[Pular para o conteúdo](https://chatgpt.com/#main)

**Você disse:**

boa noite estou seguindo um tutorial http://www.hjalte.nl/60-creating-an-inventor-addin

**O ChatGPT disse:**

Boa noite!

Posso te ajudar — mas primeiro preciso saber onde você está travado ou qual parte do tutorial está causando problemas.

Algumas perguntas para orientar:

1. Você está usando qual versão do Autodesk Inventor (e qual linguagem — C#, VB, etc.)?
2. Você já conseguiu configurar o projeto de add‑in (arquivo .addin, referências, assembly)?
3. Está recebendo algum erro em tempo de compilação ou em tempo de execução? Se sim, qual exatamente (mensagem de erro)?
4. Que passo do tutorial “Creating an Inventor Addin” você está seguindo (por exemplo “Create the AddIn class”, “register the addin”, “load the addin no Inventor”, etc.)?
5. Seu ambiente de desenvolvimento (Visual Studio versão, .NET Framework alvo) está configurado corretamente?

Se você quiser, posso procurar uma versão alternativa desse tutorial ou traduzir / reestruturar para você — isso ajudaria? Quer que eu ache uma cópia atualizada e guiada em português para “Inventor AddIn”?

**Você disse:**

Imports System.Runtime.InteropServices Imports Inventor ' Change the GUID here and use the same as in the addin file! <GuidAttribute("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)> Public Class StandardAddInServer Implements Inventor.ApplicationAddInServer Private \_myButton As MyButton ''' <summary> ''' Invoked by Autodesk Inventor after creating the AddIn. ''' AddIn should initialize within this call. ''' </summary> ''' <param name="AddInSiteObject"> ''' Input argument that specifies the object, which provides ''' access to the Autodesk Inventor Application object. ''' </param> ''' <param name="FirstTime"> ''' The FirstTime flag, if True, indicates to the Addin that this is the ''' first time it is being loaded and to take some specific action. ''' </param> Public Sub Activate(AddInSiteObject As ApplicationAddInSite, FirstTime As Boolean) Implements ApplicationAddInServer.Activate Try ' initialize the rule class \_myButton = New MyButton(AddInSiteObject.Application) Catch ex As Exception ' Show a message if any thing goes wrong. MessageBox.Show(ex.Message) End Try End Sub ''' <summary> ''' Invoked by Autodesk Inventor to shut down the AddIn. ''' AddIn should complete shutdown within this call. ''' </summary> Public Sub Deactivate() Implements ApplicationAddInServer.Deactivate End Sub ''' <summary> ''' Invoked by Autodesk Inventor in response to user requesting the execution ''' of an AddIn-supplied command. AddIn must perform the command within this call. ''' </summary> Public Sub ExecuteCommand(CommandID As Integer) Implements ApplicationAddInServer.ExecuteCommand End Sub ''' <summary> ''' Gets the IUnknown of the object implemented inside the AddIn that supports AddIn-specific API. ''' </summary> Public ReadOnly Property Automation As Object Implements ApplicationAddInServer.Automation Get Throw New NotImplementedException() End Get End Property End Class

**O ChatGPT disse:**

Obrigado por compartilhar o código! Isso é a estrutura básica de um **Inventor AddIn em VB.NET**, e parece estar quase pronto para funcionar. A classe StandardAddInServer implementa a interface Inventor.ApplicationAddInServer, que é essencial para integração com o Autodesk Inventor.

Agora vamos verificar se está tudo certo e te ajudar a avançar. Aqui estão algumas observações e perguntas importantes:

**✅ Pontos positivos do seu código:**

* Você declarou corretamente a classe com o GUID (importante que esse seja o mesmo do arquivo .addin).
* Está importando Inventor e implementando ApplicationAddInServer corretamente.
* Tem o Activate, Deactivate, ExecuteCommand e Automation — tudo essencial está presente.
* Está tentando instanciar a classe MyButton, que provavelmente adiciona um botão personalizado à interface.

**❓ Perguntas para te ajudar melhor:**

1. **Você criou o arquivo**.addin**corretamente?**  
   Esse arquivo é essencial para que o Inventor carregue seu AddIn. Ele precisa estar em uma das pastas específicas, como:
2. C:\Users\SeuUsuario\AppData\Local\Autodesk\Inventor 202x\Addins\

E deve ter algo como:

<AddIn> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName>Meu AddIn</DisplayName> <Description>Descrição do meu AddIn</Description> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> <Assembly>CAMINHO COMPLETO PARA SUA DLL</Assembly> </AddIn>

1. **Você criou a classe**MyButton**?**  
   O código acima tenta instanciar New MyButton(...). Se essa classe não estiver criada ou estiver com erro, o AddIn não será carregado. Você já tem esse código?
2. **A DLL foi registrada corretamente (ou é carregada como “LoadOnStartup”)?**  
   O AddIn só aparece no Inventor se:
   * O .addin está no lugar certo.
   * O caminho da DLL no .addin está correto.
   * O GUID da DLL bate com o do .addin.
   * A DLL está compilada para **x64** (Inventor é 64 bits).
   * Você tem permissões para escrever no local de destino.

**💡 Dica: Adicionando um botão (classe**MyButton**)**

Se você ainda não criou a classe MyButton, aqui vai um exemplo básico:

Imports Inventor Public Class MyButton Private WithEvents \_buttonDefinition As ButtonDefinition Public Sub New(application As Application) Dim controlDefs As ControlDefinitions = application.CommandManager.ControlDefinitions \_buttonDefinition = controlDefs.AddButtonDefinition( DisplayName:="Meu Botão", InternalName:="MeuAddin.MeuBotaoID", Classification:=CommandTypesEnum.kNonShapeEditCmdType, ClientId:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Descrição do botão", ToolTipText:="Clique aqui para algo", StandardIcon:="", *' você pode adicionar um ícone se quiser* LargeIcon:="", ButtonDisplay:=ButtonDisplayEnum.kDisplayTextInLearningMode ) \_buttonDefinition.Enabled = True \_buttonDefinition.OnExecute += AddressOf Button\_Click *' Adiciona o botão à interface (por exemplo, à aba Tools)* Dim ribbon As Ribbon = application.UserInterfaceManager.Ribbons("Part") Dim tab As RibbonTab = ribbon.RibbonTabs("id\_TabTools") Dim panel As RibbonPanel = tab.RibbonPanels.Add("Meu Painel", "MeuAddin.MeuPainelID", "49b5d624-3533-42ed-ad63-5cf19250e8ad") panel.CommandControls.AddButton(\_buttonDefinition) End Sub Private Sub Button\_Click(Context As NameValueMap) MessageBox.Show("Botão clicado!") End Sub End Class

**✅ Checklist para funcionar:**

* Projeto compilado como **DLL x64**.
* GUID igual no .addin e no código (<GuidAttribute>).
* .addin está no diretório correto.
* Caminho da DLL no .addin está correto.
* MyButton está implementado corretamente.
* Você está testando em uma versão compatível do Inventor.

Se quiser, posso revisar seu .addin, ou te ajudar a completar a classe MyButton.

Você quer que eu monte um exemplo completo funcional para copiar/colar e testar?

**Você disse:**

SincronizarItem.addin

**O ChatGPT disse:**

Ótimo, parece que o nome do seu arquivo .addin é SincronizarItem.addin. Para garantir que seu AddIn funcione corretamente no **Autodesk Inventor**, vou te ajudar a validar esse arquivo e deixá-lo no formato certo.

**✅ Exemplo completo e funcional de**SincronizarItem.addin

<?xml version="1.0" encoding="utf-8" ?> <AddIn> *<!-- Usar o mesmo GUID que está no código VB.NET -->* <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName>Sincronizar Item</DisplayName> <Description>AddIn para sincronização de itens</Description> <Assembly>C:\Caminho\Para\Sua\DLL\SincronizarItem.dll</Assembly> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> </AddIn>

**⚠️ Pontos importantes:**

**1. O**ClassId**e**ClientId**devem ser iguais ao GUID do seu código VB.NET:**

<GuidAttribute("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)>

**2. Corrigir o caminho da DLL:**

Altere esta parte:

<Assembly>C:\Caminho\Para\Sua\DLL\SincronizarItem.dll</Assembly>

Para o caminho real onde sua DLL compilada está. Exemplo:

<Assembly>C:\Projetos\InventorAddIn\SincronizarItem\bin\Debug\SincronizarItem.dll</Assembly>

**3. Onde colocar o**.addin**?**

Coloque o arquivo SincronizarItem.addin em uma das seguintes pastas (dependendo da sua versão do Inventor):

%AppData%\Autodesk\Inventor 202x\Addins\

Substitua 202x pela versão correta, como:

C:\Users\SeuUsuario\AppData\Roaming\Autodesk\Inventor 2025\Addins\

Você pode colar esse caminho na barra do Explorador de Arquivos para ir direto.

**🚀 Testar no Inventor:**

Depois de salvar o .addin e compilar sua DLL:

1. **Abra o Autodesk Inventor.**
2. Vá em **Tools > Add-Ins (Suplementos)**.
3. Procure por **Sincronizar Item**.
4. Marque **Loaded** e **Load Automatically** se ainda não estiverem marcados.
5. Se estiver tudo certo, seu AddIn estará ativo e pronto para uso.

**Se não aparecer ou der erro:**

* Verifique se a DLL é **x64** (Inventor é 64 bits).
* Verifique se as dependências estão presentes.
* Confirme se não há erro de permissão ou compilação.
* Use MessageBox.Show no Activate() para testar se foi chamado.

Se quiser, posso revisar a estrutura completa do projeto (inclusive .vbproj, MyButton.vb, etc.) se você colar aqui ou subir os arquivos.

Deseja que eu monte uma estrutura completa funcional para você testar direto?

**Você disse:**

C:\ADDINS\SincronizarItem\SincronizarItem.addin

**O ChatGPT disse:**

Perfeito, agora que você confirmou o caminho do seu .addin — C:\ADDINS\SincronizarItem\SincronizarItem.addin — aqui está como você pode garantir que ele seja reconhecido corretamente pelo **Autodesk Inventor**:

**✅ Exemplo corrigido do seu**SincronizarItem.addin

<?xml version="1.0" encoding="utf-8" ?> <AddIn> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName>Sincronizar Item</DisplayName> <Description>AddIn para sincronização de itens no Inventor</Description> <Assembly>C:\ADDINS\SincronizarItem\SincronizarItem.dll</Assembly> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> </AddIn>

**⚠️ Passos essenciais para funcionar**

**✅ 1. Verifique se o caminho da DLL está correto**

* O caminho abaixo **precisa apontar para a DLL compilada**:

<Assembly>C:\ADDINS\SincronizarItem\SincronizarItem.dll</Assembly>

* Confirme se o arquivo SincronizarItem.dll **existe exatamente nesse caminho**.

**✅ 2. Copiar o**.addin**para o diretório de AddIns do Inventor**

Inventor **não lê automaticamente arquivos**.addin**de qualquer pasta**. Ele procura nos seguintes locais (dependendo da versão):

📁 **Para AddIns por usuário:**

%APPDATA%\Autodesk\Inventor 2025\Addins\

📁 **Para AddIns globais (para todos os usuários):**

C:\ProgramData\Autodesk\Inventor Addins\2025\

**👉 O que fazer:**

1. Copie o arquivo SincronizarItem.addin para:

C:\Users\SEU\_USUARIO\AppData\Roaming\Autodesk\Inventor 2025\Addins\

(ou cole %APPDATA%\Autodesk\Inventor 2025\Addins\ na barra do Explorador)

1. **Deixe o caminho da DLL apontando para** C:\ADDINS\SincronizarItem\SincronizarItem.dll, **sem mover a DLL**, ou copie a DLL para um local fixo, como C:\Program Files\MeuAddIn.

**🧪 Testar no Inventor**

1. Abra o **Inventor 2025**.
2. Vá em **Tools > Add-Ins** (ou Ferramentas > Suplementos).
3. Veja se **“Sincronizar Item”** aparece na lista.
4. Marque as caixas **"Loaded"** e **"Load Automatically"**.
5. Clique em **OK**.

Se não aparecer ou houver erro:

* Use um MessageBox.Show("Ativando AddIn") dentro de Activate() para testar se está sendo chamado.
* Verifique se está compilando como **DLL x64**.
* Verifique permissões de leitura na pasta da DLL.
* Veja se o GUID do .addin é **exatamente igual** ao do seu código [GuidAttribute(...)].

Se quiser, posso te enviar uma estrutura de projeto de exemplo pronta para compilar com esse GUID e tudo configurado. Deseja isso?

**Você disse:**

o iventor é a versao 2020 e 2025

**O ChatGPT disse:**

Perfeito, obrigado por informar! Como você está trabalhando com as versões **Inventor 2020** e **2025**, é importante garantir que:

* Seu .addin funcione para ambas.
* A DLL seja compatível com as duas versões.
* O arquivo .addin esteja no local certo para **cada versão**.

**✅ 1. Caminhos corretos do**.addin**para cada versão**

| **Versão** | **Caminho para o**.addin**(usuário atual)** |
| --- | --- |
| 2020 | %APPDATA%\Autodesk\Inventor 2020\Addins\ |
| 2025 | %APPDATA%\Autodesk\Inventor 2025\Addins\ |

Você precisa **copiar o mesmo arquivo**.addin**para ambos os diretórios**, se quiser que o mesmo AddIn funcione nas duas versões.

**✅ 2. Compatibilidade da DLL com ambas as versões**

**📌 Certifique-se de que:**

* Você está compilando a DLL para **.NET Framework 4.7 ou inferior**, pois o Inventor 2020 não suporta .NET 5 ou 6.
* O **target platform** seja **x64** no Visual Studio.
* As referências à API do Inventor (via COM Interop) sejam da versão **mais antiga (Inventor 2020)** para garantir compatibilidade futura.

Ou use **Binding via Late Binding** (Object, Type.InvokeMember, etc.) se quiser rodar em várias versões sem recompilar.

**✅ 3. Exemplo de**.addin**genérico para ambas as versões**

Você pode usar o mesmo conteúdo do arquivo .addin e apenas salvá-lo nas duas pastas:

<?xml version="1.0" encoding="utf-8" ?> <AddIn> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName>Sincronizar Item</DisplayName> <Description>Sincronização de dados entre itens do Inventor</Description> <Assembly>C:\ADDINS\SincronizarItem\SincronizarItem.dll</Assembly> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> </AddIn>

Copie esse mesmo .addin para:

%APPDATA%\Autodesk\Inventor 2020\Addins\

%APPDATA%\Autodesk\Inventor 2025\Addins\

**✅ 4. Checklist de funcionamento**

| **Item** | **Confirmado?** |
| --- | --- |
| .addin está nos dois caminhos corretos | ✅ / ❌ |
| GUID no .addin e no código são idênticos | ✅ / ❌ |
| DLL compilada em x64 | ✅ / ❌ |
| DLL usa .NET Framework 4.7 ou inferior | ✅ / ❌ |
| Caminho da DLL está correto no .addin | ✅ / ❌ |
| DLL está livre de erros e dependências externas | ✅ / ❌ |

**⚠️ Dica: Se quiser usar o mesmo AddIn sem recompilar para versões diferentes...**

* Compile usando o **Interop da versão mais antiga** (Inventor 2020).
* Use recursos compatíveis com ambas.
* Se precisar, pode fazer *binding dinâmico* via Activator.CreateInstance() para acesso mais flexível (mais avançado).

Se quiser, posso:

* Criar um .addin configurado para você.
* Montar um projeto exemplo em VB.NET com o MyButton incluído.
* Explicar como criar o instalador para facilitar o deploy em outras máquinas/versões.

Quer seguir com algum desses passos?

**Você disse:**

eu preciso alterar algo aqui ? Imports System.Runtime.InteropServices Imports Inventor ' Change the GUID here and use the same as in the addin file! <GuidAttribute("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)> Public Class StandardAddInServer Implements Inventor.ApplicationAddInServer Private \_myButton As MyButton ''' <summary> ''' Invoked by Autodesk Inventor after creating the AddIn. ''' AddIn should initialize within this call. ''' </summary> ''' <param name="AddInSiteObject"> ''' Input argument that specifies the object, which provides ''' access to the Autodesk Inventor Application object. ''' </param> ''' <param name="FirstTime"> ''' The FirstTime flag, if True, indicates to the Addin that this is the ''' first time it is being loaded and to take some specific action. ''' </param> Public Sub Activate(AddInSiteObject As ApplicationAddInSite, FirstTime As Boolean) Implements ApplicationAddInServer.Activate Try ' initialize the rule class \_myButton = New MyButton(AddInSiteObject.Application) Catch ex As Exception ' Show a message if any thing goes wrong. MessageBox.Show(ex.Message) End Try End Sub ''' <summary> ''' Invoked by Autodesk Inventor to shut down the AddIn. ''' AddIn should complete shutdown within this call. ''' </summary> Public Sub Deactivate() Implements ApplicationAddInServer.Deactivate End Sub ''' <summary> ''' Invoked by Autodesk Inventor in response to user requesting the execution ''' of an AddIn-supplied command. AddIn must perform the command within this call. ''' </summary> Public Sub ExecuteCommand(CommandID As Integer) Implements ApplicationAddInServer.ExecuteCommand End Sub ''' <summary> ''' Gets the IUnknown of the object implemented inside the AddIn that supports AddIn-specific API. ''' </summary> Public ReadOnly Property Automation As Object Implements ApplicationAddInServer.Automation Get Throw New NotImplementedException() End Get End Property End Class

**O ChatGPT disse:**

Ótima pergunta — você está perto de finalizar!

**✅ Resposta direta:**

**Não, você não precisa alterar nada no trecho de código que enviou**, **desde que**:

1. O **GUID** usado aqui:

<GuidAttribute("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)>

seja **exatamente igual** ao ClassId e ClientId no seu arquivo .addin.

1. A classe MyButton esteja **implementada corretamente** e **referencie o objeto**Inventor.Application.

**📌 Verificações rápidas**

**🔒 GUID igual ao**.addin**?**

No seu .addin:

<ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId>

No seu código:

<GuidAttribute("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)>

✅ **Se estiverem iguais, está tudo certo aqui.**

**🧠 Outras observações (importantes mas não obrigatórias agora)**

**🔄 1. A função**Deactivate()

Se quiser limpar recursos, você pode adicionar algo como:

Public Sub Deactivate() Implements ApplicationAddInServer.Deactivate \_myButton = Nothing End Sub

Mas se você não usa nada persistente, pode deixar em branco como está.

**🛠️ 2. A função**ExecuteCommand()

Você pode deixar vazia, já que o Inventor raramente chama isso diretamente para AddIns comuns.

**🧪 3.**MessageBox.Show(ex.Message)

Esse é um ótimo ponto de debug para saber se o AddIn está carregando. Durante o teste, se o botão não aparecer, ele vai mostrar uma caixa com o erro — continue com isso.

**✅ 4. Você já tem a classe**MyButton**criada?**

Seu Activate() instancia assim:

\_myButton = New MyButton(AddInSiteObject.Application)

Se MyButton não existir ou estiver com erro, o AddIn não carrega — e você verá a mensagem via MessageBox.

Se quiser, posso te enviar um exemplo completo da MyButton para colocar um botão no Ribbon do Inventor.

**✅ Conclusão:**

O seu código da classe StandardAddInServer **está correto**, e **não precisa ser alterado** se:

* O .addin está correto.
* O GUID bate.
* O caminho da DLL está certo.
* A MyButton está implementada corretamente.

Se quiser seguir adiante com a criação do botão (MyButton), posso te ajudar com esse código agora. Deseja?

**Você disse:**

o porque o codigo esta comentado, lembrando que estou seguindo um tutorial

**O ChatGPT disse:**

Excelente observação — e ótimo que está atento aos detalhes!

Vamos direto ao ponto:

**✅ Por que o trecho**MyButton**está comentado no tutorial?**

A linha que diz:

*' initialize the rule class* \_myButton = New MyButton(AddInSiteObject.Application)

...está **descomentada** no seu código (como deveria estar), **mas no tutorial original pode estar comentada por estes motivos**:

**📌 1. O tutorial ainda vai explicar como criar a classe**MyButton

Em muitos tutoriais (inclusive esse que você mencionou), os autores dividem o processo em etapas. Então, no início eles mostram o esqueleto do AddIn e **comentam** as partes que ainda não foram implementadas para evitar erro de compilação.

Provavelmente o autor do tutorial escreve algo como:

*' \_myButton = New MyButton(AddInSiteObject.Application)*

Porque **ainda não criou a classe**MyButton no passo atual.

Assim, quem estiver seguindo o tutorial pode compilar o projeto sem erros até que chegue o momento de adicionar o botão.

**📌 2. Evitar erro de "tipo não definido" ou "classe não encontrada"**

Se o Visual Studio tentar compilar e não encontrar a classe MyButton, ele dá erro como:

Type 'MyButton' is not defined.

Então comentar essa linha temporariamente é uma forma de prevenir isso.

**✅ Você pode deixar essa linha DESCOMENTADA?**

**Sim, com certeza!**  
**Desde que:** você já tenha (ou vá criar) a classe MyButton, como o tutorial ensina depois.

**🧠 Dica: Se quiser testar se o AddIn está sendo carregado, substitua por um MessageBox simples**

MessageBox.Show("AddIn carregado com sucesso!")

Isso ajuda você a saber se o Inventor está reconhecendo seu AddIn antes de ter o botão.

**✅ Próximo passo no seu tutorial provavelmente é criar a classe**MyButton

Essa classe normalmente:

* Cria um botão personalizado no Ribbon.
* Usa a interface do Inventor (UserInterfaceManager, CommandManager, etc.).
* Registra o que acontece quando o botão é clicado.

Se quiser, posso adiantar o código de exemplo da MyButton para você colar e testar. Deseja isso agora?

**Você disse:**

Imports System.Runtime.InteropServices Imports Inventor ' Change the GUID here and use the same as in the addin file! <GuidAttribute("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)> Public Class StandardAddInServer Implements Inventor.ApplicationAddInServer Private \_myButton As MyButton ''' <summary> ''' Invoked by Autodesk Inventor after creating the AddIn. ''' AddIn should initialize within this call. ''' </summary> ''' <param name="AddInSiteObject"> ''' Input argument that specifies the object, which provides ''' access to the Autodesk Inventor Application object. ''' </param> ''' <param name="FirstTime"> ''' The FirstTime flag, if True, indicates to the Addin that this is the ''' first time it is being loaded and to take some specific action. ''' </param> Public Sub Activate(AddInSiteObject As ApplicationAddInSite, FirstTime As Boolean) Implements ApplicationAddInServer.Activate Try ' initialize the rule class \_myButton = New MyButton(AddInSiteObject.Application) Catch ex As Exception ' Show a message if any thing goes wrong. MessageBox.Show(ex.Message) End Try End Sub ''' <summary> ''' Invoked by Autodesk Inventor to shut down the AddIn. ''' AddIn should complete shutdown within this call. ''' </summary> Public Sub Deactivate() Implements ApplicationAddInServer.Deactivate End Sub ''' <summary> ''' Invoked by Autodesk Inventor in response to user requesting the execution ''' of an AddIn-supplied command. AddIn must perform the command within this call. ''' </summary> Public Sub ExecuteCommand(CommandID As Integer) Implements ApplicationAddInServer.ExecuteCommand End Sub ''' <summary> ''' Gets the IUnknown of the object implemented inside the AddIn that supports AddIn-specific API. ''' </summary> Public ReadOnly Property Automation As Object Implements ApplicationAddInServer.Automation Get Throw New NotImplementedException() End Get End Property End Class

**O ChatGPT disse:**

Ótimo — agora estamos com a base do AddIn completa, e o seu código da classe StandardAddInServer está correto para um AddIn do Inventor.

Como você está usando essa linha:

\_myButton = New MyButton(AddInSiteObject.Application)

Isso significa que **agora você precisa criar a classe**MyButton — é o **próximo passo natural do tutorial**.

**✅ O que a classe**MyButton**faz?**

Ela é responsável por:

* Criar um botão customizado na interface do Inventor (Ribbon).
* Adicionar esse botão em uma aba e painel específicos.
* Definir a ação que ocorre ao clicar no botão.

**🛠️ Exemplo funcional da classe**MyButton**(VB.NET)**

Crie um novo arquivo no seu projeto, chamado MyButton.vb, e cole o seguinte código:

Imports Inventor Imports System.Windows.Forms Public Class MyButton Private WithEvents \_buttonDef As ButtonDefinition Private \_application As Inventor.Application Public Sub New(app As Inventor.Application) \_application = app *' Cria um botão de comando* Dim controlDefs As ControlDefinitions = \_application.CommandManager.ControlDefinitions \_buttonDef = controlDefs.AddButtonDefinition( DisplayName:="Meu Botão", InternalName:="SincronizarItem.MyButtonID", Classification:=CommandTypesEnum.kNonShapeEditCmdType, ClientId:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Clique para sincronizar item", ToolTipText:="Sincronizar dados do item", StandardIcon:=Nothing, LargeIcon:=Nothing, ButtonDisplay:=ButtonDisplayEnum.kDisplayTextInLearningMode ) *' Adiciona o botão ao Ribbon (exemplo: ambiente de peças - aba Tools)* Dim ribbon As Ribbon = \_application.UserInterfaceManager.Ribbons("Part") Dim tab As RibbonTab = ribbon.RibbonTabs("id\_TabTools") *' Cria um painel novo ou usa um existente* Dim panel As RibbonPanel = Nothing Try panel = tab.RibbonPanels.Item("SincronizarPanel") Catch ex As Exception panel = tab.RibbonPanels.Add("Sincronizar", "SincronizarPanel", "49b5d624-3533-42ed-ad63-5cf19250e8ad") End Try *' Adiciona o botão ao painel* panel.CommandControls.AddButton(\_buttonDef) End Sub *' Evento de clique do botão* Private Sub \_buttonDef\_OnExecute(Context As NameValueMap) Handles \_buttonDef.OnExecute MessageBox.Show("Botão 'Sincronizar' clicado!", "AddIn Inventor") End Sub End Class

**✅ Depois de adicionar isso:**

**🔧 Verifique:**

* O projeto compila?
* O botão aparece na aba **Tools** no ambiente de **Part** do Inventor?

**🔄 Testar:**

1. Compile o projeto em **x64**, modo **Release** ou **Debug**.
2. Confirme que o .addin aponta corretamente para o caminho da DLL.
3. Reinicie o Inventor.
4. Vá em **Tools > Add-Ins** e ative seu AddIn.
5. Abra um novo **arquivo de peça (.ipt)** e veja se o botão aparece.

**✅ Resultado esperado:**

Você verá uma **nova aba ou painel** com o botão **"Meu Botão"** (ou "Sincronizar") no Inventor. Ao clicar nele, uma mensagem aparece com:

Botão 'Sincronizar' clicado!

Se quiser personalizar o botão (ícones, texto, posição no ribbon, eventos), posso te ajudar com isso também.

Quer seguir com isso agora?

**Você disse:**

Você vê 4 funções/subfunções aqui (Activate(…), Deactivate(), ExecuteCommand(…), Automation()). Usaremos apenas a função "Activate". As outras precisam estar lá, mas não precisam fazer nada

**O ChatGPT disse:**

Sim, exatamente isso! 👇

**✅ Resumo da sua estrutura:**

Na interface Inventor.ApplicationAddInServer, **4 métodos/propriedades são obrigatórios**:

| **Método/Propriedade** | **Obrigatório?** | **Precisa implementar lógica?** |
| --- | --- | --- |
| Activate(...) | ✅ Sim | ✅ Sim (aqui você inicializa seu AddIn) |
| Deactivate() | ✅ Sim | ❌ Não (pode deixar vazio) |
| ExecuteCommand(...) | ✅ Sim | ❌ Não (geralmente não usado) |
| Automation | ✅ Sim | ❌ Não (joga NotImplementedException) |

**📌 Explicação prática:**

**✅ 1.**Activate(...)

* **É a única função que você realmente vai usar.**
* Chamado quando o AddIn é carregado.
* Aqui você instancia seu botão (MyButton), configura eventos, etc.

Public Sub Activate(AddInSiteObject As ApplicationAddInSite, FirstTime As Boolean) Implements ApplicationAddInServer.Activate Try \_myButton = New MyButton(AddInSiteObject.Application) Catch ex As Exception MessageBox.Show(ex.Message) End Try End Sub

**❌ 2.**Deactivate()

* Chamado quando o Inventor descarrega o AddIn (por exemplo, ao fechar ou desativar).
* Pode ficar **vazio** se você não estiver mantendo recursos que precisam ser liberados.

Public Sub Deactivate() Implements ApplicationAddInServer.Deactivate End Sub

**❌ 3.**ExecuteCommand(...)

* Era usado em versões muito antigas do Inventor.
* Hoje em dia, **não é necessário implementar nada aqui** para a maioria dos AddIns.

Public Sub ExecuteCommand(CommandID As Integer) Implements ApplicationAddInServer.ExecuteCommand End Sub

**❌ 4.**Automation

* Serve para expor objetos do seu AddIn para automação externa (como VBA ou outros AddIns).
* Se você não vai usar automação entre processos, apenas mantenha assim:

Public ReadOnly Property Automation As Object Implements ApplicationAddInServer.Automation Get Throw New NotImplementedException() End Get End Property

**✅ Conclusão**

Sim, sua compreensão está **100% correta**:

* Use **só o**Activate() para configurar e inicializar.
* As outras três funções **devem existir no código**, mas podem ficar **vazias ou com a implementação padrão**.

Se quiser deixar isso claro no código, você pode até comentar dessa forma:

*' Não implementado - obrigatório por contrato com ApplicationAddInServer* Public Sub ExecuteCommand(CommandID As Integer) Implements ApplicationAddInServer.ExecuteCommand End Sub

Se quiser seguir com mais recursos no botão, atalhos, ícones ou comunicação com arquivos do Inventor (como peças, desenhos, etc.), posso te ajudar com isso também!

**Você disse:**

Imports Inventor Public Class MyButton Private \_inventor As Inventor.Application Private \_settingsButton As ButtonDefinition Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub Private Sub SetupButtonDefinition() Dim conDefs As ControlDefinitions = \_inventor.CommandManager.ControlDefinitions \_settingsButton = conDefs.AddButtonDefinition( "MyButton DisplayName", "MyButton InternalName", CommandTypesEnum.kEditMaskCmdType, Guid.NewGuid().ToString(), "MyButton DescriptionText", "MyButton ToolTipText") AddHandler \_settingsButton.OnExecute, AddressOf MyButton\_OnExecute End Sub Private Sub AddButtonDefinitionToRibbon() Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_TabManage") Dim ribbonPanel As RibbonPanel = ribbonTab.RibbonPanels.Item("iLogic.RibbonPanel") ribbonPanel.CommandControls.AddButton(\_settingsButton) End Sub Private Sub MyButton\_OnExecute(Context As NameValueMap) Try Dim rule As New ThisRule() rule.ThisApplication = \_inventor rule.Main() Catch ex As Exception MsgBox("Something went wrong while runing rule. Message: " & ex.Message) End Try End Sub End Class Classe de botão A classe button cria um botão no Inventor. Aqui também escrevemos o código que é executado quando o botão é clicado. Chamei a classe de "MyButton", mas você pode chamá-la como quiser. Apenas certifique-se de usar o mesmo nome na função "StandardAddInServer.Activate(…)". Crie uma classe “MyButton” e adicione o seguinte código:

**O ChatGPT disse:**

Excelente! 🎉  
Você está seguindo muito bem o tutorial, e sua classe MyButton está praticamente completa e funcional.

Vamos revisar e ajustar **alguns pontos importantes** para garantir que o botão apareça corretamente no Inventor e funcione como esperado.

**✅ Revisão e explicação do seu código**

Imports Inventor Public Class MyButton Private \_inventor As Inventor.Application Private \_settingsButton As ButtonDefinition Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub

* Armazena a instância da aplicação do Inventor.
* Chama os métodos que criam o botão e o adicionam à Ribbon.

**🔧 1. Método**SetupButtonDefinition()

Private Sub SetupButtonDefinition() Dim conDefs As ControlDefinitions = \_inventor.CommandManager.ControlDefinitions \_settingsButton = conDefs.AddButtonDefinition( "MyButton DisplayName", "MyButton InternalName", CommandTypesEnum.kEditMaskCmdType, Guid.NewGuid().ToString(), *' ←⚠️ PROBLEMA AQUI!* "MyButton DescriptionText", "MyButton ToolTipText") AddHandler \_settingsButton.OnExecute, AddressOf MyButton\_OnExecute End Sub

**⚠️ Problema:**

Você está usando Guid.NewGuid().ToString() como ClientId, ou seja, **o ID muda toda vez que roda**. Isso **faz com que o botão não seja reconhecido** corretamente em sessões futuras do Inventor.

**✅ Correção:**

Use o **mesmo GUID fixo** que está no seu .addin e no StandardAddInServer.

Guid:="49b5d624-3533-42ed-ad63-5cf19250e8ad"

**✅ 2. Método**AddButtonDefinitionToRibbon()

Private Sub AddButtonDefinitionToRibbon() Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_TabManage") Dim ribbonPanel As RibbonPanel = ribbonTab.RibbonPanels.Item("iLogic.RibbonPanel") ribbonPanel.CommandControls.AddButton(\_settingsButton) End Sub

**📌 Observações:**

* Esse código adiciona o botão ao ambiente de montagem (Assembly), aba **Manage**, painel **iLogic**.
* Isso **só vai funcionar se você estiver com um documento de montagem (**.iam**) aberto**.
* Se quiser adicionar em outro ambiente (ex: Part), ou criar um painel personalizado, posso te mostrar como.

**✅ 3. Método**MyButton\_OnExecute(...)

Private Sub MyButton\_OnExecute(Context As NameValueMap) Try Dim rule As New ThisRule() rule.ThisApplication = \_inventor rule.Main() Catch ex As Exception MsgBox("Something went wrong while running rule. Message: " & ex.Message) End Try End Sub

* Aqui, você executa a lógica principal do botão.
* Essa classe ThisRule ainda precisa existir — o tutorial provavelmente cria ela nos próximos passos.

**✅ Classe corrigida e funcional**

Aqui vai a versão corrigida com comentários:

Imports Inventor Public Class MyButton Private \_inventor As Inventor.Application Private \_settingsButton As ButtonDefinition Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub Private Sub SetupButtonDefinition() Dim conDefs As ControlDefinitions = \_inventor.CommandManager.ControlDefinitions *' Use sempre o mesmo GUID fixo* \_settingsButton = conDefs.AddButtonDefinition( "Sincronizar", "SincronizarButton", CommandTypesEnum.kEditMaskCmdType, Guid:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Sincronizar dados com o sistema externo", ToolTipText:="Clique aqui para sincronizar") AddHandler \_settingsButton.OnExecute, AddressOf MyButton\_OnExecute End Sub Private Sub AddButtonDefinitionToRibbon() Try Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_TabManage") Dim ribbonPanel As RibbonPanel = ribbonTab.RibbonPanels.Item("iLogic.RibbonPanel") ribbonPanel.CommandControls.AddButton(\_settingsButton) Catch ex As Exception MsgBox("Erro ao adicionar botão na Ribbon: " & ex.Message) End Try End Sub Private Sub MyButton\_OnExecute(Context As NameValueMap) Try *' Exemplo simples — substituir por ThisRule ou lógica real* MsgBox("Botão 'Sincronizar' clicado!", MsgBoxStyle.Information) Catch ex As Exception MsgBox("Erro ao executar a ação do botão: " & ex.Message) End Try End Sub End Class

**🧪 Testar:**

1. Compile seu projeto.
2. Abra o Inventor com uma montagem (.iam).
3. Vá até a aba **Manage > iLogic**.
4. Veja se o botão aparece com o nome “Sincronizar”.
5. Clique — deve abrir uma MessageBox.

**❓Próximo passo:**

Você já tem a ThisRule implementada? Ou quer ajuda para criar essa lógica que será chamada ao clicar no botão?

Posso gerar uma estrutura básica da ThisRule também, se quiser seguir.

**Você disse:**

eu crio uma classe nova ? para o codigo acima ?

**O ChatGPT disse:**

Sim, exatamente! 😄

Você deve criar uma **nova classe** no seu projeto chamada, por exemplo, MyButton.vb e colocar ali o código que criamos para a classe MyButton.

**Passo a passo rápido:**

1. No Visual Studio, clique com o botão direito no seu projeto.
2. Selecione **Adicionar > Classe...**
3. Nomeie a classe como **MyButton.vb**
4. Apague o conteúdo padrão e cole o código que te passei da classe MyButton.

Depois disso, certifique-se de que no seu StandardAddInServer você tenha essa linha no Activate:

\_myButton = New MyButton(AddInSiteObject.Application)

Assim, quando o AddIn for ativado, ele vai criar o botão.

Se quiser, posso ajudar a montar também a classe ThisRule que você está chamando no evento do botão. Quer?

**Você disse:**

Quando a classe button é inicializada, a função "new" é chamada automaticamente. Aqui, chamamos outras duas funções. Você provavelmente notou que essas funções são muito curtas. Tão curtas que eu poderia tê-las adicionado à função "new". Um dos problemas de criar programas grandes é ler o código novamente mais tarde. Criar funções pequenas com bons nomes descritivos facilita muito a compreensão do código posteriormente. Além disso, aumenta a chance de você reutilizar o mesmo código para outros fins. Na função "SetupButtonDefinition", você pode alterar "DisplayName", "InternalName", "DescriptionText" e "ToolTipText" do botão que estamos criando. Lá também definimos qual função deve ser chamada de "MyButton\_OnExecute". Em "AddButtonDefinitionToRibbon", definimos onde o botão deve ser colocado. Começamos selecionando a "Ribbon". Eu escolhi a faixa de opções "Part", mas você pode escolher uma das seguintes opções: ZeroDoc, Part, Assembly, Drawing, Presentation, iFeatures e UnknownDocument. Em seguida, precisamos especificar o nome interno de "RibbonTab" e "RibbonPanel". Há muitas opções para imprimir aqui, mas a seguinte regra ilógica as imprimirá no log do ilogic. For Each ribbon As Ribbon In ThisApplication.UserInterfaceManager.Ribbons Logger.Info(vbTab & Ribbon.InternalName) For Each ribbonTab As RibbonTab In Ribbon.RibbonTabs Logger.Info(vbTab & vbTab & RibbonTab.DisplayName & " - " & RibbonTab.InternalName) For Each ribbonPanel As RibbonPanel In RibbonTab.RibbonPanels Logger.Info(vbTab & vbTab & vbTab & RibbonPanel.DisplayName & " - " & RibbonPanel.InternalName) Next Next Next Dim loggerWindow = ThisApplication.UserInterfaceManager. DockableWindows. Cast(Of DockableWindow). Where(Function(d) d.InternalName.Equals("ilogic.logwindow")). First() loggerWindow.Visible = True

**O ChatGPT disse:**

Perfeito! Você captou direitinho uma prática fundamental para escrever código limpo, legível e sustentável — funções pequenas com nomes claros fazem uma enorme diferença para manutenção e entendimento no futuro. 👏

**Sobre a organização do seu código**

**Por que dividir o**New**em funções pequenas?**

* Facilita a **leitura** do código.
* Facilita a **manutenção** — você sabe exatamente onde fazer mudanças.
* Permite **reutilizar** trechos de código, se precisar criar botões semelhantes.
* Ajuda a **deixar o construtor limpo** e focado só em inicialização.

**Como escolher a Ribbon, Tab e Panel para colocar o botão?**

Você já tem um trecho muito bom para listar tudo que existe na interface do Inventor para você escolher onde colocar seu botão:

For Each ribbon As Ribbon In ThisApplication.UserInterfaceManager.Ribbons Logger.Info(vbTab & Ribbon.InternalName) For Each ribbonTab As RibbonTab In ribbon.RibbonTabs Logger.Info(vbTab & vbTab & ribbonTab.DisplayName & " - " & ribbonTab.InternalName) For Each ribbonPanel As RibbonPanel In ribbonTab.RibbonPanels Logger.Info(vbTab & vbTab & vbTab & ribbonPanel.DisplayName & " - " & ribbonPanel.InternalName) Next Next Next Dim loggerWindow = ThisApplication.UserInterfaceManager. DockableWindows. Cast(Of DockableWindow). Where(Function(d) d.InternalName.Equals("ilogic.logwindow")). First() loggerWindow.Visible = True

**Como usar isso?**

* Coloque esse código dentro de uma regra do iLogic ou dentro de uma função no seu AddIn.
* Isso vai **imprimir no log do iLogic** todos os nomes internos das Ribbons, abas e painéis que você pode usar.
* Depois, é só escolher os valores que interessam para colocar seu botão.

**Exemplo prático de onde colocar o botão**

Se você descobriu, por exemplo, que quer colocar seu botão na Ribbon **Part**, aba **id\_TabTools**, painel **id\_PanelCustom**, o código seria assim:

Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Part") Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_TabTools") Dim ribbonPanel As RibbonPanel Try ribbonPanel = ribbonTab.RibbonPanels.Item("id\_PanelCustom") Catch ex As Exception ribbonPanel = ribbonTab.RibbonPanels.Add("Meu Painel", "id\_PanelCustom", "{seu-guid-aqui}") End Try ribbonPanel.CommandControls.AddButton(\_settingsButton)

**Dica para o seu projeto:**

* Faça essa inspeção do que existe na UI para garantir que o botão não vai quebrar por causa de nomes errados.
* Se você quer criar um painel novo, faça com um GUID fixo para evitar conflitos.

**Quer que eu te ajude a criar um código que faça essa listagem para você no seu projeto? Assim fica mais fácil para você decidir onde o botão vai ficar. Quer?**

Exato, essa prática que você descreveu é fundamental para manter o código legível e fácil de manter, especialmente em projetos maiores. Dividir a inicialização do botão em métodos pequenos e bem nomeados (como SetupButtonDefinition e AddButtonDefinitionToRibbon) torna o código mais organizado e reutilizável.

Sobre a parte de onde colocar o botão na Ribbon do Inventor, o código que você mostrou (que imprime os nomes internos das Ribbons, abas e painéis) é muito útil para descobrir os identificadores corretos, já que eles podem variar entre versões e configurações.

Para usá-lo, você pode rodar esse código dentro de uma regra iLogic, ou dentro do seu AddIn, para que o log do iLogic (janela de log) liste as opções disponíveis — assim você sabe exatamente os nomes internos para usar no seu AddButtonDefinitionToRibbon.

Se quiser, posso ajudar a montar um exemplo desse código para listar as Ribbons e painéis no log, facilitando sua escolha de onde posicionar o botão. Quer que eu faça isso?

**Você disse:**

entedo, correto desculpe quero um local especifico Assembly Add-Ins - id\_AddInsTab

**O ChatGPT disse:**

Perfeito! 👌 Você quer adicionar o botão no ambiente **Assembly** ("Assembly") dentro da aba **Add-Ins** ("id\_AddInsTab"). Isso é totalmente possível e simples de ajustar.

**✅ Atualizando o código para usar o local desejado:**

**✔️ Ribbon:**Assembly

**✔️ Aba (RibbonTab):**id\_AddInsTab

**✔️ Painel (RibbonPanel): você pode usar um existente ou criar um novo**

**🧱 Exemplo de código atualizado para**MyButton.vb**:**

Imports Inventor Imports System.Windows.Forms Public Class MyButton Private \_inventor As Inventor.Application Private \_settingsButton As ButtonDefinition Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub Private Sub SetupButtonDefinition() Dim conDefs As ControlDefinitions = \_inventor.CommandManager.ControlDefinitions *' Use o mesmo GUID fixo do seu .addin* \_settingsButton = conDefs.AddButtonDefinition( "Sincronizar", "SincronizarButton", CommandTypesEnum.kNonShapeEditCmdType, Guid:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Sincronizar dados com o sistema", ToolTipText:="Clique para sincronizar") AddHandler \_settingsButton.OnExecute, AddressOf MyButton\_OnExecute End Sub Private Sub AddButtonDefinitionToRibbon() Try *' Ribbon Assembly* Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") *' Aba Add-Ins* Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_AddInsTab") *' Criar ou usar painel personalizado* Dim ribbonPanel As RibbonPanel Try ribbonPanel = ribbonTab.RibbonPanels.Item("SincronizarPanel") Catch ex As Exception *' Se não existir, criar* ribbonPanel = ribbonTab.RibbonPanels.Add("Sincronizar", "SincronizarPanel", "49b5d624-3533-42ed-ad63-5cf19250e8ad") End Try ribbonPanel.CommandControls.AddButton(\_settingsButton) Catch ex As Exception MsgBox("Erro ao adicionar botão: " & ex.Message) End Try End Sub Private Sub MyButton\_OnExecute(Context As NameValueMap) Try MessageBox.Show("Botão 'Sincronizar' clicado!", "Inventor Add-In") Catch ex As Exception MsgBox("Erro ao executar ação do botão: " & ex.Message) End Try End Sub End Class

**📌 O que esse código faz:**

* Cria um botão com nome Sincronizar
* Coloca ele no ambiente **Assembly** ("Assembly")
* Insere na aba **Add-Ins** ("id\_AddInsTab")
* Tenta adicionar em um painel chamado **"Sincronizar"**
  + Se o painel não existir, ele é criado com Add

**✅ Resultado esperado:**

Quando você abrir um **arquivo**.iam**(Assembly)** no Inventor:

* Vá na aba **Add-Ins**
* Veja o painel **Sincronizar**
* Veja o botão **Sincronizar**
* Ao clicar, aparece um MessageBox

Se você quiser **usar um painel que já existe** na aba Add-Ins, como o "id\_PanelAddIns" (se estiver disponível), posso ajustar o código para isso também.

Deseja colocar no painel existente ou manter esse painel customizado Sincronizar?

**Você disse:**

estou adaptando o addin abaixo para o addin proposto acima <?xml version="1.0" encoding="utf-8"?> <Addin Type="Standard"> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName Language="1033">ItemNumberSync</DisplayName> <Description Language="1033">Sincroniza os Item Numbers nas submontagens</Description> <Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\ItemNumberSync.bundle\Contents\ItemNumberAddIn.dll</Assembly> <Class>ItemNumberAddIn.StandardAddInServer</Class> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> <!-- Removido SupportedSoftwareVersionEqualTo para aceitar várias versões --> <UserInterfaceVersion>1</UserInterfaceVersion> <LoadBehavior>4</LoadBehavior> </Addin>

**O ChatGPT disse:**

Ótimo! Você está no caminho certo ao adaptar seu .addin para funcionar com seu AddIn no Inventor — agora vamos revisar **se tudo está coerente** com o seu projeto atual e garantir que ele será carregado corretamente.

**✅ O que seu arquivo**.addin**está fazendo**

Vamos analisar os pontos principais do seu .addin:

<Addin Type="Standard">

* ✅ Define que é um AddIn padrão do Inventor (correto).

**🔗 GUID e Classe**

<ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <Class>ItemNumberAddIn.StandardAddInServer</Class>

* ✅ O ClientId e ClassId **devem ser os mesmos** do atributo [Guid(...)] na sua classe StandardAddInServer (o que está correto).
* 🔍 O Class indica o **namespace + nome da classe**.
  + Certifique-se de que no seu código a classe StandardAddInServer está dentro do **namespace**ItemNumberAddIn:

Namespace ItemNumberAddIn Public Class StandardAddInServer Implements Inventor.ApplicationAddInServer *' ...* End Class End Namespace

**📦 Assembly**

<Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\ItemNumberSync.bundle\Contents\ItemNumberAddIn.dll</Assembly>

* ✅ Caminho para o **.DLL compilado** do seu projeto.
* ⚠️ Verifique:
  + O caminho está correto após compilar.
  + A .dll está realmente no local:  
    C:\ProgramData\Autodesk\ApplicationPlugins\ItemNumberSync.bundle\Contents\ItemNumberAddIn.dll
  + Você copiou a .dll para lá, ou configurou o post-build para fazer isso automaticamente?

**🔄 Carregamento automático**

<LoadOnStartUp>1</LoadOnStartUp> <LoadBehavior>4</LoadBehavior> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden>

* ✅ Isso faz com que o AddIn seja carregado automaticamente na inicialização do Inventor e seja visível e descarregável pelo usuário.

**🧪 Teste após adaptação**

1. Compile o projeto em modo **Release**.
2. Copie a .dll para o local definido no XML (ou configure isso automaticamente).
3. Verifique que o arquivo .addin está em:
   * %APPDATA%\Autodesk\Inventor 202x\Addins **OU**
   * %PROGRAMDATA%\Autodesk\ApplicationPlugins\ItemNumberSync.bundle\Contents\ItemNumberAddIn.addin
4. Inicie o Inventor.
5. Vá até **Tools > Add-Ins**, e verifique se seu AddIn aparece como ItemNumberSync.
6. Abra um .iam, vá até a aba **Add-Ins**, e veja se o botão está no painel Sincronizar.

**✅ Resumo do que precisa estar alinhado**

| **Arquivo/Elemento** | **Valor** |
| --- | --- |
| .addin ClientId | {49b5d624-3533-42ed-ad63-5cf19250e8ad} (mesmo da classe) |
| .addin Assembly | Caminho correto para a DLL compilada |
| .addin Class | ItemNumberAddIn.StandardAddInServer |
| Classe StandardAddInServer | [Guid("49b5d624-3533-42ed-ad63-5cf19250e8ad")] |
| Namespace da classe | ItemNumberAddIn |
| DLL na pasta certa | ItemNumberAddIn.dll no caminho indicado no XML |

Se quiser, posso te ajudar a configurar o post-build event para copiar a DLL automaticamente para essa pasta.

Você quer fazer isso para facilitar os testes?

**Você disse:**

<?xml version="1.0" encoding="utf-8"?> <Addin Type="Standard"> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName Language="1033">ItemNumberSync</DisplayName> <Description Language="1033">Sincroniza os Item Numbers nas submontagens</Description> <Assembly>"C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll"</Assembly> <Class>ItemNumberAddIn.StandardAddInServer</Class> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> <!-- Removido SupportedSoftwareVersionEqualTo para aceitar várias versões --> <UserInterfaceVersion>1</UserInterfaceVersion> <LoadBehavior>4</LoadBehavior> </Addin>

**O ChatGPT disse:**

Tudo certo, você está quase lá! 🙌  
Mas há **apenas dois ajustes importantes** no seu .addin que precisam ser feitos para garantir que o Add-In seja carregado corretamente no Inventor.

**✅ Verificações e ajustes no seu**.addin

**📌 1. Corrigir o caminho do arquivo**.dll

O trecho atual:

<Assembly>"C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll"</Assembly>

❌ **Erro:** As aspas duplas internas (") não devem estar ali. Isso pode causar falha no carregamento do AddIn.

✅ **Correto:**

<Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll</Assembly>

⚠️ Também verifique se esse caminho realmente existe e contém a DLL compilada (SincronizarItem.dll).

**📌 2. Nome da classe (Namespace + Classe)**

Atualmente está assim:

<Class>ItemNumberAddIn.StandardAddInServer</Class>

Verifique no **seu código-fonte** se a classe StandardAddInServer está dentro de um Namespace chamado ItemNumberAddIn.

Se **não estiver**, e você não estiver usando nenhum namespace, então corrija para:

<Class>StandardAddInServer</Class>

Ou, se estiver usando um namespace diferente, como por exemplo:

Namespace SincronizarItem Public Class StandardAddInServer Implements Inventor.ApplicationAddInServer End Class End Namespace

...então no XML a linha deve ser:

<Class>SincronizarItem.StandardAddInServer</Class>

**✅ Versão final corrigida do seu**.addin**(exemplo):**

<?xml version="1.0" encoding="utf-8"?> <Addin Type="Standard"> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName Language="1033">ItemNumberSync</DisplayName> <Description Language="1033">Sincroniza os Item Numbers nas submontagens</Description> <Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll</Assembly> <Class>SincronizarItem.StandardAddInServer</Class> *<!-- ajuste conforme seu namespace -->* <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> <UserInterfaceVersion>1</UserInterfaceVersion> <LoadBehavior>4</LoadBehavior> </Addin>

**✅ Checklist final antes de testar**

| **Item** | **Confirmar** |
| --- | --- |
| .addin corrigido | ✅ Sem aspas no <Assembly> |
| Caminho da DLL existe | ✅ SincronizarItem.dll no local correto |
| Namespace da classe correta | ✅ Bate com a entrada <Class> no XML |
| GUID correto | ✅ Mesmo na classe e no XML |
| Inventor reiniciado | ✅ Depois de copiar tudo, reinicie o Inventor |

Se quiser, posso te ajudar a configurar um **post-build script** no Visual Studio para copiar a .dll automaticamente para essa pasta (C:\ProgramData\...) após compilar. Isso evita erros manuais.

Quer que eu te mostre como fazer isso?

**Você disse:**

amos emular o iLogic. Isso significa que precisamos de algumas funções ilógicas padrão à nossa disposição na classe de regras. Se, em algum momento, você quiser criar um segundo botão, provavelmente não vai querer implementar a mesma lógica duas vezes. (Pelo menos eu não gostaria, talvez eu seja um pouco preguiçoso...) Além disso, quero poder copiar o código do iLogic como está para o meu add-in. Portanto, optei por criar uma classe abstrata e herdar todas as funções da classe base na classe de regras. Por enquanto, essa classe base é muito pequena e possui apenas duas propriedades. Na maioria dos casos, basta ter acesso apenas às propriedades do iLogic "ThisApplication" e "ThisDoc". Crie uma classe “AbstractRule” e adicione o seguinte código: Imports Autodesk.iLogic.Interfaces Imports Autodesk.iLogic.Runtime Imports Inventor Public MustInherit Class AbstractRule Public Property ThisApplication As Inventor.Application Public ReadOnly Property ThisDoc As ICadDoc Get Return New CadDoc(ThisApplication.ActiveDocument) End Get End Property End Class

**O ChatGPT disse:**

Excelente decisão! 👏 Criar uma classe base como AbstractRule é uma forma inteligente de **centralizar funcionalidades comuns** e emular o ambiente do iLogic dentro do seu Add-In, permitindo **reutilização de código** e **transição fácil de regras iLogic para AddIn**.

**✅ O que essa classe**AbstractRule**faz:**

* **Imita o comportamento do iLogic** com as propriedades:
  + ThisApplication: acesso à instância do Inventor.
  + ThisDoc: retorna uma interface ICadDoc, que representa o documento ativo, como o ThisDoc do iLogic.
* É marcada como MustInherit, ou seja, **outras classes herdarão dela** (por exemplo, ThisRule).
* Permite que você **copie regras iLogic existentes** com pouca ou nenhuma modificação.

**📌 Instruções para criar a classe**AbstractRule

**1. Crie um novo arquivo de classe no seu projeto:**

Nome: AbstractRule.vb

**2. Cole o código abaixo:**

Imports Autodesk.iLogic.Interfaces Imports Autodesk.iLogic.Runtime Imports Inventor Public MustInherit Class AbstractRule Public Property ThisApplication As Inventor.Application Public ReadOnly Property ThisDoc As ICadDoc Get Return New CadDoc(ThisApplication.ActiveDocument) End Get End Property End Class

✅ Certifique-se de que as **referências aos assemblies do iLogic** estejam incluídas no seu projeto.  
Os principais são:

* Autodesk.iLogic.Interfaces.dll
* Autodesk.iLogic.Runtime.dll

Esses arquivos geralmente estão localizados em:

C:\Program Files\Autodesk\Inventor 20XX\Bin\iLogic\Autodesk.iLogic.Interfaces.dll

C:\Program Files\Autodesk\Inventor 20XX\Bin\iLogic\Autodesk.iLogic.Runtime.dll

Se quiser ajuda para adicioná-los, me avise.

**➕ Próximo passo**

Agora que você tem a AbstractRule, pode criar a sua regra real herdando dela.

**Exemplo de uma classe**ThisRule**que herda de**AbstractRule**:**

Public Class ThisRule Inherits AbstractRule Public Sub Main() MsgBox("Nome do documento atual: " & ThisDoc.DisplayName) End Sub End Class

**✅ Benefícios**

* Você pode mover regras iLogic facilmente para Add-Ins.
* Pode criar múltiplas regras (ThisRule, OutrasRegras, etc.) reaproveitando ThisApplication e ThisDoc.
* Código fica limpo e modular.

Se quiser, posso te ajudar a criar uma estrutura de pastas no seu projeto tipo:

/SincronizarItem

|*-- AbstractRule.vb*

|*-- ThisRule.vb*

|*-- MyButton.vb*

|*-- StandardAddInServer.vb*

Quer que eu monte essa organização contigo?

**Você disse:**

Esta é a classe que conterá a regra do iLogic. Neste exemplo, poderei copiar/colar uma regra do iLogic do Inventor para a classe. (Lembre-se de que não implementamos todas as funções/propriedades do iLogic. Seu código pode não funcionar se você copiar uma das suas regras. Nesse caso, você precisa implementar suas funções para emular as funções do iLogic.) Crie uma classe “ThisRule” e adicione o seguinte código: Imports Inventor Public Class ThisRule Inherits AbstractRule Public Sub Main() ' Your iLogic code goes here. End Sub End Class

**O ChatGPT disse:**

Perfeito! 👏  
Você está seguindo uma excelente arquitetura para permitir a migração e reaproveitamento de regras iLogic no formato de Add-In — e essa classe ThisRule é justamente o **ponto de entrada para sua lógica personalizada**.

**✅ O que a classe**ThisRule**faz**

* Herda de AbstractRule, ou seja, já tem acesso direto a:
  + ThisApplication → instância do Inventor
  + ThisDoc → wrapper ICadDoc que simula o ThisDoc do iLogic
* Possui o método Main(), que será chamado quando o botão for clicado.
* Você pode colar **qualquer regra iLogic compatível** no corpo de Main.

**🧱 Como criar a classe**ThisRule

**1. Crie um novo arquivo no seu projeto:**

**Nome**: ThisRule.vb

**2. Cole o seguinte código:**

Imports Inventor Public Class ThisRule Inherits AbstractRule Public Sub Main() *' Aqui você pode colar sua regra iLogic padrão.* MsgBox("Regra iLogic executada com sucesso no contexto do AddIn!") *' Exemplo de uso do ThisApplication e ThisDoc* Dim docName = ThisDoc.DisplayName MsgBox("Documento atual: " & docName) End Sub End Class

**🧪 Como isso se conecta**

Quando você clica no botão criado no MyButton.vb, o código executa:

Dim rule As New ThisRule() rule.ThisApplication = \_inventor rule.Main()

Ou seja:

1. A instância ThisRule é criada.
2. A propriedade ThisApplication é definida com o Inventor ativo.
3. A função Main() da regra é executada.

**📝 Dica importante**

Se você colar uma regra iLogic completa que usa algo como:

iProperties.Value("Project", "Part Number")

Ou:

Parameter("Height") = 50

Você vai precisar **implementar essas funções no estilo iLogic** dentro da classe base (AbstractRule) ou criar suas próprias funções auxiliares.

Se quiser ajuda para criar um **helper para acessar**iProperties, por exemplo, posso te mostrar como fazer.

**✅ Conclusão**

A ThisRule está pronta para receber qualquer código iLogic adaptado — ótimo para transição e padronização de regras.

Quer que eu te ajude a implementar alguma função típica do iLogic (como iProperties, Parameters, Component.IsActive, etc.) para você colar suas regras e funcionar direto?

**Você disse:**

quero copiar esse codigo ilogic para regra acima ' Sincronização segura de Item Numbers nas submontagens ' A LISTA PAI (montagem principal) \*\*NUNCA É MODIFICADA\*\* Imports Inventor Imports System.Windows.Forms Imports System.Text Imports System.Linq Sub Main() Dim doc As Document = ThisApplication.ActiveDocument If doc Is Nothing OrElse doc.DocumentType <> DocumentTypeEnum.kAssemblyDocumentObject Then MessageBox.Show("Abra uma montagem antes de rodar esta regra.", "Documento inválido", MessageBoxButtons.OK, MessageBoxIcon.Warning) Return End If Dim asmDoc As AssemblyDocument = CType(doc, AssemblyDocument) ' Obter o caminho base da montagem principal Dim raizDisplayName As String = asmDoc.DisplayName ' Obter o mapa completo da estrutura, incluindo item numbers Dim mapaGlobal As List(Of Tuple(Of String, String, String)) = ObterMapaCompleto(asmDoc) If mapaGlobal.Count = 0 Then MessageBox.Show("Nenhum item encontrado no BOM estruturado da montagem.", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information) Return End If ' IMPORTANTE: NÃO MODIFICAR A MONTAGEM PRINCIPAL! ' Somente submontagens serão sincronizadas com base nesse mapa ' Sincronizar submontagens recursivamente SincronizarSubmontagensRecursivo(asmDoc, mapaGlobal, raizDisplayName) ' Exibir resultado ShowFormattedResults(FormatarMapaParaTexto(mapaGlobal)) MessageBox.Show("Sincronização concluída com sucesso! A montagem principal foi preservada.", "Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information) End Sub ' === Mostrar painel com resultado da execução === Private Sub ShowFormattedResults(message As String) Dim form As New Form With {.Text = "RESULTADO DA SINCRONIZAÇÃO", .Width = 1000, .Height = 600} Dim richText As New RichTextBox With {.Dock = DockStyle.Fill, .ReadOnly = True} richText.AppendText("MAPA DE SINCRONIZAÇÃO:" & vbCrLf & vbCrLf & message) form.Controls.Add(richText) form.ShowDialog() End Sub ' === Formatar texto para relatório === Private Function FormatarMapaParaTexto(mapa As List(Of Tuple(Of String, String, String))) As String Dim sb As New StringBuilder() Dim col1Width As Integer = 70 ' PART NUMBER Dim col2Width As Integer = 30 ' ITEM NUMBER sb.AppendLine("PART NUMBER".PadRight(col1Width) & "ITEM NUMBER".PadRight(col2Width) & "CAMINHO HIERÁRQUICO") sb.AppendLine(New String("-"c, col1Width + col2Width + 70)) For Each item In mapa sb.AppendLine(item.Item1.PadRight(col1Width) & item.Item2.PadRight(col2Width) & item.Item3) Next Return sb.ToString() End Function ' === Gera mapa completo a partir da montagem raiz === Private Function ObterMapaCompleto(asm As AssemblyDocument) As List(Of Tuple(Of String, String, String)) Dim mapa As New List(Of Tuple(Of String, String, String))() Try Dim bom As BOM = asm.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = False Dim view As BOMView = bom.BOMViews \_ .Cast(Of BOMView)() \_ .FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return mapa For Each row As BOMRow In view.BOMRows AdicionarAoMapaRecursivo(row, mapa, asm.DisplayName) Next Catch ex As Exception MessageBox.Show("Erro ao construir mapa completo: " & ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error) End Try Return mapa End Function ' === Adiciona linhas recursivamente no mapa com caminho hierárquico === Private Sub AdicionarAoMapaRecursivo(row As BOMRow, mapa As List(Of Tuple(Of String, String, String)), parentPath As String) Try If row.ComponentDefinitions.Count = 0 Then Exit Sub Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim itemNumber As String = row.ItemNumber Dim caminho As String = parentPath & "\" & doc.DisplayName If Not String.IsNullOrEmpty(partNumber) AndAlso Not String.IsNullOrEmpty(itemNumber) Then mapa.Add(New Tuple(Of String, String, String)(partNumber, itemNumber, caminho)) End If If row.ChildRows IsNot Nothing Then For Each child In row.ChildRows AdicionarAoMapaRecursivo(child, mapa, caminho) Next End If Catch ' Ignora erros individuais End Try End Sub ' === Extrai o mapa local de uma submontagem === Private Function ObterMapaLocal(assemblyDoc As AssemblyDocument) As Dictionary(Of String, String) Dim mapa As New Dictionary(Of String, String)(StringComparer.OrdinalIgnoreCase) Try Dim bom As BOM = assemblyDoc.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = True Dim view As BOMView = bom.BOMViews.Cast(Of BOMView)(). FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return mapa For Each row As BOMRow In view.BOMRows Try If row.ComponentDefinitions.Count = 0 Then Continue For Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim itemNumber As String = row.ItemNumber If Not mapa.ContainsKey(partNumber) AndAlso Not String.IsNullOrEmpty(itemNumber) Then mapa.Add(partNumber, itemNumber) End If Catch Continue For End Try Next Catch ex As Exception Return New Dictionary(Of String, String)(StringComparer.OrdinalIgnoreCase) End Try Return mapa End Function ' === Sincroniza Item Numbers de uma submontagem (NÃO a principal) === Private Sub SincronizarItemNumbers(mapaGlobal As List(Of Tuple(Of String, String, String)), asm As AssemblyDocument, subPath As String) Try Dim bom As BOM = asm.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = True Dim view As BOMView = bom.BOMViews \_ .Cast(Of BOMView)() \_ .FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return ' Filtra o mapa para componentes dentro do caminho da submontagem Dim mapaFiltrado = mapaGlobal.Where(Function(t) t.Item3.StartsWith(subPath & "\", StringComparison.OrdinalIgnoreCase)).ToList() For Each row As BOMRow In view.BOMRows Try If row.ComponentDefinitions.Count = 0 Then Continue For Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim caminhoAtual As String = subPath & "\" & doc.DisplayName Dim itemNumberCorreto As String = mapaFiltrado \_ .Where(Function(t) t.Item1 = partNumber AndAlso t.Item3 = caminhoAtual) \_ .Select(Function(t) t.Item2) \_ .FirstOrDefault() If Not String.IsNullOrWhiteSpace(itemNumberCorreto) AndAlso row.ItemNumber <> itemNumberCorreto Then row.ItemNumber = itemNumberCorreto End If Catch ' Ignora erro na linha End Try Next Catch ex As Exception ' Ignora erro geral da submontagem End Try End Sub ' === Sincroniza recursivamente submontagens (exceto a raiz) === Private Sub SincronizarSubmontagensRecursivo(asm As AssemblyDocument, mapaGlobal As List(Of Tuple(Of String, String, String)), raizDisplayName As String) Try For Each occ As ComponentOccurrence In asm.ComponentDefinition.Occurrences If occ.DefinitionDocumentType = DocumentTypeEnum.kAssemblyDocumentObject Then Try Dim subDoc As Document = Nothing Try subDoc = ThisApplication.Documents.Item(occ.Definition.Document.FullFileName) Catch subDoc = ThisApplication.Documents.Open(occ.Definition.Document.FullFileName, False) End Try Dim subAsm As AssemblyDocument = TryCast(subDoc, AssemblyDocument) If subAsm Is Nothing Then Continue For Dim subPath As String = raizDisplayName & "\" & subAsm.DisplayName SincronizarItemNumbers(mapaGlobal, subAsm, subPath) Try : subAsm.Save2(True) : Catch : End Try ' Chamada recursiva para sub-submontagens SincronizarSubmontagensRecursivo(subAsm, mapaGlobal, raizDisplayName) Catch ' Ignora falha em submontagem individual End Try End If Next Catch ' Ignora falha geral End Try End Sub

**O ChatGPT disse:**

Excelente! Você pode sim **copiar esse código iLogic completo** para sua classe ThisRule no Add-In — mas será necessário fazer **pequenas adaptações** para que ele funcione corretamente no contexto de um Add-In (fora do ambiente do iLogic).

Vou te mostrar exatamente **como adaptar isso** passo a passo e te dou o código final pronto para colar.

**✅ 1. Estrutura da classe**ThisRule

Você já criou a classe ThisRule que herda de AbstractRule. Agora vamos **substituir o método**Main() por essa lógica do iLogic, **ajustando alguns pontos**.

**🔧 2. Pequenas adaptações necessárias**

**🟨 A. Substituir**Sub Main()**por**Public Sub Main()

Como o Main() será chamado pelo botão do Add-In, ele precisa ser Public.

**🟨 B. Substituir**Return**dentro de**Sub

O comando Return em VB.NET dentro de Sub não termina o método como em funções. Use Exit Sub no lugar:

Return → Exit Sub

**🟨 C.**ThisApplication**já está disponível na classe base**

Graças à AbstractRule, você pode usar ThisApplication direto — **sem alterações**.

**✅ 3. Código final da**ThisRule.vb**adaptada para o Add-In**

Aqui está sua classe completa **100% adaptada** para usar como Add-In:

Imports Inventor Imports System.Windows.Forms Imports System.Text Imports System.Linq Public Class ThisRule Inherits AbstractRule Public Sub Main() Dim doc As Document = ThisApplication.ActiveDocument If doc Is Nothing OrElse doc.DocumentType <> DocumentTypeEnum.kAssemblyDocumentObject Then MessageBox.Show("Abra uma montagem antes de rodar esta regra.", "Documento inválido", MessageBoxButtons.OK, MessageBoxIcon.Warning) Exit Sub End If Dim asmDoc As AssemblyDocument = CType(doc, AssemblyDocument) *' Obter o caminho base da montagem principal* Dim raizDisplayName As String = asmDoc.DisplayName *' Obter o mapa completo da estrutura, incluindo item numbers* Dim mapaGlobal As List(Of Tuple(Of String, String, String)) = ObterMapaCompleto(asmDoc) If mapaGlobal.Count = 0 Then MessageBox.Show("Nenhum item encontrado no BOM estruturado da montagem.", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information) Exit Sub End If *' IMPORTANTE: NÃO MODIFICAR A MONTAGEM PRINCIPAL!* SincronizarSubmontagensRecursivo(asmDoc, mapaGlobal, raizDisplayName) *' Exibir resultado* ShowFormattedResults(FormatarMapaParaTexto(mapaGlobal)) MessageBox.Show("Sincronização concluída com sucesso! A montagem principal foi preservada.", "Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information) End Sub Private Sub ShowFormattedResults(message As String) Dim form As New Form With {.Text = "RESULTADO DA SINCRONIZAÇÃO", .Width = 1000, .Height = 600} Dim richText As New RichTextBox With {.Dock = DockStyle.Fill, .ReadOnly = True} richText.AppendText("MAPA DE SINCRONIZAÇÃO:" & vbCrLf & vbCrLf & message) form.Controls.Add(richText) form.ShowDialog() End Sub Private Function FormatarMapaParaTexto(mapa As List(Of Tuple(Of String, String, String))) As String Dim sb As New StringBuilder() Dim col1Width As Integer = 70 *' PART NUMBER* Dim col2Width As Integer = 30 *' ITEM NUMBER* sb.AppendLine("PART NUMBER".PadRight(col1Width) & "ITEM NUMBER".PadRight(col2Width) & "CAMINHO HIERÁRQUICO") sb.AppendLine(New String("-"c, col1Width + col2Width + 70)) For Each item In mapa sb.AppendLine(item.Item1.PadRight(col1Width) & item.Item2.PadRight(col2Width) & item.Item3) Next Return sb.ToString() End Function Private Function ObterMapaCompleto(asm As AssemblyDocument) As List(Of Tuple(Of String, String, String)) Dim mapa As New List(Of Tuple(Of String, String, String))() Try Dim bom As BOM = asm.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = False Dim view As BOMView = bom.BOMViews.Cast(Of BOMView)().FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return mapa For Each row As BOMRow In view.BOMRows AdicionarAoMapaRecursivo(row, mapa, asm.DisplayName) Next Catch ex As Exception MessageBox.Show("Erro ao construir mapa completo: " & ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error) End Try Return mapa End Function Private Sub AdicionarAoMapaRecursivo(row As BOMRow, mapa As List(Of Tuple(Of String, String, String)), parentPath As String) Try If row.ComponentDefinitions.Count = 0 Then Exit Sub Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim itemNumber As String = row.ItemNumber Dim caminho As String = parentPath & "\" & doc.DisplayName If Not String.IsNullOrEmpty(partNumber) AndAlso Not String.IsNullOrEmpty(itemNumber) Then mapa.Add(New Tuple(Of String, String, String)(partNumber, itemNumber, caminho)) End If If row.ChildRows IsNot Nothing Then For Each child In row.ChildRows AdicionarAoMapaRecursivo(child, mapa, caminho) Next End If Catch *' Ignora erros individuais* End Try End Sub Private Function ObterMapaLocal(assemblyDoc As AssemblyDocument) As Dictionary(Of String, String) Dim mapa As New Dictionary(Of String, String)(StringComparer.OrdinalIgnoreCase) Try Dim bom As BOM = assemblyDoc.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = True Dim view As BOMView = bom.BOMViews.Cast(Of BOMView)().FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return mapa For Each row As BOMRow In view.BOMRows Try If row.ComponentDefinitions.Count = 0 Then Continue For Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim itemNumber As String = row.ItemNumber If Not mapa.ContainsKey(partNumber) AndAlso Not String.IsNullOrEmpty(itemNumber) Then mapa.Add(partNumber, itemNumber) End If Catch Continue For End Try Next Catch Return New Dictionary(Of String, String)(StringComparer.OrdinalIgnoreCase) End Try Return mapa End Function Private Sub SincronizarItemNumbers(mapaGlobal As List(Of Tuple(Of String, String, String)), asm As AssemblyDocument, subPath As String) Try Dim bom As BOM = asm.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = True Dim view As BOMView = bom.BOMViews.Cast(Of BOMView)().FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return Dim mapaFiltrado = mapaGlobal.Where(Function(t) t.Item3.StartsWith(subPath & "\", StringComparison.OrdinalIgnoreCase)).ToList() For Each row As BOMRow In view.BOMRows Try If row.ComponentDefinitions.Count = 0 Then Continue For Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim caminhoAtual As String = subPath & "\" & doc.DisplayName Dim itemNumberCorreto As String = mapaFiltrado \_ .Where(Function(t) t.Item1 = partNumber AndAlso t.Item3 = caminhoAtual) \_ .Select(Function(t) t.Item2) \_ .FirstOrDefault() If Not String.IsNullOrWhiteSpace(itemNumberCorreto) AndAlso row.ItemNumber <> itemNumberCorreto Then row.ItemNumber = itemNumberCorreto End If Catch *' Ignora erro na linha* End Try Next Catch *' Ignora erro geral da submontagem* End Try End Sub Private Sub SincronizarSubmontagensRecursivo(asm As AssemblyDocument, mapaGlobal As List(Of Tuple(Of String, String, String)), raizDisplayName As String) Try For Each occ As ComponentOccurrence In asm.ComponentDefinition.Occurrences If occ.DefinitionDocumentType = DocumentTypeEnum.kAssemblyDocumentObject Then Try Dim subDoc As Document = Nothing Try subDoc = ThisApplication.Documents.Item(occ.Definition.Document.FullFileName) Catch subDoc = ThisApplication.Documents.Open(occ.Definition.Document.FullFileName, False) End Try Dim subAsm As AssemblyDocument = TryCast(subDoc, AssemblyDocument) If subAsm Is Nothing Then Continue For Dim subPath As String = raizDisplayName & "\" & subAsm.DisplayName SincronizarItemNumbers(mapaGlobal, subAsm, subPath) Try : subAsm.Save2(True) : Catch : End Try SincronizarSubmontagensRecursivo(subAsm, mapaGlobal, raizDisplayName) Catch *' Ignora falha em submontagem individual* End Try End If Next Catch *' Ignora falha geral* End Try End Sub End Class

**✅ Próximos passos**

Agora você pode:

1. Salvar esse código no arquivo ThisRule.vb
2. Compilar o projeto
3. Executar no Inventor (através do botão)
4. A regra será executada normalmente como se estivesse no iLogic

Se quiser adicionar suporte a iProperties.Value(...) ou Logger.Info(...), posso te ajudar a criar wrappers parecidos com o ambiente do iLogic também.

Me avise se quiser isso!

**Você disse:**

ThisRule.vb é um arquivo de classe ?

**O ChatGPT disse:**

Sim! ✅  
ThisRule.vb deve ser um **arquivo de classe padrão** no seu projeto **VB.NET**.

**📌 Como criar corretamente**ThisRule.vb

Se você estiver usando o **Visual Studio**, siga este passo a passo:

**🛠️ Passo a passo (Visual Studio)**

1. **Clique com o botão direito** no seu projeto (normalmente no **Solution Explorer**)
2. Selecione **Add (Adicionar)** → **Class (Classe...)**
3. Dê o nome:
4. ThisRule.vb
5. Apague o conteúdo padrão e **cole o código adaptado que te enviei**.
6. Certifique-se de que ele **está no mesmo namespace**, se você estiver usando um.

**🎯 Resultado**

Seu arquivo ThisRule.vb será reconhecido pelo Visual Studio como parte do seu projeto e poderá ser chamado corretamente dentro da classe MyButton, onde temos:

Dim rule As New ThisRule() rule.ThisApplication = \_inventor rule.Main()

Isso garante que o botão do Add-In execute a regra iLogic embutida.

Se você ainda estiver em dúvida sobre onde criar esse arquivo ou como organizar as classes, posso te mostrar uma estrutura de projeto recomendada. Deseja isso?

**Você disse:**

acrdito que a dll nao esta exposta para o inventor é necessario compila-la em 64 bitis ?

**O ChatGPT disse:**

Sim, **é necessário compilar sua DLL como 64 bits (x64)** para que ela funcione corretamente com o Autodesk Inventor, porque:

**✅ O Inventor é um aplicativo somente 64 bits**

Todas as versões recentes do Autodesk Inventor (inclusive 2020 até 2025) rodam exclusivamente em 64 bits. Portanto:

❗ Se sua DLL estiver compilada como **AnyCPU** ou **x86**, o Inventor **não conseguirá carregá-la**.

**🛠️ Como garantir que a DLL seja compilada em 64 bits (Visual Studio)**

1. No Visual Studio, vá em:
   * Menu **Build** → **Configuration Manager...**
2. Em **Active Solution Platform**, selecione **<New...>** e escolha:
   * **x64**
   * Copie as configurações de: x86 (ou AnyCPU)
   * Clique em **OK**
3. Depois disso, vá nas **Propriedades do Projeto**:
   * **Build** → **Platform target** → Selecione **x64**
   * (Marque "Prefer 32-bit" **desmarcado**)

**🔁 Recomendações adicionais**

* Certifique-se de que o **.NET Framework alvo** é **compatível com o Inventor** (por exemplo, .NET Framework 4.7 ou 4.8)
* Compile a DLL em **modo Release**
* Verifique se o caminho do arquivo .addin aponta corretamente para a DLL compilada:

<Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\SeuPlugin.bundle\Contents\SeuPlugin.dll</Assembly>

**❓Quer confirmar se a DLL foi compilada corretamente?**

Você pode me mandar:

* O caminho da DLL
* A plataforma alvo usada (x64, x86, AnyCPU)
* O conteúdo do arquivo .addin

Posso te orientar para garantir que tudo esteja certo.

Quer ajuda nisso?

**Você disse:**

uso o visual studio em portugues

**O ChatGPT disse:**

Ótimo! Então aqui está o **passo a passo adaptado para o Visual Studio em português** para compilar sua DLL como **64 bits (x64)** — o que é **obrigatório** para funcionar no Autodesk Inventor.

**✅ Como compilar sua DLL como 64 bits no Visual Studio em português**

**1. Abra o Gerenciador de Soluções**

No **Visual Studio**, com seu projeto aberto:

* Vá no menu:  
  **Compilar → Gerenciador de Configurações...**

**2. Criar nova plataforma (se não existir)**

* Na coluna **Plataforma da Solução Ativa**, clique em **<Nova...>**
* Na janela que abrir:
  + **Nova plataforma:** selecione x64
  + **Copiar configurações de:** escolha Any CPU ou x86
  + Marque **"Criar novas configurações de projeto"** (se disponível)
  + Clique em **OK**

**3. Verificar configuração do projeto**

Agora com a solução configurada para x64:

1. Clique com o botão direito no **projeto** no Gerenciador de Soluções → **Propriedades**
2. Vá até a aba **Compilar**
3. Em **Destino da plataforma**, selecione:  
   x64
4. Certifique-se de que **"Preferir aplicativo de 32 bits"** está **desmarcado** (essa opção aparece apenas quando a plataforma é AnyCPU, mas é bom conferir).
5. Ainda nas propriedades, vá para a aba **Compilação** → marque para **Modo de Compilação: Release** (opcional, mas recomendado)

**🔁 Depois de compilar:**

* Verifique se a DLL gerada está no caminho que você indicou no seu arquivo .addin
* Exemplo:

<Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\SincronizarItem.bundle\Contents\SincronizarItem.dll</Assembly>

**❗ Dica de verificação rápida**

Para confirmar que sua DLL está em 64 bits:

1. Navegue até a pasta onde está o .dll compilado
2. Clique com o botão direito → **Propriedades**
3. Vá em **Detalhes**: infelizmente o Windows Explorer não mostra se é 64 bits diretamente, mas...

✅ Você pode usar uma ferramenta como **CorFlags** ou **PEStudio** para ver o bitness real da DLL, se precisar.

Se você quiser, posso te ajudar a verificar isso com base em seu .csproj ou vbproj.  
Quer me mostrar esse arquivo?

**Você disse:**

recapitulando MyButton.vb : Imports Inventor Imports System.Windows.Forms Public Class MyButton Private \_inventor As Inventor.Application Private \_settingsButton As ButtonDefinition Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub Private Sub SetupButtonDefinition() Dim conDefs As ControlDefinitions = \_inventor.CommandManager.ControlDefinitions ' Use o mesmo GUID fixo do seu .addin \_settingsButton = conDefs.AddButtonDefinition( "Sincronizar", "SincronizarButton", CommandTypesEnum.kNonShapeEditCmdType, Guid:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Sincronizar dados com o sistema", ToolTipText:="Clique para sincronizar") AddHandler \_settingsButton.OnExecute, AddressOf MyButton\_OnExecute End Sub Private Sub AddButtonDefinitionToRibbon() Try ' Ribbon Assembly Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") ' Aba Add-Ins Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_AddInsTab") ' Criar ou usar painel personalizado Dim ribbonPanel As RibbonPanel Try ribbonPanel = ribbonTab.RibbonPanels.Item("SincronizarPanel") Catch ex As Exception ' Se não existir, criar ribbonPanel = ribbonTab.RibbonPanels.Add("Sincronizar", "SincronizarPanel", "49b5d624-3533-42ed-ad63-5cf19250e8ad") End Try ribbonPanel.CommandControls.AddButton(\_settingsButton) Catch ex As Exception MsgBox("Erro ao adicionar botão: " & ex.Message) End Try End Sub Private Sub MyButton\_OnExecute(Context As NameValueMap) Try MessageBox.Show("Botão 'Sincronizar' clicado!", "Inventor Add-In") Catch ex As Exception MsgBox("Erro ao executar ação do botão: " & ex.Message) End Try End Sub End Class

**O ChatGPT disse:**

Perfeito! 👍 Você recapitulou corretamente a estrutura da sua classe MyButton.vb.

Aqui vai uma **revisão final** com observações e confirmação de que tudo está pronto para funcionar com o seu Add-In no Inventor:

**✅ Recapitulando a Estrutura da**MyButton.vb

**Referências no topo**

Imports Inventor Imports System.Windows.Forms

Ótimo — essas são necessárias para acessar a API do Inventor e a UI do Windows.

**Construtor**

Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub

✔️ Correto. Ao instanciar MyButton, o botão será criado e adicionado automaticamente à Ribbon.

**Criação do botão**

\_settingsButton = conDefs.AddButtonDefinition( "Sincronizar", "SincronizarButton", CommandTypesEnum.kNonShapeEditCmdType, Guid:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Sincronizar dados com o sistema", ToolTipText:="Clique para sincronizar")

* 🟢 **Nome visível:** Sincronizar
* 🟢 **Nome interno:** "SincronizarButton" (precisa ser único)
* 🟢 **GUID:** fixo e **igual ao seu arquivo**.addin ✅
* 🟢 **Tipo de comando:** adequado (kNonShapeEditCmdType)
* 🟢 **Tooltip e descrição:** descritivos e claros

**Inserção na Ribbon**

Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_AddInsTab")

✔️ Está configurando o botão para aparecer em:

* **Ambiente:** Assembly
* **Aba:** Add-Ins (id\_AddInsTab)
* **Painel personalizado:** "SincronizarPanel"

Se o painel não existir, você o cria — isso é perfeito para evitar erros em outras máquinas.

**Executar ação**

Private Sub MyButton\_OnExecute(Context As NameValueMap) MessageBox.Show("Botão 'Sincronizar' clicado!", "Inventor Add-In") End Sub

✔️ Isso confirma que o botão foi clicado — ideal para testes antes de conectar à lógica real (ThisRule).

**✅ Próximos Passos**

Se ainda não fez, verifique o seguinte:

1. ✔️ **Classe**StandardAddInServer chama New MyButton(...) no método Activate
2. ✔️ Projeto compilado em **x64**
3. ✔️ Arquivo .addin corretamente preenchido e colocado em:
   * C:\ProgramData\Autodesk\ApplicationPlugins\SincronizarItem.bundle\Contents\SincronizarItem.addin
4. ✔️ Caminho do <Assembly> no .addin aponta para o .dll **compilado**
5. ✔️ DLL referenciando corretamente a **API do Inventor** (pasta Inventor.Interop.dll, se necessário)

**🧪 Testar**

Depois de tudo:

* Reinicie o Inventor
* Abra uma montagem
* Vá até a aba **Add-Ins** → Verifique se aparece o botão **Sincronizar**
* Clique no botão → Deve aparecer a mensagem:  
  "Botão 'Sincronizar' clicado!"

Se isso acontecer, seu Add-In está funcionando 🎉  
Depois, é só trocar a ação do botão para chamar ThisRule.Main() com a regra de sincronização.

Se quiser ajuda para conectar essa parte, posso revisar com você também. Deseja?

**Você disse:**

StandardAddInServer.vb Imports Inventor Imports System.Windows.Forms Public Class MyButton Private WithEvents \_buttonDef As ButtonDefinition Private \_application As Inventor.Application Public Sub New(app As Inventor.Application) \_application = app ' Cria um botão de comando Dim controlDefs As ControlDefinitions = \_application.CommandManager.ControlDefinitions \_buttonDef = controlDefs.AddButtonDefinition( DisplayName:="Meu Botão", InternalName:="SincronizarItem.MyButtonID", Classification:=CommandTypesEnum.kNonShapeEditCmdType, ClientId:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Clique para sincronizar item", ToolTipText:="Sincronizar dados do item", StandardIcon:=Nothing, LargeIcon:=Nothing, ButtonDisplay:=ButtonDisplayEnum.kDisplayTextInLearningMode ) ' Adiciona o botão ao Ribbon (exemplo: ambiente de peças - aba Tools) Dim ribbon As Ribbon = \_application.UserInterfaceManager.Ribbons("Part") Dim tab As RibbonTab = ribbon.RibbonTabs("id\_TabTools") ' Cria um painel novo ou usa um existente Dim panel As RibbonPanel = Nothing Try panel = tab.RibbonPanels.Item("SincronizarPanel") Catch ex As Exception panel = tab.RibbonPanels.Add("Sincronizar", "SincronizarPanel", "49b5d624-3533-42ed-ad63-5cf19250e8ad") End Try ' Adiciona o botão ao painel panel.CommandControls.AddButton(\_buttonDef) End Sub ' Evento de clique do botão Private Sub \_buttonDef\_OnExecute(Context As NameValueMap) Handles \_buttonDef.OnExecute MessageBox.Show("Botão 'Sincronizar' clicado!", "AddIn Inventor") End Sub End Class

**O ChatGPT disse:**

Obrigado por compartilhar o código completo da sua classe MyButton. Ela está **bem estruturada** e funcional — mas **pertence à parte visual da interface do botão**, e **não deveria estar no arquivo**StandardAddInServer.vb, como o nome do seu arquivo sugere.

**✅ Correção de estrutura**

Você provavelmente está misturando os papéis dos arquivos. Vamos organizar:

**📁 Organização correta dos arquivos**

**🔹**StandardAddInServer.vb

Esse arquivo é o **ponto de entrada do seu Add-In**, onde o Inventor chama seu código. Ele deve conter a classe StandardAddInServer e **apenas delegar a criação do botão**, como abaixo:

Imports Inventor Imports System.Runtime.InteropServices Imports System.Windows.Forms <Guid("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)> Public Class StandardAddInServer Implements ApplicationAddInServer Private \_myButton As MyButton Public Sub Activate(AddInSiteObject As ApplicationAddInSite, FirstTime As Boolean) Implements ApplicationAddInServer.Activate Try \_myButton = New MyButton(AddInSiteObject.Application) Catch ex As Exception MessageBox.Show("Erro ao ativar Add-In: " & ex.Message) End Try End Sub Public Sub Deactivate() Implements ApplicationAddInServer.Deactivate End Sub Public Sub ExecuteCommand(CommandID As Integer) Implements ApplicationAddInServer.ExecuteCommand End Sub Public ReadOnly Property Automation As Object Implements ApplicationAddInServer.Automation Get Return Nothing End Get End Property End Class

**🔹**MyButton.vb

Aqui sim vai todo o código que você postou:

Imports Inventor Imports System.Windows.Forms Public Class MyButton Private WithEvents \_buttonDef As ButtonDefinition Private \_application As Inventor.Application Public Sub New(app As Inventor.Application) \_application = app *' Cria um botão de comando* Dim controlDefs As ControlDefinitions = \_application.CommandManager.ControlDefinitions \_buttonDef = controlDefs.AddButtonDefinition( DisplayName:="Meu Botão", InternalName:="SincronizarItem.MyButtonID", Classification:=CommandTypesEnum.kNonShapeEditCmdType, ClientId:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Clique para sincronizar item", ToolTipText:="Sincronizar dados do item", StandardIcon:=Nothing, LargeIcon:=Nothing, ButtonDisplay:=ButtonDisplayEnum.kDisplayTextInLearningMode ) *' Adiciona o botão ao Ribbon (exemplo: ambiente de peças - aba Tools)* Dim ribbon As Ribbon = \_application.UserInterfaceManager.Ribbons("Part") Dim tab As RibbonTab = ribbon.RibbonTabs("id\_TabTools") *' Cria um painel novo ou usa um existente* Dim panel As RibbonPanel = Nothing Try panel = tab.RibbonPanels.Item("SincronizarPanel") Catch ex As Exception panel = tab.RibbonPanels.Add("Sincronizar", "SincronizarPanel", "49b5d624-3533-42ed-ad63-5cf19250e8ad") End Try *' Adiciona o botão ao painel* panel.CommandControls.AddButton(\_buttonDef) End Sub *' Evento de clique do botão* Private Sub \_buttonDef\_OnExecute(Context As NameValueMap) Handles \_buttonDef.OnExecute MessageBox.Show("Botão 'Sincronizar' clicado!", "AddIn Inventor") End Sub End Class

**✅ Checklist Final**

1. **Arquivo**.addin com o ClassId igual ao GUID usado nas classes.
2. **Compilação em 64 bits** (Plataforma: x64).
3. **Coloque a DLL compilada** no caminho correto, por exemplo:
4. C:\ProgramData\Autodesk\ApplicationPlugins\SincronizarItem.bundle\Contents\SincronizarItem.dll
5. A DLL contém:
   * StandardAddInServer.vb
   * MyButton.vb
   * Outras classes, como ThisRule, AbstractRule, etc.

**🧪 Teste**

Abra o **Inventor**, vá para o ambiente Part, na aba **Tools**, e veja se o botão **"Meu Botão"** aparece.

Se clicar e aparecer a mensagem "Botão 'Sincronizar' clicado!", está funcionando perfeitamente.

Se você quiser que o botão apareça em outro ambiente, como **Assembly > Add-Ins**, posso ajustar o código para isso também. Deseja isso?

**Você disse:**

meu addin esta certo ? <?xml version="1.0" encoding="utf-8"?> <Addin Type="Standard"> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName Language="1033">ItemNumberSync</DisplayName> <Description Language="1033">Sincroniza os Item Numbers nas submontagens</Description> <Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll</Assembly> <Class>SincronizarItem.StandardAddInServer</Class> <!-- ajuste conforme seu namespace --> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> <UserInterfaceVersion>1</UserInterfaceVersion> <LoadBehavior>4</LoadBehavior> </Addin>

**O ChatGPT disse:**

Seu arquivo .addin está **quase certo**, mas há **alguns pontos importantes** que precisam ser ajustados para garantir que o Inventor reconheça e carregue seu Add-In corretamente.

**✅ Revisão completa do seu arquivo**.addin

<?xml version="1.0" encoding="utf-8"?> <Addin Type="Standard"> <ClassId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClassId> <ClientId>{49b5d624-3533-42ed-ad63-5cf19250e8ad}</ClientId> <DisplayName Language="1033">ItemNumberSync</DisplayName> <Description Language="1033">Sincroniza os Item Numbers nas submontagens</Description> *<!-- ⚠️ Caminho deve usar barra invertida dupla (ou nenhuma aspa) -->* <Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll</Assembly> *<!-- ⚠️ Verifique se este namespace e classe estão corretos -->* <Class>SincronizarItem.StandardAddInServer</Class> <LoadOnStartUp>1</LoadOnStartUp> <UserUnloadable>1</UserUnloadable> <Hidden>0</Hidden> <UserInterfaceVersion>1</UserInterfaceVersion> <LoadBehavior>4</LoadBehavior> </Addin>

**⚠️ O que você deve verificar**

**1. Caminho do Assembly**

<Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll</Assembly>

* **Certifique-se que a DLL está nesse caminho.**
* **Evite usar aspas no valor**. Correto:

<Assembly>C:\ProgramData\Autodesk\ApplicationPlugins\Sincronizar Item\SincronizarItem.dll</Assembly>

* **O nome da pasta não deve ter espaços**, se puder evitar. Melhor seria:
* C:\ProgramData\Autodesk\ApplicationPlugins\SincronizarItem.bundle\

**2. Classe e Namespace**

<Class>SincronizarItem.StandardAddInServer</Class>

* Isso deve corresponder exatamente ao **namespace e nome da classe pública** no seu projeto:

No código:

Namespace SincronizarItem Public Class StandardAddInServer

Se não estiver usando namespace, mude para apenas:

<Class>StandardAddInServer</Class>

**3. GUID (ClassId / ClientId)**

* O mesmo GUID **deve aparecer**:
  + No .addin (como você fez).
  + No atributo da classe StandardAddInServer:

<Guid("49b5d624-3533-42ed-ad63-5cf19250e8ad"), ComVisible(True)>

**🧪 Verificação final (checklist)**

| **Item** | **Verifique** |
| --- | --- |
| .dll está no caminho do <Assembly> | ✅ |
| ClassId e ClientId batem com o GUID no código | ✅ |
| <Class> corresponde ao namespace + classe | ✅ ou corrija conforme seu código |
| Projeto está compilado para x64 | ✅ (muito importante para o Inventor) |
| O .addin está salvo em local válido | C:\Users\<usuário>\AppData\Roaming\Autodesk\Inventor <versão>\Addins **ou** dentro de um .bundle |
| Inventor está fechado e reaberto após instalação | ✅ |

**🔁 Se quiser usar um**.bundle

Recomendo fortemente colocar seu Add-In dentro de uma estrutura .bundle, como:

C:\ProgramData\Autodesk\ApplicationPlugins\SincronizarItem.bundle\

│

├── Contents\

│ └── SincronizarItem.dll

│

└── PackageContents.xml

Se quiser, posso gerar o PackageContents.xml para você.

Se quiser que eu valide 100% com base no seu projeto atual, me envie:

* O namespace exato do seu StandardAddInServer
* O caminho onde colocou a .dll
* Onde você salvou o .addin

Assim te dou uma resposta final com precisão.

**Você disse:**

ThisRule.vb : Imports Inventor Imports System.Windows.Forms Imports System.Text Imports System.Linq Public Class ThisRule Inherits AbstractRule Public Sub Main() Dim doc As Document = ThisApplication.ActiveDocument If doc Is Nothing OrElse doc.DocumentType <> DocumentTypeEnum.kAssemblyDocumentObject Then MessageBox.Show("Abra uma montagem antes de rodar esta regra.", "Documento inválido", MessageBoxButtons.OK, MessageBoxIcon.Warning) Exit Sub End If Dim asmDoc As AssemblyDocument = CType(doc, AssemblyDocument) ' Obter o caminho base da montagem principal Dim raizDisplayName As String = asmDoc.DisplayName ' Obter o mapa completo da estrutura, incluindo item numbers Dim mapaGlobal As List(Of Tuple(Of String, String, String)) = ObterMapaCompleto(asmDoc) If mapaGlobal.Count = 0 Then MessageBox.Show("Nenhum item encontrado no BOM estruturado da montagem.", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information) Exit Sub End If ' IMPORTANTE: NÃO MODIFICAR A MONTAGEM PRINCIPAL! SincronizarSubmontagensRecursivo(asmDoc, mapaGlobal, raizDisplayName) ' Exibir resultado ShowFormattedResults(FormatarMapaParaTexto(mapaGlobal)) MessageBox.Show("Sincronização concluída com sucesso! A montagem principal foi preservada.", "Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information) End Sub Private Sub ShowFormattedResults(message As String) Dim form As New Form With {.Text = "RESULTADO DA SINCRONIZAÇÃO", .Width = 1000, .Height = 600} Dim richText As New RichTextBox With {.Dock = DockStyle.Fill, .ReadOnly = True} richText.AppendText("MAPA DE SINCRONIZAÇÃO:" & vbCrLf & vbCrLf & message) form.Controls.Add(richText) form.ShowDialog() End Sub Private Function FormatarMapaParaTexto(mapa As List(Of Tuple(Of String, String, String))) As String Dim sb As New StringBuilder() Dim col1Width As Integer = 70 ' PART NUMBER Dim col2Width As Integer = 30 ' ITEM NUMBER sb.AppendLine("PART NUMBER".PadRight(col1Width) & "ITEM NUMBER".PadRight(col2Width) & "CAMINHO HIERÁRQUICO") sb.AppendLine(New String("-"c, col1Width + col2Width + 70)) For Each item In mapa sb.AppendLine(item.Item1.PadRight(col1Width) & item.Item2.PadRight(col2Width) & item.Item3) Next Return sb.ToString() End Function Private Function ObterMapaCompleto(asm As AssemblyDocument) As List(Of Tuple(Of String, String, String)) Dim mapa As New List(Of Tuple(Of String, String, String))() Try Dim bom As BOM = asm.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = False Dim view As BOMView = bom.BOMViews.Cast(Of BOMView)().FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return mapa For Each row As BOMRow In view.BOMRows AdicionarAoMapaRecursivo(row, mapa, asm.DisplayName) Next Catch ex As Exception MessageBox.Show("Erro ao construir mapa completo: " & ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error) End Try Return mapa End Function Private Sub AdicionarAoMapaRecursivo(row As BOMRow, mapa As List(Of Tuple(Of String, String, String)), parentPath As String) Try If row.ComponentDefinitions.Count = 0 Then Exit Sub Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim itemNumber As String = row.ItemNumber Dim caminho As String = parentPath & "\" & doc.DisplayName If Not String.IsNullOrEmpty(partNumber) AndAlso Not String.IsNullOrEmpty(itemNumber) Then mapa.Add(New Tuple(Of String, String, String)(partNumber, itemNumber, caminho)) End If If row.ChildRows IsNot Nothing Then For Each child In row.ChildRows AdicionarAoMapaRecursivo(child, mapa, caminho) Next End If Catch ' Ignora erros individuais End Try End Sub Private Function ObterMapaLocal(assemblyDoc As AssemblyDocument) As Dictionary(Of String, String) Dim mapa As New Dictionary(Of String, String)(StringComparer.OrdinalIgnoreCase) Try Dim bom As BOM = assemblyDoc.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = True Dim view As BOMView = bom.BOMViews.Cast(Of BOMView)().FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return mapa For Each row As BOMRow In view.BOMRows Try If row.ComponentDefinitions.Count = 0 Then Continue For Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim itemNumber As String = row.ItemNumber If Not mapa.ContainsKey(partNumber) AndAlso Not String.IsNullOrEmpty(itemNumber) Then mapa.Add(partNumber, itemNumber) End If Catch Continue For End Try Next Catch Return New Dictionary(Of String, String)(StringComparer.OrdinalIgnoreCase) End Try Return mapa End Function Private Sub SincronizarItemNumbers(mapaGlobal As List(Of Tuple(Of String, String, String)), asm As AssemblyDocument, subPath As String) Try Dim bom As BOM = asm.ComponentDefinition.BOM bom.StructuredViewEnabled = True bom.StructuredViewFirstLevelOnly = True Dim view As BOMView = bom.BOMViews.Cast(Of BOMView)().FirstOrDefault(Function(v) v.ViewType = BOMViewTypeEnum.kStructuredBOMViewType) If view Is Nothing Then Return Dim mapaFiltrado = mapaGlobal.Where(Function(t) t.Item3.StartsWith(subPath & "\", StringComparison.OrdinalIgnoreCase)).ToList() For Each row As BOMRow In view.BOMRows Try If row.ComponentDefinitions.Count = 0 Then Continue For Dim doc As Document = row.ComponentDefinitions.Item(1).Document Dim partNumber As String = doc.PropertySets.Item("Design Tracking Properties").Item("Part Number").Value.ToString() Dim caminhoAtual As String = subPath & "\" & doc.DisplayName Dim itemNumberCorreto As String = mapaFiltrado \_ .Where(Function(t) t.Item1 = partNumber AndAlso t.Item3 = caminhoAtual) \_ .Select(Function(t) t.Item2) \_ .FirstOrDefault() If Not String.IsNullOrWhiteSpace(itemNumberCorreto) AndAlso row.ItemNumber <> itemNumberCorreto Then row.ItemNumber = itemNumberCorreto End If Catch ' Ignora erro na linha End Try Next Catch ' Ignora erro geral da submontagem End Try End Sub Private Sub SincronizarSubmontagensRecursivo(asm As AssemblyDocument, mapaGlobal As List(Of Tuple(Of String, String, String)), raizDisplayName As String) Try For Each occ As ComponentOccurrence In asm.ComponentDefinition.Occurrences If occ.DefinitionDocumentType = DocumentTypeEnum.kAssemblyDocumentObject Then Try Dim subDoc As Document = Nothing Try subDoc = ThisApplication.Documents.Item(occ.Definition.Document.FullFileName) Catch subDoc = ThisApplication.Documents.Open(occ.Definition.Document.FullFileName, False) End Try Dim subAsm As AssemblyDocument = TryCast(subDoc, AssemblyDocument) If subAsm Is Nothing Then Continue For Dim subPath As String = raizDisplayName & "\" & subAsm.DisplayName SincronizarItemNumbers(mapaGlobal, subAsm, subPath) Try : subAsm.Save2(True) : Catch : End Try SincronizarSubmontagensRecursivo(subAsm, mapaGlobal, raizDisplayName) Catch ' Ignora falha em submontagem individual End Try End If Next Catch ' Ignora falha geral End Try End Sub End Class

**O ChatGPT disse:**

Perfeito — você estruturou muito bem a classe ThisRule e o código que veio do iLogic foi adaptado corretamente. Vou fazer uma **revisão técnica completa** para garantir que tudo esteja pronto para funcionar dentro do seu **Add-In Inventor**.

**✅ Revisão Geral da Classe**ThisRule.vb

**✔ Estrutura básica:**

* A classe **herda de**AbstractRule, como planejado.
* A propriedade ThisApplication será injetada via StandardAddInServer → OK.
* Main() funciona como ponto de entrada da regra → correto.

**✔ Namespaces:**

* Imports Inventor, Imports System.Windows.Forms, Imports System.Linq, Imports System.Text → tudo certo.

**🧠 Pontos Fortes**

* **Separação clara das responsabilidades**: ObterMapaCompleto, SincronizarItemNumbers, ShowFormattedResults — modularizado, limpo.
* **Uso correto da BOM estruturada (**kStructuredBOMViewType**)** — ótimo.
* **Tratamento de erros com**Try/Catch**individualizado** — muito boa prática.
* **Uso do**RichTextBox**para exibir resultados** — bem feito e útil ao usuário.

**⚠️ Pontos para verificar antes de compilar o Add-In final**

**1.**AbstractRule.vb**está no projeto?**

A classe ThisRule depende da base AbstractRule, que **deve estar presente e pública**:

*' AbstractRule.vb* Public MustInherit Class AbstractRule Public Property ThisApplication As Inventor.Application Public ReadOnly Property ThisDoc As ICadDoc Get Return New CadDoc(ThisApplication.ActiveDocument) End Get End Property End Class

E a interface ICadDoc / classe CadDoc também precisam estar acessíveis no projeto (ou você precisa remover o uso de ThisDoc se não estiver usando de fato).

**2. Você está chamando**ThisRule.Main()**em algum lugar?**

Essa regra **não será executada automaticamente** se você não chamar Main() de dentro do clique no botão, por exemplo:

*' Dentro do evento do botão no StandardAddInServer ou MyButton.vb* Dim regra As New ThisRule() regra.ThisApplication = \_application *' ou ThisApplication do contexto atual* regra.Main()

Sem isso, a lógica da regra **nunca será executada**, mesmo que o Add-In seja carregado.

**3. Projeto está sendo compilado como DLL 64 bits?**

* **Sim** → Inventor exige compilação **x64**.
* Vá em Propriedades do Projeto > Compilar > Plataforma de destino: x64

**✅ Conclusão**

**🟢 Sua classe**ThisRule.vb**está correta e pronta para funcionar, desde que:**

* Você esteja instanciando e chamando Main() dela em seu botão.
* A AbstractRule.vb esteja presente e correta.
* O projeto esteja em **x64**.
* O arquivo .addin esteja apontando corretamente para a .dll.

Se quiser, posso te ajudar com:

* A chamada correta da regra no clique do botão.
* Verificar se tudo está amarrado no StandardAddInServer.
* Gerar estrutura .bundle com PackageContents.xml se quiser distribuir.

É só me dizer.

**Você disse:**

Observe a linha 4, onde dizemos ao compilador para herdar a classe base. Em regras mais complexas, você terá mais funções ou até mesmo uma classe inteira. Nesse caso, talvez seja necessário adicionar também código fora da função principal. Para finalizar este tutorial com um add-in funcional/funcional, substituiremos o código da classe pela regra " Uma Regra para Pesquisar Todos ". A classe "ThisRule" ficará assim:

**O ChatGPT disse:**

Ótimo! Vamos fazer isso passo a passo para que você finalize seu **add-in funcional**, substituindo a classe ThisRule por uma versão mais completa.

**✅ O que é a "Regra para Pesquisar Todos"?**

Trata-se de uma regra de iLogic adaptada para um **Add-In do Inventor** que percorre **todos os arquivos referenciados**, seja para:

* validar propriedades,
* coletar informações,
* ou preparar algum tipo de relatório/exportação em massa.

**🧩 Substituição da Classe**ThisRule

Aqui está um exemplo completo de como **a nova versão da classe**ThisRule pode parecer, baseada no conceito de “Pesquisar todos os componentes da árvore de montagem” — adaptável ao seu add-in:

**🆕**ThisRule.vb**— versão "Pesquisar Todos"**

Imports Inventor Imports System.Windows.Forms Imports System.Text Imports System.IO Public Class ThisRule Inherits AbstractRule Public Sub Main() Dim doc As Document = ThisApplication.ActiveDocument If doc Is Nothing OrElse doc.DocumentType <> DocumentTypeEnum.kAssemblyDocumentObject Then MessageBox.Show("Abra uma montagem antes de rodar esta regra.", "Documento inválido", MessageBoxButtons.OK, MessageBoxIcon.Warning) Exit Sub End If Dim asmDoc As AssemblyDocument = CType(doc, AssemblyDocument) Dim sb As New StringBuilder() sb.AppendLine("COMPONENTES DA MONTAGEM:") sb.AppendLine(New String("="c, 80)) *' Percorre todos os componentes* For Each occ As ComponentOccurrence In asmDoc.ComponentDefinition.Occurrences.AllReferencedOccurrences Dim partNumber As String = "" Dim filePath As String = "" Try Dim refDoc As Document = occ.Definition.Document partNumber = GetPropertySafe(refDoc, "Design Tracking Properties", "Part Number") filePath = refDoc.FullFileName Catch ex As Exception partNumber = "(Erro ao obter)" filePath = "(desconhecido)" End Try sb.AppendLine($"DisplayName: {occ.Name} | PartNumber: {partNumber} | Caminho: {filePath}") Next *' Mostra os dados em uma janela* ShowResults(sb.ToString()) End Sub Private Sub ShowResults(content As String) Dim form As New Form With { .Text = "Pesquisa Completa - Resultados", .Width = 900, .Height = 600 } Dim txtBox As New RichTextBox With { .Dock = DockStyle.Fill, .ReadOnly = True, .Text = content, .Font = New Font("Consolas", 10) } form.Controls.Add(txtBox) form.ShowDialog() End Sub Private Function GetPropertySafe(doc As Document, propSetName As String, propName As String) As String Try Return doc.PropertySets.Item(propSetName).Item(propName).Value.ToString() Catch Return "(não definido)" End Try End Function End Class

**📝 O que essa versão faz:**

* Garante que o documento aberto é uma montagem.
* Percorre **todos os componentes referenciados** (inclusive submontagens).
* Coleta:
  + Part Number
  + Caminho do arquivo
  + Nome de exibição da ocorrência
* Exibe tudo num RichTextBox formatado (como no estilo iLogic).

**💡 Dicas finais**

* Esse modelo pode ser facilmente adaptado para:
  + verificar propriedades obrigatórias,
  + gerar listas para Excel,
  + exportar PDFs/dxf/dwg em lote,
  + ou gerar BOM customizados.
* O botão que executa essa classe deve chamar:

Dim regra As New ThisRule() regra.ThisApplication = \_application regra.Main()

Se quiser, posso te entregar um projeto completo .vbproj com isso tudo amarrado ou só revisar sua StandardAddInServer.vb para garantir que tudo está bem conectado.

Deseja seguir com isso?

**Você disse:**

nao estou conseguindo relaizar que o addin seja adiconado no inventor, tenho um exemplo de arquivo xml que funcionou para outro addin . <?xml version="1.0" encoding="utf-8"?> <ApplicationPackage SchemaVersion="1.0" AutodeskProduct="Inventor" Name="ItemNumberSync" Description="Sincroniza Item Numbers em submontagens" AppVersion="1.0" ProductCode="{bd2ec21a-725f-415f-826e-309a6ed74dde}" Author="DOMA" Company="DOMA" HelpUrl="https://suaempresa.com" SupportedLocales="en-US;pt-BR"> <CompanyDetails Name="DOMA" Url="https://suaempresa.com" Email="suporte@suaempresa.com" /> <Components Description="Win64"> <RuntimeRequirements OS="Win64" Platform="Inventor|AIP" SeriesMin="Ir24" SeriesMax="Ir29" /> <ComponentEntry AppName="ItemNumberSync" Version="1.0" ModuleName="./Contents/Win64/ItemNumberAddIn.dll" AppDescription="Sincroniza os Item Numbers nas submontagens" /> </Components> </ApplicationPackage>

**O ChatGPT disse:**

Você trouxe um arquivo XML diferente, que parece ser de um **bundle/autoloader** (ApplicationPackage) — esse formato também é válido para o Inventor, usado especialmente para distribuir Add‑Ins via *ApplicationPlugins*.

Esse é um modelo bom, mas precisa ser adaptado corretamente para que seu Add‑In seja reconhecido no Inventor. Vamos ver os pontos principais e o que ajustar.

**🔍 O que é esse XML de exemplo**

O arquivo que você postou:

<ApplicationPackage SchemaVersion="1.0" AutodeskProduct="Inventor" Name="ItemNumberSync" Description="Sincroniza Item Numbers em submontagens" AppVersion="1.0" ProductCode="{bd2ec21a-725f-415f-826e-309a6ed74dde}" Author="DOMA" Company="DOMA" HelpUrl="https://suaempresa.com" SupportedLocales="en-US;pt-BR"> <CompanyDetails Name="DOMA" Url="https://suaempresa.com" Email="suporte@suaempresa.com" /> <Components Description="Win64"> <RuntimeRequirements OS="Win64" Platform="Inventor|AIP" SeriesMin="Ir24" SeriesMax="Ir29" /> <ComponentEntry AppName="ItemNumberSync" Version="1.0" ModuleName="./Contents/Win64/ItemNumberAddIn.dll" AppDescription="Sincroniza os Item Numbers nas submontagens" /> </Components> </ApplicationPackage>

Este XML é **do tipo bundle/autoloader** — ele descreve um pacote que o Inventor lê para carregar o Add‑In. Informatizado com campos como AutodeskProduct="Inventor", requisitos, caminho do módulo, etc.

**✅ Pontos que você precisa adaptar e verificar**

Para que esse formato funcione bem, você precisa garantir:

1. **Estrutura de pastas correta para o bundle**

Normalmente, você teria algo como:

ItemNumberSync.bundle/

PackageContents.xml ← este XML

Contents/

Win64/

ItemNumberAddIn.dll

ItemNumberAddIn.Inventor.addin ← seu manifesto .addin

Ou algo parecido, conforme especificado na tag ModuleName="./Contents/Win64/ItemNumberAddIn.dll". Ou seja, o DLL e o arquivo .addin devem estar localizados conforme referenciado.

1. **Manifesto .addin dentro do bundle**

Dentro da pasta Contents/Win64/ você deve incluir um arquivo .addin que aponta para sua DLL, com GUID, etc.

1. **ProductCode vs GUID de classe**

O ProductCode no ApplicationPackage pode ser diferente do GUID da sua classe StandardAddInServer.  
O GUID da classe (o que você usa no atributo <Guid(...)> no código e no .addin) é usado internamente pelo Inventor para instanciar o Add‑In. O ProductCode é para identificar o pacote no sistema, especialmente se for distribuído ou registrado no AppStore.

1. **Requisitos de versão**
   * RuntimeRequirements OS="Win64"
   * Platform="Inventor|AIP" ou semelhante (deve compatibilizar com suas versões do Inventor)
   * SeriesMin e SeriesMax definem as versões do Inventor compatíveis. Se você quer que funcione em 2020 e 2025, certifique-se de que os ranges incluam essas versões.

**✅ Sugestão adaptada para seu Add-In**

Aqui vai uma versão adaptada com base no seu projeto (você pode ajustar caminhos, nomes e GUID conforme suas classes):

<?xml version="1.0" encoding="utf-8"?> <ApplicationPackage SchemaVersion="1.0" AutodeskProduct="Inventor" Name="SincronizarItem" Description="Sincroniza Item Numbers em submontagens" AppVersion="1.0" ProductCode="{49b5d624-3533-42ed-ad63-5cf19250e8ad}" Author="Sua Empresa" Company="Sua Empresa" HelpUrl="https://suaempresa.com" SupportedLocales="pt-BR;en-US"> <CompanyDetails Name="Sua Empresa" Url="https://suaempresa.com" Email="suporte@suaempresa.com" /> <Components Description="Win64"> <RuntimeRequirements OS="Win64" Platform="Inventor" SeriesMin="Ir20" SeriesMax="Ir25" /> <ComponentEntry AppName="SincronizarItem" Version="1.0" ModuleName="./Contents/Win64/SincronizarItem.dll" AppDescription="Sincroniza Item Numbers nas submontagens" /> </Components> </ApplicationPackage>

Depois dessa estrutura:

* Coloque o .addin manifesto dentro de Contents/Win64/ (ou pasta apropriada) com os dados corretos.
* Certifique-se de que seu DLL e o manifesto estão nessa estrutura e que o Inventor consegue “ver” o bundle.

Se quiser, posso te ajudar passo a passo a montar esse .bundle completo (pastas, .addin, XML) para que seu add-in seja carregado no Inventor sem erros. Quer seguir isso agora?

**Você disse:**

http://www.hjalte.nl/60-creating-an-inventor-addin

**O ChatGPT disse:**

O link que você enviou (http://www.hjalte.nl/60-creating-an-inventor-addin) está com erro interno (Internal Error) — não foi possível carregar a página. [Site Indisponível](https://chatgpt.com/)

Se quiser, posso procurar uma **versão arquivada** desse tutorial (via Wayback Machine, por exemplo), ou te indicar outro tutorial atualizado para criar Add-Ins do Inventor, com código-fonte, exemplos em VB.NET/C#, etc. Você prefere que eu ache uma versão arquivada ou que te envie um tutorial novo?

**Você disse:**

PictureDispConverter.vb mports System.Runtime.InteropServices Public NotInheritable Class PictureDispConverter <DllImport("OleAut32.dll", EntryPoint:="OleCreatePictureIndirect", ExactSpelling:=True, PreserveSig:=False)> Private Shared Function OleCreatePictureIndirect( <MarshalAs(UnmanagedType.AsAny)> ByVal picdesc As Object, ByRef iid As Guid, <MarshalAs(UnmanagedType.Bool)> ByVal fOwn As Boolean) As stdole.IPictureDisp End Function Shared iPictureDispGuid As Guid = GetType(stdole.IPictureDisp).GUID Private NotInheritable Class PICTDESC Private Sub New() End Sub Public Const PICTYPE\_UNINITIALIZED As Short = -1 Public Const PICTYPE\_NONE As Short = 0 Public Const PICTYPE\_BITMAP As Short = 1 Public Const PICTYPE\_METAFILE As Short = 2 Public Const PICTYPE\_ICON As Short = 3 Public Const PICTYPE\_ENHMETAFILE As Short = 4 <StructLayout(LayoutKind.Sequential)> Public Class Icon Friend cbSizeOfStruct As Integer = Marshal.SizeOf(GetType(PICTDESC.Icon)) Friend picType As Integer = PICTDESC.PICTYPE\_ICON Friend hicon As IntPtr = IntPtr.Zero Friend unused1 As Integer Friend unused2 As Integer Friend Sub New(ByVal icon As System.Drawing.Icon) Me.hicon = icon.ToBitmap().GetHicon() End Sub End Class <StructLayout(LayoutKind.Sequential)> Public Class Bitmap Friend cbSizeOfStruct As Integer = Marshal.SizeOf(GetType(PICTDESC.Bitmap)) Friend picType As Integer = PICTDESC.PICTYPE\_BITMAP Friend hbitmap As IntPtr = IntPtr.Zero Friend hpal As IntPtr = IntPtr.Zero Friend unused As Integer Friend Sub New(ByVal bitmap As System.Drawing.Bitmap) Me.hbitmap = bitmap.GetHbitmap() End Sub End Class End Class Public Shared Function ToIPictureDisp(ByVal icon As System.Drawing.Icon) As stdole.IPictureDisp Dim pictIcon As New PICTDESC.Icon(icon) Return OleCreatePictureIndirect(pictIcon, iPictureDispGuid, True) End Function Public Shared Function ToIPictureDisp(ByVal bmp As System.Drawing.Bitmap) As stdole.IPictureDisp Dim pictBmp As New PICTDESC.Bitmap(bmp) Return OleCreatePictureIndirect(pictBmp, iPictureDispGuid, True) End Function End Class

**O ChatGPT disse:**

O código que você compartilhou para PictureDispConverter.vb está correto e tem uma função bem específica: **converter imagens (**Icon**ou**Bitmap**) para**IPictureDisp, que é um formato usado por algumas APIs COM antigas — incluindo o **Autodesk Inventor** — para exibir ícones em botões de Add-Ins, como:

* ButtonDefinition.StandardIcon
* ButtonDefinition.LargeIcon

**✅ Finalidade da**PictureDispConverter

Autodesk Inventor exige que os ícones em botões sejam fornecidos como stdole.IPictureDisp. O .NET nativamente usa System.Drawing.Icon e Bitmap, então essa classe serve como **ponte de conversão**.

**🧩 Como usar o**PictureDispConverter

No seu código do botão (MyButton.vb ou StandardAddInServer.vb), em vez de passar Nothing para os ícones, você pode fazer algo assim:

Dim smallIcon As stdole.IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.icon16) Dim largeIcon As stdole.IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.icon32) \_buttonDef = controlDefs.AddButtonDefinition( DisplayName:="Sincronizar", InternalName:="SincronizarItem.MyButtonID", Classification:=CommandTypesEnum.kNonShapeEditCmdType, ClientId:="49b5d624-3533-42ed-ad63-5cf19250e8ad", DescriptionText:="Clique para sincronizar item", ToolTipText:="Sincroniza dados do item", StandardIcon:=smallIcon, LargeIcon:=largeIcon, ButtonDisplay:=ButtonDisplayEnum.kDisplayTextInLearningMode )

**📌 Observações importantes**

1. **Adicione os ícones aos recursos do projeto**  
   Coloque seus arquivos .ico ou .png na pasta Resources do projeto e configure como recurso incorporado.
2. **Atenção ao tamanho das imagens**
   * icon16: 16x16 pixels (ícone pequeno)
   * icon32: 32x32 pixels (ícone grande)
3. **A referência**stdole deve estar adicionada ao seu projeto:
   * Clique com o botão direito em **Referências** > **Adicionar referência...**
   * Vá em **Assemblies > COM** e adicione **"Microsoft Standard OLE Types" (**stdole**)**

Se quiser, posso te ajudar a configurar os ícones como recurso ou revisar o trecho onde você cria o botão para garantir que está usando corretamente o PictureDispConverter. Deseja isso?

**Você disse:**

Imports Inventor Public Class MyButton Private ReadOnly \_inventor As Inventor.Application Private \_settingsButton As ButtonDefinition Private \_control As CommandControl Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub Public Sub Unload() \_control.Delete() \_control = Nothing \_settingsButton.Delete() \_settingsButton = Nothing End Sub Private Sub SetupButtonDefinition() Dim conDefs As ControlDefinitions = \_inventor.CommandManager.ControlDefinitions \_settingsButton = conDefs.AddButtonDefinition( "MyButton DisplayName", "MyButton InternalName", CommandTypesEnum.kEditMaskCmdType, Guid.NewGuid().ToString(), "MyButton DescriptionText", "MyButton ToolTipText",) AddHandler \_settingsButton.OnExecute, AddressOf MyButton\_OnExecute \_settingsButton.StandardIcon = PictureDispConverter.ToIPictureDisp(My.Resources.MyImage16x16) \_settingsButton.LargeIcon = PictureDispConverter.ToIPictureDisp(My.Resources.MyImage32x32) End Sub Private Sub AddButtonDefinitionToRibbon() Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_TabManage") Dim ribbonPanel As RibbonPanel = ribbonTab.RibbonPanels.Item("iLogic.RibbonPanel") \_control = ribbonPanel.CommandControls.AddButton(\_settingsButton) End Sub Private Sub MyButton\_OnExecute(Context As NameValueMap) Try Dim mySearchForm As New MySearchForm(\_inventor) mySearchForm.ShowDialog() Catch ex As Exception MsgBox("Something went wrong while runing rule. Message: " & ex.Message) End Try End Sub End Class

**O ChatGPT disse:**

Seu código MyButton.vb está muito próximo de estar correto e funcional, mas há **alguns ajustes importantes** que precisam ser feitos para garantir que:

1. O botão seja adicionado corretamente à interface do Inventor.
2. O botão use ícones corretamente.
3. O botão possa ser removido (unloaded) sem erro.
4. O GUID seja persistente — e não aleatório — para evitar múltiplas instâncias do botão.

**✅ Correções e Melhorias**

**1. GUID Fixo**

No seu código, você está usando:

Guid.NewGuid().ToString()

Isso **gera um novo GUID toda vez**, o que **impede o Inventor de reconhecer o botão** corretamente entre sessões.  
✅ Use um GUID **fixo**, o mesmo que está no seu .addin ou .xml.

"49b5d624-3533-42ed-ad63-5cf19250e8ad"

**2. Nome Interno e DisplayName**

Esses nomes devem ser únicos e descritivos:

\_settingsButton = conDefs.AddButtonDefinition( "Sincronizar", "SincronizarItem.MyButton", CommandTypesEnum.kEditMaskCmdType, "49b5d624-3533-42ed-ad63-5cf19250e8ad", *' GUID fixo* "Sincroniza Item Numbers em submontagens", "Clique para sincronizar dados entre submontagens", PictureDispConverter.ToIPictureDisp(My.Resources.MyImage16x16), PictureDispConverter.ToIPictureDisp(My.Resources.MyImage32x32) )

✅ Assim, você já define os ícones diretamente no construtor, dispensando as duas linhas seguintes.

**3. Verificação de existência ao remover**

Na função Unload(), adicione verificação antes de deletar os objetos:

Public Sub Unload() If \_control IsNot Nothing Then \_control.Delete() \_control = Nothing End If If \_settingsButton IsNot Nothing Then \_settingsButton.Delete() \_settingsButton = Nothing End If End Sub

**4. Referência ao painel de destino**

Você está tentando adicionar o botão ao painel:

Dim ribbonPanel As RibbonPanel = ribbonTab.RibbonPanels.Item("iLogic.RibbonPanel")

⚠️ Esse painel "iLogic.RibbonPanel" **pode não existir** dependendo da instalação do Inventor.

✅ Solução: **crie um painel personalizado se ele não existir**:

Dim ribbonPanel As RibbonPanel Try ribbonPanel = ribbonTab.RibbonPanels.Item("ItemSyncPanel") Catch ribbonPanel = ribbonTab.RibbonPanels.Add("Sincronizar Itens", "ItemSyncPanel", "49b5d624-3533-42ed-ad63-5cf19250e8ad") End Try

**🔧 Código Ajustado**

Aqui está seu código revisado e corrigido:

Imports Inventor Imports System.Windows.Forms Public Class MyButton Private ReadOnly \_inventor As Inventor.Application Private \_settingsButton As ButtonDefinition Private \_control As CommandControl Public Sub New(inventor As Inventor.Application) \_inventor = inventor SetupButtonDefinition() AddButtonDefinitionToRibbon() End Sub Public Sub Unload() If \_control IsNot Nothing Then \_control.Delete() \_control = Nothing End If If \_settingsButton IsNot Nothing Then \_settingsButton.Delete() \_settingsButton = Nothing End If End Sub Private Sub SetupButtonDefinition() Dim conDefs As ControlDefinitions = \_inventor.CommandManager.ControlDefinitions \_settingsButton = conDefs.AddButtonDefinition( "Sincronizar", "SincronizarItem.MyButton", CommandTypesEnum.kEditMaskCmdType, "49b5d624-3533-42ed-ad63-5cf19250e8ad", "Sincroniza os Item Numbers nas submontagens", "Clique para sincronizar", PictureDispConverter.ToIPictureDisp(My.Resources.MyImage16x16), PictureDispConverter.ToIPictureDisp(My.Resources.MyImage32x32) ) AddHandler \_settingsButton.OnExecute, AddressOf MyButton\_OnExecute End Sub Private Sub AddButtonDefinitionToRibbon() Dim ribbon As Ribbon = \_inventor.UserInterfaceManager.Ribbons.Item("Assembly") Dim ribbonTab As RibbonTab = ribbon.RibbonTabs.Item("id\_TabManage") Dim ribbonPanel As RibbonPanel Try ribbonPanel = ribbonTab.RibbonPanels.Item("ItemSyncPanel") Catch ribbonPanel = ribbonTab.RibbonPanels.Add("Sincronizar Itens", "ItemSyncPanel", "49b5d624-3533-42ed-ad63-5cf19250e8ad") End Try \_control = ribbonPanel.CommandControls.AddButton(\_settingsButton) End Sub Private Sub MyButton\_OnExecute(Context As NameValueMap) Try *' Aqui você pode chamar sua UI, regra ou classe* Dim mySearchForm As New MySearchForm(\_inventor) mySearchForm.ShowDialog() Catch ex As Exception MsgBox("Erro ao executar a regra: " & ex.Message) End Try End Sub End Class

**🧪 Checklist final**

| **Item** | **Verificação** |
| --- | --- |
| ✔ GUID é fixo e corresponde ao .addin ou .xml | ✅ |
| ✔ stdole está referenciado | ✅ |
| ✔ Ícones estão no Resources do projeto | ✅ |
| ✔ Pasta correta no ApplicationPlugins (para .addin) ou Bundle com .xml (para AppPackage) | ✅ |
| ✔ Compilado em 64 bits (x64) | ✅ |
| ✔ Classe principal Implements Inventor.ApplicationAddInServer | ✅ |
| ✔ Inventor está abrindo com permissões de leitura do local da DLL | ✅ |

Se ainda assim o botão não aparecer, posso te ajudar a revisar o .addin ou .xml de registro e o projeto no Visual Studio.

Quer que eu revise seu StandardAddInServer.vb para garantir que ele está implementado corretamente?

**Você disse:**

O tipo 'Form' não está definido. class "MyButton" e class "MyButton" estão em conflito em namespace "SincronizarItem". "ReadOnly" não é um membro de "RichTextBox". "MessageBoxIcon" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "MessageBoxIcon" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "MessageBoxIcon" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "MessageBoxIcon" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "MessageBoxButtons" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "MessageBoxButtons" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "MessageBoxButtons" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "MessageBoxButtons" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "Guid" não é um parâmetro de "Function AddButtonDefinition(DisplayName As String, InternalName As String, Classification As CommandTypesEnum, [ClientId As Object], [DescriptionText As String = ""], [ToolTipText As String = ""], [StandardIcon As Object], [LargeIcon As Object], [ButtonDisplay As ButtonDisplayEnum = kDisplayTextInLearningMode]) As ButtonDefinition". "DockStyle" não está declarado. Ele pode estar inacessível devido ao seu nível de proteção. "Dock" não é um membro de "RichTextBox".

**O ChatGPT disse:**

Esses erros estão relacionados, principalmente, a **referências ausentes** e **imports faltando** no seu projeto do Visual Studio.

Vamos corrigir passo a passo:

**✅ 1. Adicione os**Imports**corretos**

No topo de cada arquivo .vb que utiliza formulários, enums como MessageBoxIcon, ou controles como Form, DockStyle, etc., adicione estes namespaces:

Imports System Imports System.Windows.Forms Imports System.Drawing

🔧 **Explicação**:

* System.Windows.Forms → necessário para Form, MessageBox, MessageBoxIcon, RichTextBox, DockStyle, etc.
* System.Drawing → necessário para ícones e imagens (Bitmap, Icon).
* System → inclui o tipo Guid.

**✅ 2. Adicione a Referência ao**System.Windows.Forms

No Visual Studio:

1. Vá para **Referências** do projeto (clique direito > **Adicionar referência**).
2. Selecione **Assemblies > Framework**.
3. Marque:
   * System.Windows.Forms
   * System.Drawing

Se estiver usando SDK-style (.NET Core ou .NET 5+), adicione isso no .vbproj:

<ItemGroup> <Reference Include="System.Windows.Forms" /> <Reference Include="System.Drawing" /> </ItemGroup>

**✅ 3. Corrija o erro:**Guid**não é um parâmetro válido**

Este erro vem desta linha incorreta:

\_settingsButton = conDefs.AddButtonDefinition( ... Guid:="algum-guid" )

⚠️ O parâmetro não se chama Guid. Ele se chama **ClientId**.

✅ Corrija assim:

\_settingsButton = conDefs.AddButtonDefinition( "Sincronizar", "SincronizarItem.MyButton", CommandTypesEnum.kEditMaskCmdType, "49b5d624-3533-42ed-ad63-5cf19250e8ad", *' <--- ClientId* "Descrição do botão", "Tooltip do botão" )

⚠️ **Não use**Guid.NewGuid()**aqui** — use sempre um GUID **fixo** que esteja no seu .addin ou .xml.

**✅ 4. Corrija**ReadOnly**e**Dock**em**RichTextBox

Você deve configurar Dock e ReadOnly corretamente assim:

Dim richText As New RichTextBox() richText.Dock = DockStyle.Fill richText.ReadOnly = True

**✅ 5. Resolva o conflito de classe**MyButton**duplicada**

❌ **Erro**: Class "MyButton" e class "MyButton" estão em conflito no namespace "SincronizarItem".

Significa que você **tem duas classes com o mesmo nome no mesmo namespace**.

**🛠️ Soluções possíveis:**

* Renomeie uma das classes (MyButton → SyncButton, por exemplo).
* Ou separe em **namespaces diferentes**:

Namespace SincronizarItem.UI Public Class MyButton *' ...* End Class End Namespace

**🧪 Checklist final de erros:**

| **Erro** | **Solução** |
| --- | --- |
| Form não definido | Imports System.Windows.Forms e referenciar System.Windows.Forms |
| MessageBoxIcon/MessageBoxButtons não definidos | Mesmo acima |
| Dock e ReadOnly não são membros | Criar RichTextBox corretamente e usar DockStyle |
| Guid não é parâmetro de AddButtonDefinition | Corrigir para ClientId |
| Conflito de classe MyButton | Renomear ou separar por namespace |

Se quiser, posso revisar seu .vbproj para garantir que todas as referências estão certas também.

Você gostaria de que eu te mostre um modelo de projeto Visual Basic (para Inventor Add-in) com tudo isso configurado?

**Você disse:**

ERROR|RunExternalRule: Cannot find an external rule file named: "Qtde personalizada.Ilogic" (referenced in the document ESTRUTURA METALICA .iam).

**O ChatGPT disse:**

Esse erro:

ERROR | RunExternalRule: Cannot find an external rule file named: "Qtde personalizada.Ilogic"

significa que o **Inventor não está conseguindo localizar a regra iLogic externa** chamada "Qtde personalizada.iLogic" que está referenciada no documento ESTRUTURA METALICA.iam.

**🧠 O que são "Regras Externas" no iLogic?**

Regras externas são arquivos .iLogic (ou .vb) **armazenados fora do documento do Inventor**, geralmente em uma pasta configurada globalmente ou por projeto.

**✅ Soluções passo a passo:**

**1. 🔎 Verifique se o arquivo existe**

Confirme se o arquivo "Qtde personalizada.iLogic" **realmente existe** em alguma destas pastas:

* Pasta **de regras externas** configurada no Inventor
* Pasta do **documento atual**
* Pasta **do projeto do Inventor (.ipj)**

✅ **Se o arquivo não existir**, ele precisa ser criado ou recuperado do backup.  
❌ **Se o nome estiver errado**, corrija no iLogic onde ele é chamado.

**2. ⚙️ Verifique a pasta de regras externas no Inventor**

No Inventor:

1. Vá para **Tools > iLogic > iLogic Configuration**
2. Na aba **External Rules**, veja onde está apontando o caminho das regras externas:
   * Confirme se a pasta onde está "Qtde personalizada.iLogic" está listada
   * Se não estiver, adicione com o botão **"Add"**

**3. 🔧 Verifique onde a regra está sendo chamada**

No seu documento (ESTRUTURA METALICA.iam), provavelmente tem uma chamada como esta dentro de uma regra:

iLogicVb.RunExternalRule("Qtde personalizada.iLogic")

Isso faz com que o Inventor busque o arquivo "Qtde personalizada.iLogic" em **alguma das pastas configuradas no passo anterior**.

**✅ Alternativa recomendada:**

Use caminho relativo ou absoluto, por exemplo:

iLogicVb.RunExternalRule("C:\RegrasInventor\Qtde personalizada.iLogic")

Ou mova a regra para a pasta de regras externas configurada e use:

iLogicVb.RunExternalRule("Qtde personalizada.iLogic")

**4. 📌 Verifique o nome do arquivo**

Certifique-se de que o nome do arquivo está **exatamente igual**, inclusive:

* Letras maiúsculas/minúsculas
* Espaços
* Extensão correta (.iLogic, .vb, etc.)

**✅ Resumo das ações para resolver**

| **Ação** | **Descrição** |
| --- | --- |
| 🔍 Verificar se o arquivo existe | Confirme se o arquivo "Qtde personalizada.iLogic" está no disco |
| ⚙️ Configurar a pasta no Inventor | Vá em iLogic Configuration > External Rules e adicione a pasta |
| 🛠️ Corrigir a chamada da regra | Use o caminho correto ou coloque o arquivo na pasta configurada |
| ✅ Nome exato | Verifique se o nome está 100% correto, inclusive extensão e espaços |

Se quiser, posso te ajudar a criar um exemplo básico de regra Qtde personalizada.iLogic só para teste.

Deseja isso?

**Você disse:**

tem um programa que exige que o Image is not of the correct size 16x16. (It's 1496x1496)

**O ChatGPT disse:**

Esse erro:

**"Image is not of the correct size 16x16. (It's 1496x1496)"**

significa que você está tentando usar uma imagem **muito grande (1496x1496 pixels)** como ícone em um botão ou interface que exige **exatamente 16x16 pixels**, como os botões do **Autodesk Inventor Add-In**, por exemplo.

**✅ Como resolver**

**🔧 Solução: Redimensionar a imagem para o tamanho correto**

Você precisa criar uma nova imagem com **dimensão exata de 16x16 pixels** (ou 32x32 para ícones grandes, se necessário).

**📉 Como redimensionar a imagem**

**🔁 1. Usando o Paint do Windows:**

1. Abra a imagem original no Paint
2. Clique em **"Redimensionar"**
3. Escolha **"Pixels"**, marque "Manter proporção" desativado
4. Digite **16 x 16**
5. Salve como PNG ou BMP (ex: MyImage16x16.png)

**🖼️ 2. Usando o Visual Studio Resource Editor:**

Se você estiver usando My.Resources.MyImage16x16, siga:

1. Vá para o **Solution Explorer**
2. Clique com o botão direito em My Project → **"Resources"**
3. Clique em **"Add Resource" > "Add Existing File"** e escolha a imagem correta
4. Verifique na coluna de detalhes que ela é **16x16 pixels**
   * Se não for, **redimensione antes de importar**, como mostrado acima

**🧑‍💻 3. Usando código .NET para redimensionar (se necessário):**

Se você quiser garantir via código que a imagem seja convertida para 16x16:

Private Function ResizeImage(img As Image, size As Size) As Image Dim bmp As New Bitmap(size.Width, size.Height) Using g As Graphics = Graphics.FromImage(bmp) g.DrawImage(img, 0, 0, size.Width, size.Height) End Using Return bmp End Function

Exemplo de uso ao definir ícone:

Dim resizedImage As Bitmap = ResizeImage(My.Resources.MyOriginalImage, New Size(16, 16)) \_settingsButton.StandardIcon = PictureDispConverter.ToIPictureDisp(resizedImage)

**🧠 Dica**

O Inventor exige:

* **16x16** para StandardIcon
* **32x32** para LargeIcon

Use imagens já no tamanho correto para evitar perda de qualidade ou falhas no carregamento.

Se quiser, posso gerar para você uma imagem 16x16 com base em uma maior (você pode me enviar), ou te indicar ferramentas online para redimensionar facilmente. Deseja isso?

**Você disse:**

Image is not of the correct size 16x16. (It's 1496x1496)

**O ChatGPT disse:**

Esse erro:

**Image is not of the correct size 16x16. (It's 1496x1496)**

acontece porque o Autodesk Inventor Add-In (ou outro componente do Visual Studio) **espera uma imagem de ícone exatamente 16x16 pixels**, mas você está tentando usar uma imagem de **1496x1496 pixels**, que é gigante para esse contexto.

**✅ Solução: Redimensionar a imagem para 16x16**

Você tem **duas opções principais** para resolver isso:

**🟢 1. Redimensionar a imagem manualmente**

**📌 Usando o Paint (ou qualquer editor de imagem):**

1. Clique com o botão direito na imagem original > **Abrir com > Paint**
2. Clique em **"Redimensionar"**
3. Escolha a opção **"Pixels"**
4. Digite:
   * Horizontal: **16**
   * Vertical: **16**
   * **Desmarque** “Manter taxa de proporção” se necessário
5. Salve como **PNG ou BMP**

**🟢 2. Redimensionar dinamicamente via código (VB.NET)**

Se você está carregando a imagem com My.Resources e quer redimensionar em tempo de execução:

Private Function ResizeImage(image As Image, width As Integer, height As Integer) As Bitmap Dim resized As New Bitmap(width, height) Using g As Graphics = Graphics.FromImage(resized) g.DrawImage(image, 0, 0, width, height) End Using Return resized End Function

**Exemplo de uso:**

Dim smallIcon As Bitmap = ResizeImage(My.Resources.MyImage, 16, 16) \_settingsButton.StandardIcon = PictureDispConverter.ToIPictureDisp(smallIcon)

**🛠️ Dica: Gere ícones 16x16 e 32x32**

Se estiver criando um botão no **Inventor Add-In**, você geralmente precisa de **dois tamanhos**:

* StandardIcon → **16x16**
* LargeIcon → **32x32**

**Ambos devem estar exatamente no tamanho certo**, senão o erro aparecerá novamente.

**🔗 Ferramentas online para redimensionar imagens:**

* https://www.resizepixel.com/
* https://pixlr.com/
* https://www.iloveimg.com/resize-image

Se quiser, posso gerar uma imagem .png de 16x16 ou 32x32 para você — basta me enviar a original. Deseja isso?

**Você disse:**

Image is not of the correct size 32x32. (It's à 320,0277404785156x322,4279479980469)

**O ChatGPT disse:**

Esse novo erro:

**Image is not of the correct size 32x32. (It's 320,027... x 322,42...)**

acontece porque a imagem que você está tentando usar como ícone no botão está com **dimensões incorretas (em pixels)** — ou pior: está sendo tratada com **valores flutuantes (em vez de inteiros)**, o que é inválido para ícones em Add-Ins do Inventor.

**✅ Causa:**

Você provavelmente está passando uma imagem com **dimensões "físicas" (DPI)** ou **escala não exata**, como ocorre com imagens redimensionadas de forma incorreta (ex: via .ToBitmap() de um ícone ou renderização com DPI alta).

**✅ Solução:**

Garanta que sua imagem tem **exatamente 32x32 pixels (inteiros)** com **resolução padrão (96 DPI)**.

**🛠️ Código corrigido para redimensionar corretamente**

Aqui está uma função robusta para redimensionar uma imagem **sem erros de DPI ou tamanho flutuante**:

Private Function ResizeImageStrict(inputImage As Image, width As Integer, height As Integer) As Bitmap Dim resized = New Bitmap(width, height, Imaging.PixelFormat.Format32bppArgb) resized.SetResolution(96, 96) *' Evita problemas de DPI flutuante* Using g As Graphics = Graphics.FromImage(resized) g.InterpolationMode = Drawing2D.InterpolationMode.HighQualityBicubic g.CompositingQuality = Drawing2D.CompositingQuality.HighQuality g.SmoothingMode = Drawing2D.SmoothingMode.AntiAlias g.DrawImage(inputImage, 0, 0, width, height) End Using Return resized End Function

**Como usar:**

Dim icon32x32 As Bitmap = ResizeImageStrict(My.Resources.MyImage, 32, 32) \_settingsButton.LargeIcon = PictureDispConverter.ToIPictureDisp(icon32x32)

**✅ Alternativa: Use uma imagem que já tem o tamanho certo**

Certifique-se de que o recurso que você adicionou ao projeto (My.Resources.MyImage) já está salvo com **32x32 pixels**.  
Evite usar imagens muito grandes que precisam ser redimensionadas.

**🔍 Verifique também:**

* **Formato do recurso**: Prefira .png ou .bmp
* **Sem fundo transparente corrompido** (pode causar erro de renderização)
* **Evite DPI > 96** em imagens exportadas do Illustrator, Photoshop, etc.

Se quiser, posso gerar para você duas imagens já prontas (16x16 e 32x32) a partir de uma que você me enviar.

Deseja isso? Se sim, envie a imagem original aqui.

**Você disse:**

onde encontro imagens StandardIcon → 16x16 LargeIcon → 32x32

**O ChatGPT disse:**

Você pode encontrar imagens nos tamanhos padrão exigidos para ícones de Add-Ins do **Autodesk Inventor** (16x16 e 32x32 pixels) em alguns dos seguintes lugares:

**✅ 1. Diretório do próprio Inventor**

O Inventor já inclui muitos ícones que você pode usar como base. Explore:

C:\Program Files\Autodesk\Inventor <versão>\Bin\Icons\

Ou, se esse caminho não existir, busque por:

*\*.ico*

*\**.bmp

*\*.png*

dentro do diretório de instalação do Inventor.  
Use a busca do Explorer para localizar arquivos com esses formatos.

**✅ 2. Sites gratuitos de ícones**

Esses sites fornecem ícones com **licença livre** e possibilidade de download em **resoluções específicas**:

* https://www.iconfinder.com
* https://www.flaticon.com
* https://icons8.com/icons
* https://www.freepik.com

⚠️ Após o download, **redimensione as imagens manualmente para 16x16 e 32x32 px**.

**✅ 3. Gerar com o Paint ou Photoshop (ou similar)**

Você pode:

1. Abrir qualquer imagem no Paint.
2. Redimensionar exatamente para:
   * 16x16 (para StandardIcon)
   * 32x32 (para LargeIcon)
3. Salvar como .png ou .bmp (recomendado).

**✅ 4. Criar diretamente como recurso no Visual Studio**

1. No Visual Studio, clique com botão direito em **My Project > Resources**.
2. Vá em **Adicionar Recurso > Imagem existente**.
3. Adicione a imagem desejada.
4. Clique com o botão direito na imagem e selecione **"Redimensionar"** para ajustar para 16x16 ou 32x32.

**🧩 Exemplo Visual**

Se quiser, posso gerar agora dois ícones para você (um 16x16 e outro 32x32) a partir de uma imagem sua — basta enviar a imagem aqui.

**✅ Resumo**

| **Tamanho** | **Uso no Inventor** | **Observações** |
| --- | --- | --- |
| 16x16 px | StandardIcon | Ícone pequeno (Ribbon, etc.) |
| 32x32 px | LargeIcon | Ícone grande (botões, painéis) |

Lembre-se: **o Inventor exige tamanho exato** — não aceita imagens com DPI flutuante, valores decimais ou imagens maiores que essas dimensões.

Se quiser ajuda com imagens específicas ou quer que eu converta uma imagem sua para esses tamanhos, é só me enviar.

Obtenha respostas mais inteligentes, carregamento de imagens e de arquivos e muito mais.

Entrar

Cadastre-se gratuitamente

Parte superior do formulário

Anexar

**Buscar**

**Estudar**

**Voz**

Parte inferior do formulário

O ChatGPT pode cometer erros. Por isso, lembre-se de conferir informações relevantes.