# Projeto: Captura de Consumo de Alarmes FTAE via Injeção COM

## Objetivo

O projeto visa analisar em profundidade como o **FactoryTalk Alarms and Events (FTAE)** do **FactoryTalk Studio v13** (ambiente distribuído) consome e manipula eventos de alarme.  
A ideia é **interceptar as chamadas COM** feitas pelo cliente e servidor, permitindo investigar:

* Como ocorre a **escuta** de alarmes.
* O comportamento interno ao **receber eventos**.
* Como cada alarme é **configurado e instanciado**.

## Ambiente de Teste

* **SO:** Windows 10
* **IDE:** Visual Studio 2022
* **Linguagem:** C# 7.3
* **Framework:** .NET Framework 4.8
* **Ferramentas auxiliares:** PowerShell 5.1 para execução de testes e automação.
* **Execução:** Cliente e Servidor FTAE rodando na **mesma máquina**.

## Estrutura da Solução

A solução (Banner.sln) contém dois projetos principais

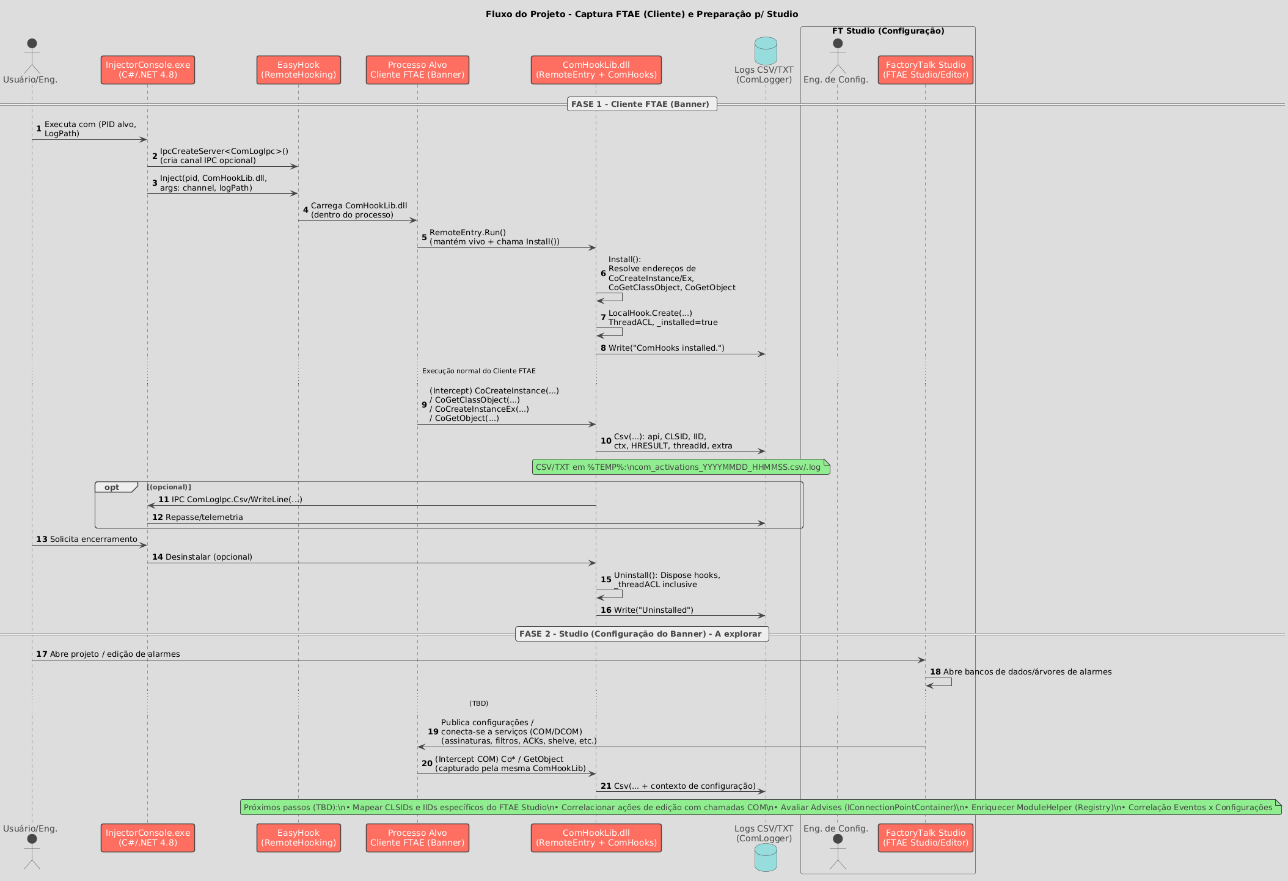


Figura 1 - Fluxo do prgrama

## Etapa 1/4 – Base de Injeção e Placeholders

* InjectorConsole implementado (Program.cs) com suporte a parâmetros PID, LogPath e DLL.
* RemoteEntry (ComHookLib) carregando hooks de UI (UiHook) e inicializando COM em modo placeholder.
* UiHook captura eventos de janela (window.create, window.show, window.foreground).
* ComHooks apenas inicializado como placeholder (sem interceptar CoCreateInstance ainda).
* Logging estruturado ativo: Jsonl, ComLogger, ComLogIpc (NamedPipe + fallback para arquivo).

Status atual:

* ✔️ Injeção confirmada
* ✔️ Hooks de UI ativos
* ✔️ Placeholder COM inicializado
* ⏸️ Hooks reais de COM ainda não instalados (aguardam etapa 2/4).

## Etapa 2/4 – Instalação dos Hooks COM

* Hooks reais instalados para interceptação das funções CoCreateInstance, CoCreateInstanceEx e CoGetClassObject.
* Logs capturam:
* CLSID e IID utilizados em cada chamada.
* Tradução de CLSID para ProgID quando disponível (Native.ProgIdFromClsidSafe).
* Flags de contexto (CLSCTX) em formato legível.
* Resultado da chamada (HRESULT) e tempo decorrido (elapsed\_ms).
* Exemplos observados em ambiente de teste:
* RnaClientCore.RnaNamespace.1
* RSHMI.ParSrv.1
* HMIDiagnostics.CHMIDiagClt.1
* HMI.Connection.ConnectionServer.1
* FTAlarmSummary.FTAlarmBannerCtl.1 ✅ (relacionado diretamente ao **Alarm Banner**).

Status atual:

* ✔️ Injeção confirmada
* ✔️ Hooks de UI ativos (UiHook)
* ✔️ Hooks COM instalados e capturando chamadas reais
* ⏸️ Ainda sem enriquecimento semântico (tradução detalhada de HRESULTs, IIDs e classificação de objetos).

## Próxima etapa: 3/4 – Enriquecimento da Captura

* Traduzir HRESULTs para nomes legíveis (ex.: 0x80004002 → E\_NOINTERFACE).
* Mapear IIDs para interfaces conhecidas (IUnknown, IDispatch, IConnectionPointContainer).
* Identificar e marcar objetos críticos do FTAE (Alarm Banner, Alarm Summary, Subscriptions).
* Estruturar o log para facilitar análise futura (JSONL enriquecido).