        }]
      },
      "field": "messages"
    }]
  }]
}
```

Message Status Event

```json
```

```
{
  "object": "whatsapp_business_account",
  "entry": [{
    "id": "WABA_ID",
    "changes": [{
      "value": {
        "messaging_product": "whatsapp",
        "metadata": {
          "display_phone_number": "15550555555",
          "phone_number_id": "PHONE_NUMBER_ID"
        },
        "statuses": [{
          "id": "MESSAGE_ID",
          "status": "delivered",
          "timestamp": "1234567890",
          "recipient_id": "15551234567"
        }]
      },
      "field": "messages"
    }]
  }]
}
```

Webhook Security Best Practices

1. **Verify Webhook Signatures**: Validate that requests come from WhatsApp
2. **Use HTTPS**: Always use SSL/TLS encryption
3. **Implement Rate Limiting**: Protect against DoS attacks
4. **Process Asynchronously**: Use queues for heavy processing
5. **Respond Quickly**: Return 200 status within 20 seconds

---

5. Implementation Recommendations for WhatsWay

Architecture Considerations

1. Multi-Channel Management

- Store WhatsApp phone numbers with their configuration
- Implement channel switching logic
- Track rate limits per number
- Handle multiple webhook endpoints

2. Message Routing

```javascript
// Example routing logic
async function sendMessage(channelId, messageType, recipient, content) {
  const channel = await getChannel(channelId);

  if (messageType === 'marketing' && channel.mmLiteEnabled) {
    // Route to MM Lite endpoint
    return sendViaMMlite(channel, recipient, content);
  } else {
    // Route to standard Cloud API
    return sendViaCloudAPI(channel, recipient, content);
  }
}
```

3. Queue Management with BullMQ

```javascript
// Message queue setup
const messageQueue = new Queue('whatsapp-messages', {
  connection: redis,
  defaultJobOptions: {
    attempts: 3,
    backoff: {
      type: 'exponential',
      delay: 2000
    }
  }
});

// Process messages respecting rate limits
messageQueue.process('send-message', async (job) => {
  const { channelId, recipient, message } = job.data;

  // Check rate limits
  const canSend = await checkRateLimit(channelId);
  if (!canSend) {
    throw new Error('Rate limit exceeded');
  }

  // Send message
  return await sendWhatsAppMessage(channelId, recipient, message);
});
```

4. Database Schema Considerations

```sql
-- Channels table
CREATE TABLE whatsapp_channels (
  id UUID PRIMARY KEY,
  account_id UUID REFERENCES accounts(id),
  phone_number VARCHAR(20) UNIQUE,
  phone_number_id VARCHAR(50),
  waba_id VARCHAR(50),
  access_token TEXT ENCRYPTED,
  mm_lite_enabled BOOLEAN DEFAULT FALSE,
  rate_limit_tier VARCHAR(20),
  created_at TIMESTAMP DEFAULT NOW()
);

-- Messages table
CREATE TABLE messages (
  id UUID PRIMARY KEY,
  channel_id UUID REFERENCES whatsapp_channels(id),
  recipient_number VARCHAR(20),
  message_type VARCHAR(20), -- 'marketing', 'utility', 'authentication'
  template_name VARCHAR(100),
  status VARCHAR(20), -- 'queued', 'sent', 'delivered', 'read', 'failed'
  whatsapp_message_id VARCHAR(100),
  sent_via VARCHAR(20), -- 'cloud_api', 'mm_lite'
  conversation_id VARCHAR(100),
  cost DECIMAL(10, 4),
  created_at TIMESTAMP DEFAULT NOW(),
  sent_at TIMESTAMP,
  delivered_at TIMESTAMP,
  read_at TIMESTAMP
);
```

Key Technical Considerations

1. **Access Token Management**
   - Implement token refresh logic
   - Store tokens securely (encrypted)
   - Handle token expiration gracefully
2. **Template Management**
   - Cache approved templates
   - Track template status via webhooks
   - Implement template versioning

3. **Error Handling**
    - Implement exponential backoff for retries
    - Handle specific error codes appropriately
    - Log errors for debugging
4. **Compliance & Quality**
    - Implement opt-in/opt-out management
    - Monitor quality rating via webhooks
    - Respect 24-hour session windows
    - Handle spam reports properly

---

6. Cost Considerations

Pricing Model

- **API Access**: Free
- **Conversation-Based Pricing**:
    - User-Initiated: Lower cost
    - Business-Initiated: Higher cost
    - Varies by country and message type

Cost Optimization Strategies

1. Use MM Lite for marketing messages (better delivery = better ROI)
2. Batch messages within 24-hour windows
3. Monitor and optimize template performance
4. Implement proper user segmentation

---

7. Migration Strategy

For Existing On-Premises API Users

1. **Immediate Action Required**: Must migrate before October 2025
2. **Migration Benefits**:
    - No infrastructure costs
    - Automatic updates
    - Better reliability
    - New features access

Migration Steps

1. Create Meta Developer App
2. Configure Cloud API access
3. Update API endpoints in code
4. Migrate webhook configurations
5. Test thoroughly in sandbox
6. Gradual rollout to production

---

8. Conclusion & Next Steps

For WhatsWay Implementation

1. **Start with Cloud API**: Get basic functionality working
2. **Request MM Lite Access**: Apply for beta/early access
3. **Build Flexible Architecture**: Support both APIs from day one
4. **Focus on Compliance**: Implement proper opt-in and quality monitoring
5. **Plan for Scale**: Design queue and rate limiting from the start

Resources

- Meta Developer Documentation: https://developers.facebook.com/docs/whatsapp
- Cloud API Getting Started: Available in Meta Developer Dashboard
- Community Support: Meta Developer Community Forums
- Updates: Follow Meta's WhatsApp Business Platform changelog

Support Channels

- Technical Issues: Meta Developer Support
- API Questions: Stack Overflow (tag: whatsapp-cloud-api)
- Business Queries: WhatsApp Business Solution Providers (BSPs)

Report generated: August 2025
Note: WhatsApp API features and availability are subject to change. Always refer to official Meta documentation for the most current information.