Partner API and Embed Integration Guide

Agent instruction	Copy

```
Implement ChatArchitect WhatsApp Partner embedded
registration in this codebase following the official guide:
https://support.chatarchitect.com/books/whatsapp-for-
partners/page/partner-api-and-embed-integration-guide.
First inspect the existing frontend/backend/env/storage
patterns, then add the backend session creation endpoint,
frontend iframe flow with postMessage handling and signed
webhook verification/storage.
Keep all Partner API calls server-side, never expose
partner token to browser code.
```

This guide explains how a partner backend creates a WhatsApp registration session, embeds the registration UI in an iframe, receives completion events, and retrieves final app credentials.

Overview

The integration is backend-first:

1. Your backend calls the Partner API to create a registration session.
2. The API returns a `registration_id` and an `iframe_url`.
3. Your frontend renders the `iframe_url` in an iframe.

4. The user completes phone verification, the form, and Meta Embedded Signup inside the iframe.
5. On completion, ChatArchitect sends a signed webhook to your backend.
6. Your backend stores the returned `app_id` and `app_secret`.
7. If webhook delivery fails or you need a fallback, your backend can call the result endpoint.

The iframe never exposes `app_secret` to the browser, the DOM, URLs, or `postMessage`. Final credentials are delivered only to your backend through the webhook or the Partner Result API.

Base URL

Use the public base URL provided by ChatArchitect for your environment:

```
https://<chatarchitect-registration-domain>
```

All Partner API endpoints in this guide are relative to that base URL.

Authentication

All Partner API requests require a Bearer token:

```
Authorization: Bearer <partner_token>
```

ChatArchitect provides this token to you. Treat it as a secret. The same value is also used as the HMAC secret for webhook signature verification.

Create Embed Session

Create a new registration session from your backend.

```
POST /partner?action=create_session
Authorization: Bearer <partner_token>
Content-Type: application/json
```

Request body:

```
{
  "parent_origin": "https://partner.example",
```

```
  "webhook_url": "https://partner.example/api/chatarchitect/whatsapp-registration-webhook",
  "partner_user_id": "user_456",
  "partner_state": "opaque-state-value",
  "integration": "My CRM",
  "lang": "en"
}
```

Fields:

Field	Required	Description
`parent_origin`	Yes	Origin of the parent page that will host the iframe, for example `https://partner.example`. Use scheme, host, and optional port only. Must start with `http://` or `https://`. Used as the target origin for iframe `postMessage` events.
`webhook_url`	Yes	HTTPS URL that receives the completion webhook. Must start with `https://`.
`partner_user_id`	No	Your user/account/customer id. Returned unchanged in the webhook.
`partner_state`	No	Opaque value for your own correlation or CSRF/session checks. Returned unchanged in the webhook.
`integration`	No	Optional custom integration name. It can be any string. If omitted or blank, the registration uses `Partner`.
`lang`	No	Initial language code. Defaults to `en`. Supported values: `de`, `en`, `es`, `pt`, `ru`.

Successful response:

```
{
  "registration_id": "abc123def456ghi789jkl0",
  "iframe_url": "https://<chatarchitect-registration-
domain>/?s=abc123def456ghi789jkl0&mode=embed",
  "expires_at": "2026-07-05T10:00:00+00:00",
  "status": "started"
}
```

Notes:

- Store `registration_id` in your backend. You need it for result polling and webhook retry.
- `iframe_url` is safe to send to your frontend.
- The stored session mode is authoritative. Adding `mode=embed` to another URL does not convert a standalone session into an embed session.

Example with `curl`:

```
curl -X POST "https://<chatarchitect-registration-domain>/partner?action=create_session" \
  -H "Authorization: Bearer <partner_token>" \
  -H "Content-Type: application/json" \
  -d '{
    "parent_origin": "https://partner.example",
    "webhook_url": "https://partner.example/api/chatarchitect/whatsapp-registration-webhook",
    "partner_user_id": "user_456",
    "partner_state": "opaque-state-value",
    "integration": "My CRM",
    "lang": "en"
  }'
```

Embed the Iframe

Render the returned `iframe_url` in your frontend.

```
<iframe
  id="ca-whatsapp-registration"
  src="https://<chatarchitect-registration-domain>/?s=abc123def456ghi789jkl0&mode=embed"
  style="width: 100%; height: 720px; border: 0;"
  allow="clipboard-write; encrypted-media; fullscreen"
></iframe>
```

Avoid a fixed `min-height` if the iframe sits inside a modal or another constrained container. The iframe sends `height` events as its content changes, so the parent page can shrink or grow the iframe instead of forcing an extra scrollbar.

If you use the `sandbox` attribute, include enough permissions for the registration flow and Meta Embedded Signup:

```
<iframe
  src="https://<chatarchitect-registration-domain>/?s=abc123def456ghi789jkl0&mode=embed"
  sandbox="allow-scripts allow-forms allow-same-origin allow-popups allow-popups-to-escape-
```

```
sandbox"
></iframe>
```

Avoid placing secrets in the iframe URL. The only required iframe state is already contained in `iframe_url`.

Iframe Events

The iframe sends status-only events to the parent window using `postMessage`.

Message shape:

```
{
  "type": "ca.whatsapp.registration",
  "event": "ready",
  "registration_id": "abc123def456ghi789jkl0"
}
```

Events:

Event	Meaning
`ready`	Iframe loaded and initialized.
`height`	Iframe height changed. Payload can include `height`.
`phone_verified`	User verified the WhatsApp phone number.
`facebook_started`	User clicked Connect WhatsApp and Meta Embedded Signup started.
`completed`	Registration completed. Fetch credentials from your backend storage or Partner Result API.
`failed`	Registration failed. Payload can include `message`.
`cancelled`	User cancelled Meta Embedded Signup.

Parent-page listener example:

```
<script>
  const iframe = document.getElementById('ca-whatsapp-registration');
  const expectedOrigin = 'https://<chatarchitect-registration-domain>';
  const minHeight = 480;
  const maxModalHeight = () => Math.max(minHeight, window.innerHeight - 220);
```

```
  window.addEventListener('message', (event) => {
    if (event.origin !== expectedOrigin) return;

    const data = event.data || {};
    if (data.type !== 'ca.whatsapp.registration') return;

    if (data.event === 'height' && data.height) {
      const contentHeight = Math.max(minHeight, Number(data.height));
      iframe.style.height = `${Math.min(contentHeight, maxModalHeight())}px`;
    }

    if (data.event === 'completed') {
      // Your backend should receive the signed webhook.
      // Optionally call your own backend to refresh registration status.
    }

    if (data.event === 'failed') {
      // Show a retry/support state in your UI.
    }
  });
</script>
```

`postMessage` never contains `app_secret`.

Completion Webhook

When an embed registration completes, ChatArchitect sends a `POST` request to the `webhook_url` from the session creation request.

Payload:

```
{
  "event": "whatsapp.registration.completed",
  "registration_id": "abc123def456ghi789jkl0",
  "partner_user_id": "user_456",
  "partner_state": "opaque-state-value",
  "app_id": "ca_app_123",
  "app_secret": "secret_value",
  "phone": "421233221242",
```

```
  "integration": "My CRM",
  "completed_at": "2026-07-04T10:00:00+00:00"
}
```

Headers:

```
Content-Type: application/json
X-CA-Event-Id: evt_<registration_id>_<attempt>
X-CA-Timestamp: <unix_timestamp>
X-CA-Signature: sha256=<hmac_sha256>
```

Signature formula:

```
sha256=<HMAC_SHA256(X-CA-Timestamp + "." + raw_request_body, partner_token)>
```

Verification requirements:

- Read the raw request body before JSON parsing.
- Reject stale timestamps according to your replay window, for example 5 minutes.
- Compute HMAC SHA-256 using your Partner API Bearer token as the secret.
- Compare the computed signature with `X-CA-Signature` using constant-time comparison.
- Process events idempotently by `X-CA-Event-Id` or `registration_id`.

Node.js verification example:

```
import crypto from 'node:crypto';

function verifyChatArchitectWebhook({ rawBody, timestamp, signature, partnerToken }) {
  const expected = 'sha256=' + crypto
    .createHmac('sha256', partnerToken)
    .update(`${timestamp}.${rawBody}`)
    .digest('hex');

  const expectedBuffer = Buffer.from(expected);
  const signatureBuffer = Buffer.from(signature || '');
  if (expectedBuffer.length !== signatureBuffer.length) {
    return false;
  }

  return crypto.timingSafeEqual(
    expectedBuffer,
    signatureBuffer
```

```
  );
}
```

Respond with any `2xx` status when the webhook is accepted. Non-`2xx` responses are treated as failed delivery and can be retried through the Partner API.

Get Registration Result

Use the result endpoint as a backend fallback or status check.

```
GET /partner?action=result&registration_id=<registration_id>
Authorization: Bearer <partner_token>
```

If registration is not completed yet:

```
{
  "status": "phone_verified"
}
```

Possible in-progress statuses include:

```
started
phone_code_sent
phone_verified
form_completed
fb_started
expired
```

If registration is completed:

```
{
  "status": "completed",
  "registration_id": "abc123def456ghi789jkl0",
  "app_id": "ca_app_123",
  "app_secret": "secret_value",
  "phone": "421233221242",
  "integration": "My CRM",
  "completed_at": "2026-07-04T10:00:00+00:00"
}
```

The result endpoint only works for the partner token that created the session.

Retry Webhook

Retry webhook delivery after a completed registration.

```
POST /partner?action=retry_webhook
Authorization: Bearer <partner_token>
Content-Type: application/json
```

Request body:

```
{
  "registration_id": "abc123def456ghi789jkl0"
}
```

Response:

```
{
  "status": "sent",
  "attempts": 2,
  "last_error": ""
}
```

If the registration is not completed, the endpoint returns `409`.

Error Responses

Common error shape:

```
{
  "status": "error",
  "message": "Invalid registration id"
}
```

Common HTTP statuses:

Status	Meaning

`400`	Invalid action, request body, `parent_origin`, `webhook_url`, or registration id.
`401`	Missing or invalid Bearer token.
`404`	Session not found, or it belongs to another partner.
`409`	Retry requested before registration completion.
`500`	Server-side failure.

Security Checklist

- Call Partner API only from your backend, never from browser JavaScript.
- Keep the Partner API token secret. It is also the webhook signing secret.
- Use HTTPS for your parent page and webhook endpoint in production.
- Verify every webhook signature before trusting `app_id` or `app_secret`.
- Store `app_secret` only in backend storage.
- Treat iframe `postMessage` events as status hints only, not as a source of credentials.
- Validate `event.origin` when receiving iframe messages.
- Handle duplicate webhooks idempotently.

Recommended Flow

```
sequenceDiagram
    participant PartnerBackend as Partner backend
    participant PartnerFrontend as Partner frontend
    participant CA as ChatArchitect registration
    participant User as User

    PartnerBackend->>CA: POST /partner?action=create_session
    CA-->>PartnerBackend: registration_id, iframe_url
    PartnerBackend-->>PartnerFrontend: iframe_url
    PartnerFrontend->>CA: Load iframe_url
    CA-->>PartnerFrontend: postMessage ready/height
    User->>CA: Verify phone, fill form, complete Meta signup
    CA-->>PartnerFrontend: postMessage completed
    CA->>PartnerBackend: Signed completion webhook
    PartnerBackend-->>CA: 2xx accepted
    PartnerBackend->>PartnerBackend: Store app_id and app_secret
```

Revision #6
Created 2026-07-05 09:38:27 UTC by New Admin
Updated 2026-07-05 10:15:09 UTC by New Admin