Guia de Integração — Twilio + Storage (EF Core / Dapper)

````
# Guia de Integração — Twilio + Storage (EF Core / Dapper)

## 1) Mínimo necessário (`Program.cs`)
```csharp
using WhatsApp.Messaging.Twilio.DependencyInjection; // AddWhatsAppMessagingTwilio

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddWhatsAppMessagingTwilio(builder.Configuration);

var app = builder.Build();
app.Run();
```

---

## 2) Com **EF Core** como storage (`Program.cs`)
```csharp
using Microsoft.EntityFrameworkCore;
using WhatsApp.Messaging.Twilio.DependencyInjection;
using WhatsApp.Messaging.Storage.EFCore.DependencyInjection; // AddWhatsAppEfStorage

var builder = WebApplication.CreateBuilder(args);

builder.Services
    .AddWhatsAppMessagingTwilio(builder.Configuration)
    .AddWhatsAppEfStorage(db =>
        db.UseSqlServer(builder.Configuration.GetConnectionString("WhatsApp")));

var app = builder.Build();
app.Run();
```

---

## 3) Com **Dapper** como storage (`Program.cs`)
```csharp
using WhatsApp.Messaging.Twilio.DependencyInjection;
using WhatsApp.Messaging.Storage.Dapper.DependencyInjection; // AddWhatsAppDapperStorage
````
````

```
var builder = WebApplication.CreateBuilder(args);

builder.Services
    .AddWhatsAppMessagingTwilio(builder.Configuration)
    .AddWhatsAppDapperStorage(builder.Configuration.GetConnectionString("WhatsApp"));

var app = builder.Build();
app.Run();
```

---

4) `appsettings.json` (exemplo)
```json
{
  "Twilio": {
    "AccountSid": "ACxxxxxxxx...",
    "AuthToken": "your_auth_token",
    "FromNumber": "whatsapp:+14155238886",
    "ContentTemplates": {
      "confirmacao_atendimento": {
        "sid": "HX11111111111111111111111111111111",
        "locale": "pt_BR",
        "defaults": { "empresa": "Sensia" }
      },
      "permissao_contato": { "sid": "HX22222222222222222222222222222222" },
      "recusa_agradecimento": { "sid": "HX33333333333333333333333333333333" }
    }
  },
  "ConnectionStrings": {
    "WhatsApp": "Server=.;Database=WhatsAppDb;Trusted_Connection=True;TrustServerCertificate=True"
  }
}
```

Handlers e webhook

1) Implemente o handler (no seu projeto Web/API)

```csharp
using WhatsApp.Messaging.Core;
using WhatsApp.Messaging.Twilio.Sending;

public class MeuHandler : IIncomingMessageHandler
{
    private readonly ITwilioTemplateSender _tpl;
    public MeuHandler(ITwilioTemplateSender tpl) => _tpl = tpl;

    public Task<bool> HandleAsync(IncomingMessage msg, CancellationToken ct) =>
        _tpl.SendTemplateAsync(
            key: "confirmacao_atendimento",
            to: msg.From,
            variables: new Dictionary<string, object> { ["nome"] = "Carla" },
            ct: ct
        ).ContinueWith(_ => true, ct);
}
```

2) Registre o handler na DI
```csharp
builder.Services.AddSingleton<IIncomingMessageHandler, MeuHandler>();
```

3) Endpoint de webhook que dispara o pipeline
```csharp
using WhatsApp.Messaging.Core;

app.MapPost("/webhooks/whatsapp", async (HttpRequest req, IWebhookParser parser,
IIncomingMessagePipeline pipeline) =>
{
    var form    = (await req.ReadFormAsync()).ToDictionary(k => k.Key, v => v.Value.ToString());
    var headers = req.Headers.ToDictionary(h => h.Key, h => h.Value.ToString(),
StringComparer.OrdinalIgnoreCase);
    var query   = req.Query.ToDictionary(q => q.Key, q => q.Value.ToString(),
StringComparer.OrdinalIgnoreCase);

    var wh = new WebhookRequest { Headers = headers, Query = query, Form = form, Body = req.Body }
    if (!parser.CanParse(wh)) return Results.BadRequest("Invalid webhook provider");

    var msg = await parser.ParseAsync(wh, req.HttpContext.RequestAborted);
```

```
    await pipeline.DispatchAsync(msg, req.HttpContext.RequestAborted); // chama seu MeuHandler
    return Results.Ok();
});

app.Run();
```