

Dominando o Suporte a Rascunhos (Drafts) em Fiori com CAP Node.js

Um Guia do Desenvolvedor para o Ciclo de Vida do
Rascunho



Este guia destila a documentação oficial em um formato visual e açãoável, capacitando você a implementar e gerenciar a funcionalidade de rascunho com confiança em seus projetos.

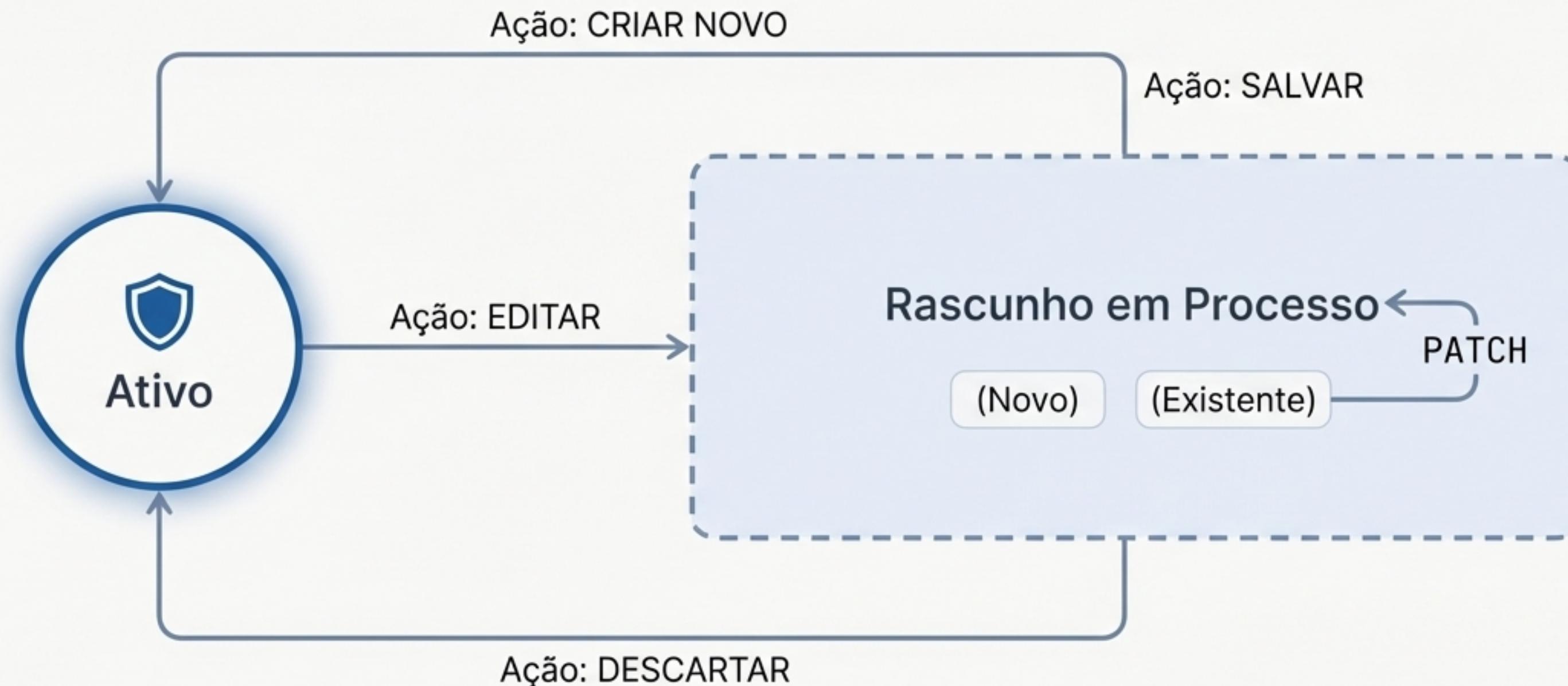
O Conceito Central: Entidade Ativa vs. Entidade de Rascunho



Como é no CSN

```
// Em seus handlers, o CAP resolve o 'target'  
// para a entidade correta (ativa ou rascunho).  
srv.on('READ', MyEntity.drafts, (req, next) => {  
  assert.equal(req.target.name, MyEntity.drafts)  
  return next()  
})
```

O Ciclo de Vida do Rascunho em um Relance



O ciclo de vida do rascunho é uma série de eventos bem definidos, acionados pelas interações do usuário na UI Fiori. Vamos explorar cada um deles.



Início do Ciclo: Criando um Novo Rascunho (Evento `NEW`)

```
// Handler registrado em MyEntity.drafts
srv.before('NEW', MyEntity.drafts, req => {
    // Perfeito para inicializar dados
    // ou gerar chaves primárias.
    req.data.ID = uuid()
})

srv.on('NEW', MyEntity.drafts, /*...*/)
srv.after('NEW', MyEntity.drafts, /*...*/)
```

O que acontece?

O usuário clica em 'Criar' na UI.

Resultado

Uma nova entrada é criada na tabela `MyEntity.drafts`. A entidade ativa ainda não existe.

Ponto Chave

Os handlers `before`, `on` e `after` são registrados diretamente na entidade de rascunho (`MyEntity.drafts`). Use o `before` para modificar os dados iniciais do rascunho.

Editando uma Entidade Ativa (Evento `EDIT`)



```
// ATENÇÃO: Handler registrado na entidade ATIVA!
srv.before('EDIT', MyEntity, req => {
    // Lógica a ser executada ANTES da
    // criação do rascunho.
})

srv.on('EDIT', MyEntity, /*...*/)
srv.after('EDIT', MyEntity, /*...*/)
```

O que acontece?

O usuário clica em 'Editar' em um registro existente.

Resultado

Uma cópia do registro ativo é criada em `MyEntity.drafts`.

Lógica Invertida

Por razões lógicas, os handlers para o evento `EDIT` são registrados na entidade ativa (`MyEntity`), não na de rascunho. O CAP gerencia a criação do rascunho nos bastidores.

Modificando o Rascunho (Evento `PATCH`)



```
// Handler registrado em MyEntity.drafts
srv.before('PATCH', MyEntity.drafts, req => {
  // Validar ou modificar dados
  // a cada alteração de campo.
})

// PATCH é um alias para o evento UPDATE
// srv.before('UPDATE', MyEntity.drafts, ...)
```

O que acontece?

O usuário **modifica** o valor de um campo no formulário de edição.

Funcionamento

Este evento é, na verdade, um **alias para o evento `UPDATE` padrão do CRUD**, mas disparado especificamente na entidade de rascunho.

Uso

Ideal para validações em tempo real ou lógica de campo dependente enquanto o usuário edita.

Persistindo as Mudanças (Evento `SAVE`)



```
// Handler registrado em MyEntity.drafts
srv.before('SAVE', MyEntity.drafts, /*...*/)

srv.on('SAVE', MyEntity.drafts, req => {
    // Lógica customizada antes da ativação
    // do rascunho.
})
```

O que acontece?

O usuário clica em 'Salvar'. O rascunho é 'ativado'.

Resultado (Depende da Origem)

Se originado de `NEW` → resulta em um `CREATE` na entidade ativa (`MyEntity`).

Se originado de `EDIT` → resulta em um `UPDATE` na entidade ativa (`MyEntity`).

Importante

O evento `SAVE` em uma entidade de rascunho é um evento **distinto** do `SAVE` em uma entidade ativa. Em entidades ativas, `SAVE` é apenas um atalho para `CREATE` + `UPDATE`. Aqui, ele representa o ato de **ativar o rascunho**.

Abandonando o Rascunho (Evento `DISCARD`)



```
// Handler registrado em MyEntity.drafts
srv.on('DISCARD', MyEntity.drafts, req => {
  // Lógica de limpeza, se necessário,
  // antes da exclusão do rascunho.
})
// 'CANCEL' também funciona como sinônimo
// srv.on('CANCEL', MyEntity.drafts, ...)
```

O que acontece?

O usuário clica em 'Cancelar' ou descarta as alterações.

Resultado

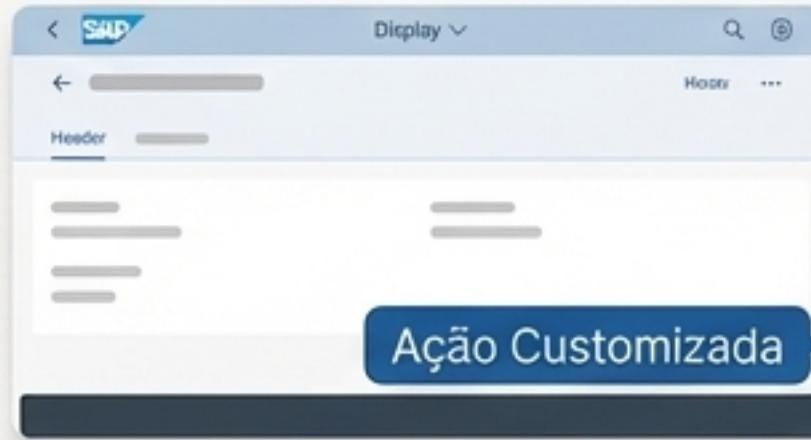
A entrada em `MyEntity.drafts` é **excluída**.
A entidade ativa permanece **inalterada**.

Sinônimo

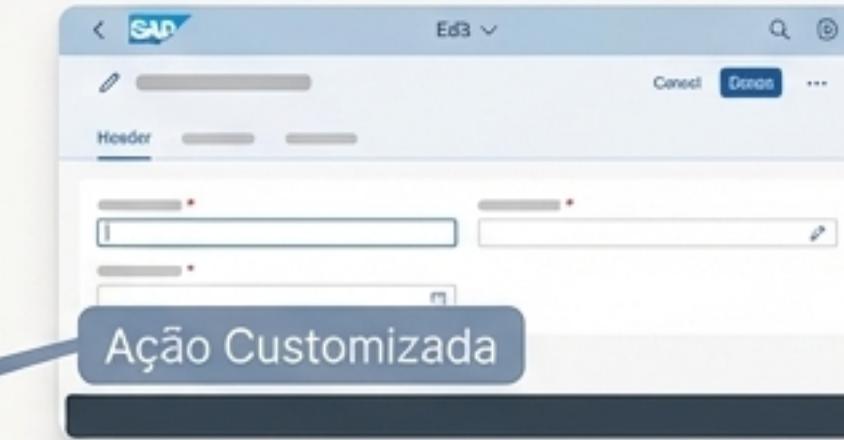
O evento `CANCEL` pode ser usado no lugar de `DISCARD` e funciona da mesma forma.

Indo Além do Padrão: Ações Customizadas

Entidade Ativa



Entidade de Rascunho



Ações e funções customizadas definidas na sua entidade ativa são herdadas pela entidade de rascunho. Isso permite implementar lógicas diferentes dependendo do contexto.

Lógica Separada

```
// Lógica para a entidade ativa  
srv.on('someAction', MyEntity, /*...*/)  
  
// Lógica diferente para a entidade de rascunho  
srv.on('someAction', MyEntity.drafts, /*...*/)
```

Lógica Compartilhada

```
// Mesma lógica para ambas  
srv.on('someAction', [  
    MyEntity,  
    MyEntity.drafts  
, /*...*/])
```



Regras de Convivência: Gerenciando Concorrência com `Draft Locks`

Conceito

Problema

O que acontece se dois usuários tentarem editar o mesmo registro ao mesmo tempo?

Solução CAP

Para prevenir inconsistências, a entidade ativa é **automaticamente bloqueada** para modificações por outros usuários assim que um rascunho é criado (EDIT).

Liberação

O bloqueio é liberado quando o rascunho é salvo (SAVE), descartado (DISCARD) ou quando o tempo limite é atingido.

Configuração

Timeout Padrão

15 minutos.

Como Configurar

Adicione a propriedade ao seu package.json ou similar.

```
cds.fiori.draft_lock_timeout=30min  
// valores aceitos: '1h', '10min', 1000 (ms)
```

Após o `draft_lock_timeout` expirar, qualquer usuário pode excluir o rascunho de outro usuário para criar o seu próprio. Não podem existir dois rascunhos para a mesma entidade simultaneamente.



Regras de Convivência: Limpeza Automática com `Draft Timeouts`

Conceito

Problema

Rascunhos inativos e abandonados podem se acumular no banco de dados.

Solução CAP

Rascunhos inativos são **excluídos automaticamente** após um período de tempo.

Timeout Padrão

30 dias.

Configuração

Como Configurar

```
{  
  "cds": {  
    "fiori": {  
      "draft_deletion_timeout": "28d"  
    }  
  }  
}
```

Opções de Valor

- `false`: Desativa a exclusão automática.
- `30d`: Número de dias.
- `72h`: Número de horas.
- `1000`: Número de milissegundos.

Background Técnico

A exclusão não é um job periódico (cron). Ela é implementada como um efeito colateral da criação de novos rascunhos, otimizando recursos.

Manobra Avançada: Contornando o Rascunho (`Bypassing Drafts`)

Caso de Uso

Por que fazer isso?

Útil para cenários onde não há UI, como serviços técnicos ou integrações de sistemas que precisam modificar dados diretamente sem criar um rascunho.

Como Habilitar

Ateve a feature flag na sua configuração.

```
{  
  "cds": {  
    "fiori": {  
      "bypass_draft": true  
    }  
  }  
}
```

Exemplos HTTP

Criando uma instância ativa diretamente

```
POST /Books  
{  
  "ID": 123,  
  "IsActiveEntity": true  
}
```

Modificando uma instância ativa diretamente

```
PATCH /Books(ID=123,IsActiveEntity=true)  
{  
  "title": "How to be more active"  
}
```

Aviso

Esta feature cria pontos de entrada adicionais em sua aplicação. Handlers customizados serão acionados com payloads de delta, e não com o objeto de negócios completo.

Manobra Avançada: Controle Total com APIs Programáticas

Você pode invocar as ações de rascunho diretamente em seus handlers de serviço para orquestrar lógicas de negócio complexas.



```
// Criar um novo rascunho  
await srv.new(MyEntity.drafts, data)
```



```
// Criar um rascunho a partir de uma instância ativa  
await srv.edit(MyEntity, keys)  
// ou a forma alternativa:  
// await srv.new(MyEntity.drafts).for(keys)
```



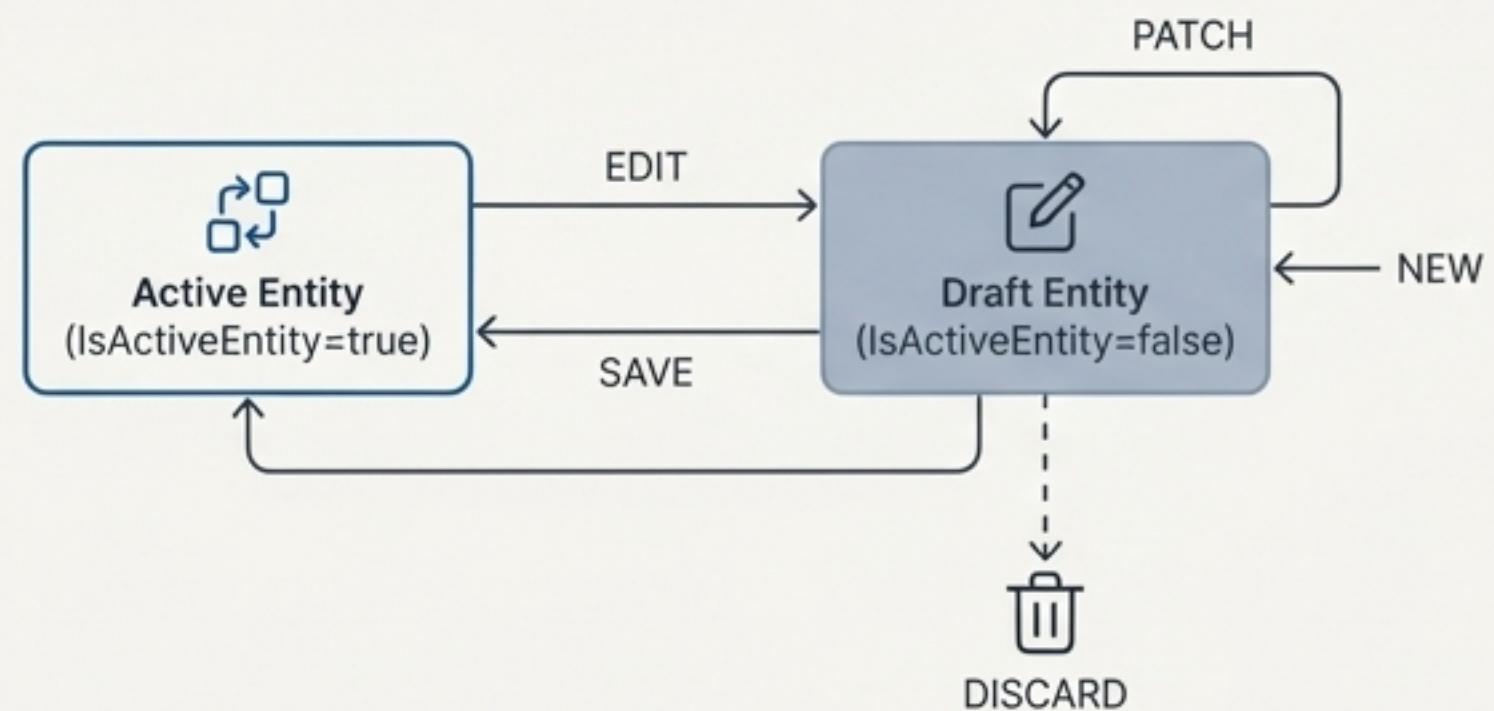
```
// Ativar (salvar) um rascunho  
await srv.save(MyEntity.drafts, keys)
```



```
// Descartar um rascunho  
await srv.discard(MyEntity.drafts, keys)
```

Estas APIs são essenciais para construir fluxos de trabalho customizados ou acionar ações de rascunho a partir de eventos que não vêm da UI.

Resumo e Pontos-Chave



- ☞ **Dualidade é Fundamental:** Entenda o modelo de Entidade Ativa vs. Entidade de Rascunho. É a base de tudo.
- ☞ **Domine os Eventos:** O ciclo de vida é governado pelos eventos `NEW`, `EDIT`, `PATCH`, `SAVE` e `DISCARD`. Saiba onde registrar seus handlers.
- ☞ **Configure o Ambiente:** Não esqueça de ajustar os timeouts de `lock` e `deletion` para garantir uma aplicação robusta e segura.
- ☞ **Use as Ferramentas Avançadas:** Lembre-se das APIs programáticas e da opção de `bypass` para cenários que vão além da UI padrão.

Recursos Essenciais e Próximos Passos

Este guia fornece a base para você trabalhar com rascunhos de forma eficaz. Para detalhes mais aprofundados e tópicos relacionados, consulte a documentação oficial.

SAP Fiori Draft Support in CAP

cap.cloud.sap/docs/fiori/drafts

A documentação oficial completa, incluindo exemplos e casos de uso adicionais.



Construa com confiança.