# API - Antes de avançar

A Oystr oferece duas formas principais de execução via API (além das integrações diretas via webhook ou APIs dos próprios Apps). O primeiro modelo é o **1:1**, indicado para execuções individuais, onde cada requisição dispara um robô de forma isolada e o resultado é consultado posteriormente de maneira assíncrona. O segundo é o modelo de **lote**, voltado para processamento em escala, no qual múltiplos itens são organizados em uma fila, processados dentro de uma execução e posteriormente disponibilizados por meio de relatórios e resultados consolidados.

No momento, indicamos a integração via modelo **1:1** para a maioria dos casos. A integração por **lote** deve ser avaliada e aprovada previamente junto à equipe da Oystr.


- Boas Práticas
- Autenticação na API
- API de Itens -1:1
- API de Lotes - 1:N
- Robôs
- Intimações
- Autenticação 2 Fatores
- Distribuição

# Boas Práticas

Para garantir uma integração estável e eficiente com a API da Oystr, tanto em Lotes ou Itens, recomendamos observar as seguintes boas práticas:

- **Frequência de envio de requisições**
  Evite o envio excessivo de chamadas em curtos períodos de tempo. Altas taxas de requisição podem gerar sobrecarga, aumento de filas internas e impactar o tempo de processamento dos robôs. Isso também poderá acarretar em problemas de bloqueios externos e autenticação/identidade com os sistemas envolvidos.
- **Frequência de consulta de resultados (polling)**
  As consultas de resultados devem respeitar os intervalos recomendados. Polling muito frequente pode causar consumo desnecessário de recursos e degradação de performance. Para consulta de resultados recomendamos intervalos entre 30 e 120 segundos.

- **Consulta de dados antigos**
  Resultados de execuções ficam disponíveis para consulta por um período limitado. Atualmente, a Oystr mantém os dados disponíveis por até 7 dias. Após esse período, os dados podem não estar mais acessíveis.

- **Validação de credenciais antes da execução**
  Evite executar robôs com credenciais potencialmente inválidas ou expiradas. Esse tipo de execução consome recursos desnecessários (como captcha, proxy e processamento) e pode impactar negativamente o desempenho geral da operação.

- **Evitar execuções duplicadas em curto intervalo**
  Não execute o mesmo robô com os mesmos parâmetros repetidamente em um curto espaço de tempo. Isso pode gerar consumo desnecessário de recursos e aumentar o risco de bloqueios externos (ex: sistemas de terceiros). Sempre que possível, utilize mecanismos de controle como `cid` para evitar duplicidade.

- **Registro de requisições e respostas (logging)**
  Recomendamos armazenar as requisições enviadas e as respostas recebidas (incluindo IDs de execução) em logs ou arquivos persistentes. Isso é fundamental para rastreabilidade, auditoria e investigação de possíveis problemas em ambiente de produção.

# Autenticação na API

## Autenticação na API

Todas as requisições http feitas aos servidores da Oystr devem conter, no header da requisição, o parâmetro ***X-Oystr-Auth***, com o seu ***token*** de autenticação.

Todas as requisições para nossa API demandam o uso do **header *X-Oystr-Auth***, ele é obrigatório. Caso o token não seja informado no header da requisição ou caso o token seja inválido, o retorno do servidor, nas demais requisições, será **HTTP/1.1 403 Forbidden**, indicando a ausência de permissões para efetuar a requisição.

O método /ping é utilizado apenas para realização de autenticação na API. Para saber como obter a o seu token, acesse o link abaixo e siga as instruções ali definidas.

## Como obter a chave de API

Antes de mais nada, a primeira coisa que você precisa fazer é obter a sua **chave de API (API Key)**. Sem ela, você não conseguirá se autenticar na API e, consequentemente, não conseguirá consumir seus métodos.

Abaixo, segue um passo a passo de como obter a sua chave de API:

- Acesse o Console/Painel da Oystr (https://console4.oystr.com.br);

Sua chave será criada e exibida no console. Para consumir a API, basta copiar a chave criada e passar no ***HEADER***, conforme indicado no método de ***Autenticação na API***.

# API de Itens -1:1

## Conceitos

Todos os robôs da Oystr que são utilizados através da API 1:1 possuem um nome e versão. Diferente do modelo em lote, onde existe o conceito de filas e execuções com múltiplos itens, na API 1:1 cada requisição representa a execução de um único item de forma isolada. **Não há o conceito de fila ou execução agrupada neste modelo**.

Cada chamada realizada na API 1:1 dispara uma execução individual de robô, responsável por processar os dados enviados e retornar um resultado específico para aquele item. Essas execuções são independentes entre si e não compartilham estado ou fila.

De maneira geral, para executar um robô utilizando a API 1:1, temos o seguinte fluxo:

| Ordem | Tipo | Descrição |
|---|---|---|
| *0.1* | *Opcional* | *Validar as credenciais que serão utilizadas pelos robôs para acessar um sistema (tribunal, gestão jurídico, etc)* |
| 1 | **Obrigatório** | Criar execução do item e receber o identificador da execução (id assíncrono) |
| 2 | **Obrigatório** | Consultar o retorno/status da execução |
| 3 | **Obrigatório** | Obter o resultado final da execução |

## Respostas Assíncronas da API

Antes de continuar, é importante entender o conceito de que muitas das chamadas na **API da Oystr** retornam respostas de maneira **assíncrona**. Isso quer dizer que a resposta da chamada **não estará pronta de imediato** e você receberá um id para consultar o resultado da chamada posteriormente.

O design da API leva em consideração **o tempo que os robôs demoram para executar uma determinada tarefa**. Lembre-se, algumas tarefas podem demorar horas para finalizar. Isso significa que seu código de integração precisa estar preparado para lidar com essas respostas assíncronas. A seguir, veja um exemplo de chamada que recebe um id de consulta assíncrona:

```
http POST https://api4.oystr.com.br/v1/[método de retorno assíncrono] X-Oystr-Auth:[token]

HTTP/1.1 200 OK
Content-Type: application/json

{
   "id": "[id]"
}
```

O retorno de um método assíncrono é um **json** com um único campo: **id**. Precisamos fazer uma segunda chamada na API, usando o **id retornado na chamada anterior** para consultar se o resultado já está pronto. Considere uma chamada para validação de credencial no robô de testes “sample”, usando o usuário “zzz” e a senha “zzz”. Segue exemplo:

```
http POST https://api4.oystr.com.br/v1/credentials/validate/sample/v1.0 X-Oystr-Auth:[token] username=zzz
password=zzz

HTTP/1.1 200 OK
Content-Type: application/json

{
   "id": "zqwG3hcEGxYl457T8A1d6qS77PCky71HgWdBDj4Z1sTmV6mqlp5VCcCs1ZutIEue"
}
```

Após esta chamada, agora temos em mãos o **id de consulta do resultado da nossa chamada assíncrona**. O recomendado é que se consulte no **mínimo a cada 10 segundos e no máximo a cada 120 segundos**.

A partir de agora, precisamos consultar continuamente a API, até que tenhamos a confirmação de conclusão da execução.

Neste momento, podemos ter 3 tipos de código de resposta: 200, 404 ou 500. Abaixo, segue descrição de cada tipo de resposta.

| Código | Tipo | Descrição |
|---|---|---|
| 200 | OK - Resultado pronto | Com o payload da resposta, você receberá os dados relacionados ao seu resultado. |
| 404 \| 204 | Resultado não disponível ainda | Neste caso indica-se que os dados ainda não estão prontos. |
| 500 | Erro interno | Neste caso entrar em contato conosco e informar o ID da chamada. |

Segue um exemplo de consulta, com resposta:

```
http GET https://api4.oystr.com.br/v1/cached/[id] X-Oystr-Auth:[token]
HTTP/1.1 404 NOT FOUND // indica que o resultado não está pronto
// 3 segundos depois
http GET https://api4.oystr.com.br/v1/cached/[id] X-Oystr-Auth:[token]
HTTP/1.1 200 OK
Content-Type: application/json
{
   "Message"     : null,
   "notAllowed"  : null,
   "notAvailable": null,
   "Valid"       : true
}
```

# 0.1 - Validar Credenciais

```
POST: https://api4.oystr.com.br/credentials/validate/:botId/v3.0.0-dev
```

## Usuário/Senha - UP

objeto ***"credentials"***, com par **usuário e senha**, será utilizado pelo robô para conseguir acesso ao sistema no qual ele irá executar as tarefas. Por exemplo, em um robô de tribunal onde a tarefa é obter a última movimentação processual, esse usuário/senha será utilizado para entrar no sistema do tribunal.

Quando informado na requisição (via arquivo ***payload*** ou especificado no ***body*** da requisição), é necessário seguir o formato abaixo:

```
{
  "username": "...",
  "password": "...",
  "credentialsOption": "..."
}
```

## Parâmetros

| Parâmetro | Tipo | Descrição |
|---|---|---|
| **username** | *String (obrigatório)* | Usuário utilizado para acessar o sistema em que o robô irá executar a tarefa. |
| **password** | *String (obrigatório)* | Senha correspondente ao usuário informado. |

| Parâmetro | Tipo | Descrição |
|---|---|---|
| **credentialsOption** | *String (opcional)* | Valor opcional que será informado indicando particularidades das opções informadas. Quando necessário, a documentação do robô irá apontar. |

## Certificado Digital A1 - CP

O objeto *“credentials”* com certificado A1 será utilizado pelo robô para conseguir acesso ao sistema no qual ele irá executar as tarefas. Por exemplo, em um robô unificado intimações onde a tarefa é obter a todas as intimações disponíveis para o advogado em diversos tribunais do Brasil, esse certificado A1 será utilizado para entrar no sistema de cada tribunal especificado.

Quando informado na requisição (via arquivo ***payload*** ou especificado no ***body*** da requisição), é necessário seguir o formato abaixo:

```
{
  "username": "...",
  "base64Cert": "...",
  "pin": "...",
  "credentialsOption": "..."
}
```

## Parâmetros

| Parâmetro | Tipo | Descrição |
|---|---|---|
| **username** | *String (obrigatório)* | Usuário utilizado para acessar o sistema em que o robô irá executar a tarefa. |
| **base64Cert** | *String (obrigatório)* | Conteúdo do certificado A1 codificado em Base64 |
| **pin** | *String (obrigatório)* | PIN ou senha utilizada para utilizar o certificado. |
| **pin** | *String (opcional)* | Valor opcional que será informado indicando particularidades das opções informadas. Quando necessário, a documentação do robô irá apontar. |

# 1 - Criar Execução do Item 1:1

```
POST: https://api4.oystr.com.br/v1/execute/single
```

### Código de retorno

| Retorno | Descrição |
|---|---|
| 202 | inicia a execução de um ítem. Retona o id para consulta futur |

| Retorno | Descrição |
| --- | --- |
| 5xx ou 4xx | Falha ao iniciar a execução do ítem. Nestes casos é necessário utilizar (manualmente via suporte) o header X-Oystr-TraceId retornado para verificar se o ítem foi recebido pela API e está em execução ou não. |

### Formato da Requisição (via HTTP)

```
{
   "dry": false,
   "bot": "",
   "version": "",
   "cid": "",
   "timeout": "",
   "deadline": "",
   "credentials": {
      "username": "",
      "password": ""
   },
   "data": {},
   "files": [{
      "bound": true,
      "property": "",
      "description": "",
      "name": "",
      "data": ""
   }]
}
```

| Campo | Obrigatório | Formato | Default | Descrição |
| --- | --- | --- | --- | --- |
| dry | Não | Boolean | false | Indica se é uma execução de teste |
| bot | Sim | String | | Id do robô |
| version | Sim | String | | Versão do robô |
| cid | Não | String | vazio | Identificador único da requisição[1] |
| force | Não | Boolean | vazio | Indica se um item único, discriminado pelo cid deve ser reexecutado ou não |
| timeout | Não | Duration | 5 minutos | Tempo máximo aguardando o retorno de um item desde o início da execução[2]. Exemplo: 30s |

| Campo | Obrigatório | Formato | Default | Descrição |
|---|---|---|---|---|
| deadline | Não | ZonedDateTime | vazio | Data máxima para o início da execução do item[3] no formato ISO 8601. Exemplo 2022-01-01T00:00:00.0-03:00 |
| credentials | Robô | json | | Credenciais usadas na requisição |
| data | Sim | json | | Dados passados diretamente para o robô. Cada robô possuí um formato específico. |
| file | Robô | json | | Arquivos passados para o robô |

1. ID usado para evitar requisições duplicadas. Caso o cid não seja único a resposta será do tipo Duplicate . Caso o cid seja inválido a requisição será rejeitada
2. . Após este período a resposta será do tipo Timeout
3. Caso a execução do ítem não tenha iniciado até esta data a resposta será do tipo Overdue.

# 2 - Consultar o retorno da execução - (assíncrono)

```
GET: https://api.oystr.com.br/v1/execute/single/:bot/:cached?format=latest
```

**Código de retorno**

| Retorno | Descrição |
|---|---|
| 404 | id inválido ou ítem rejeitado. Nunca haverá uma resposta para este item. |
| 202 | Reposta indisponível, aguarde |
| 200 | Resposta disponível |

# 3 - Interpretar e obter o resultado final da execução

Após uma resposta 200, teremos como payload da resposta o resultado final da execução. Os resultados ficam disponíveis no modelo do endereço abaixo de acordo com o header **X-Oystr-ReturnPath**:

```
GET: https://api.oystr.com.br/v1/execute/single/:bot/:cached?format=latest
```

O payload recebido depende do valor da variável forma:

**Formato Legado** - format=legacy

```
GET: https://api.oystr.com.br/v1/execute/single/:bot/:cached?format=latest
```

```
{
   "type": "",
   "payload": {}
}
```

**Formato Recomendado** - format=latest

O payload em json obtido quando a resposta está disponível segue o seguinte modelo:

```
{
   "account": 0,
   "user": 0,
   "bot": "",
   "version": "",
   "cid": "",
   "request": "",
   "started": "",
   "ended": "",
   "taskTime": "",
   "timeout": "",
   "finishedAs": "",
   "retry": "",
   "dry": false,
   "result": {}
}
```

Descrição dos campos:

| Campo | Formato | Descrição |
|---|---|---|
| account | Long | Conta que realizou a requisição |
| user | Long | Usuário que realizou a requisição |
| bot | String | Robô |
| version | String | Versão do robô |
| cid | String | id de integração |
| request | String | Id da requisição |

| Campo | Formato | Descrição |
|---|---|---|
| started | ZonedDateTime | Data de início da execução do item no formato ISO-8601 |
| ended | ZonedDateTime | Data de início da execução do item no formato ISO-8601 |
| taskTime | Duration | Tempo de execução do item |
| timeout | Duration | Timeout efetivamente usado |
| retry | Enum | indica se é possível re-executar o item |
| dry | Json | Indica se é uma execução DE TESTE |
| finishedAs | Enum | Tipo de resultado (tabela abaixo) |
| result | Json | Payload com o resultado |

# Tipos de retorno

Os tipos de resultados possíveis:

| Resposta | Reexecução Recomendada? | Origem | Descrição |
|---|---|---|---|
| Response | Não | Bot | Resposta válida |
| PartialResponse | Não | Bot | |
| NotAllowed | Sim | Bot | Não é permitida a execução do item |
| NotFound | Não | Bot | Item não encontrado |
| NotConsistent | Não | Bot | Dados inconsistentes |
| NotAvailable | Sim | Bot | Item não disponivel |
| ProxyError | Sim | Bot | Erro de proxy |
| CaptchaError | Sim | Bot | Erro de captcha |
| Forbidden | Sim | Bot | Execução do item não é permitida |
| BotError | Sim | Bot | Erro no robô |

| Resposta | Reexecução Recomendada? | Origem | Descrição |
|---|---|---|---|
| Other | Não | Bot | Outros |
| RequestHasViolations | Não | API | Pedido contém informações inválidas. O item não foi executado. |
| NotAndApiRequest | Não | API | Formato inválido de requisição. O item não foi executado |
| NotAccepted | Não | API | Execução do item nõa foi aceita. O item pode ou não ter sido executado |
| Timeout | Não | API | Timeout durante a execução do item. O item pode ou não ter sido executado |
| Overdue | Não | API | Execução do item ocorreria apís o deadline. Item não foi executado |
| Duplicate | Não | API | Item duplicado baseado no cid. Item não foi executado |
| UnexpectedError | Não | API | Erro inesperado. O item pode ou não ter sido executado |
| Unknown | Não | API | Desconhecido. O item pode ou nõa ter sido executado |

# Requisições Duplicadas

A Oystr não testa os dados enviados para a execução com a exceção dos atributos cid e force . Caso o cid esteja presente na requisição, a Oystr fará a verificação se este cid foi executado nos últimos 15 dias para esta conta e robô. Caso o cid seja "duplicado", o ítem será executado apenas se o parâmetro force estiver presente com o valor true , caso contrário a resposta do ítem será do tipo Duplicate .

A Oystr emprega uma estratégia estatística para verifica ser o ítem já foi executado. Esta estratégia permite falsos positivos, ou seja:

1. ítem não foi executado (com certeza) para esta conta e robô nos últimos 15 dias, ou2
2. . ítem possivelmente já foi executado antes (false positivo)

Um cid , se presente, deve ter entre 1 e 50 carateres alphanuméricos ou '-' de acordo com a seguinte expressão regular: [0-9a-zA-Z\\-]{1,50} . Caso o cid seja inválido o ítem não será executado, a requisição será ignorada e ficará sem resposta, retornando 404 em consultas futuras.

# Reexecução

Em alguns casos é possível que um ítem seja reexecutado em caso de erro, mesmo que um cid tenha sido utilizado. Os critérios que definem se um ítem pode ou não ser reexecutado ficam a cargo do cliente da API que deve observar os seguintes critérios:

1. Se houve uma execução anterior "do mesmo" íten, esta deve ter retornado como insucesso, com o parâmetro retry = SAFE . Caso contrário não é seguro reexecutar o ítem.
2. . Caso um cid seja utilizado, ele deve ser único ou o parâmetro force = true deve ser utilziado para ignorar o teste de duplicidade.

A Oystr não verifica se a requisição anterior retornou com retry = SAFE durante a nova execução, mesmo que o cid seja o mesmo. Esta verificação fica a cargo do cliente/usuário da api.

# API de Lotes - 1:N

## Conceitos

Todos os robôs da Oystr que são utilizados através da API de lotes tem um nome e versão. Os robôs da Oystr trabalham e operam em execuções. Uma execução contém itens que fazem parte de uma fila. Cada item da fila é consumida por uma execução de robô. As execuções podem ter vários robôs trabalhando ao mesmo tempo. Após a execução é possível você ter acesso aos dados de relatório e saída do robô.

De maneira geral, para se criar uma execução e rodar um robô, temos que realizar as seguintes tarefas na ordem descrita:

| Ordem | Tipo | Descrição |
|---|---|---|
| *0.1* | **Opcional** | *Validar as credenciais que serão utilizadas pelos robôs para acessar um sistema (tribunal, gestão jurídico, etc)* |
| *0.2* | **Opcional** | *Executar uma proto fila para obter itens* |
| 1 | **Obrigatório** | Criar uma fila de itens |
| 2 | **Obrigatório** | Anexar arquivos a fila de itens (apenas para os robôs que fazem upload de arquivos) |
| 3 | **Obrigatório** | Iniciar a execução |
| 4 | **Obrigatório** | Consultar o status da execução |
| 5 | **Obrigatório** | Consultar o informações da execução |
| 6 | **Obrigatório** | Validar as respostas/dados/journal da execução |
| 7 | **Obrigatório** | Obter o relatório/resultado da execução |

**Para utilizar nossa API de lotes, entre em contato conosco para solicitar a autorização e documentação.**

# Robôs

# API - 1:1

## 1 - TRIBUNAIS

### 1.1 - PROTOCOLO

| ID ROBÔ | SISTEMA |
|---|---|
| pje-tj-protocolo | PJE |
| pje-tj-protocolo-habilitacao | PJE |
| pje-trt-protocolo | PJE |
| pje-trt-protocolo-habilitacao | PJE |
| eproc-protocolo | EPROC |
| esaj-protocolo | ESAJ |
| tj-rj-protocolo | PROPRIO |
| projudi-protocolo | PROJUDI |
| tj-se-protocolo | PROPRIO |

### 1.2 - AJUIZAMENTO

| ID ROBÔ | SISTEMA |
|---|---|
| esaj-ajuizamento | ESAJ |
| pje-tj-ajuizamento | PJE |

| | |
|---|---|
| projudi-pr-ajuizamento | PROJUDI |
| eproc-ajuizamento | EPROC |
| projudi-go-ajuizamento | PROJUDI |
| tj-se-ajuizamento | PROPRIO |

## 1.3 - CÓPIA INTEGRAL

| ID ROBÔ | SISTEMA |
|---|---|
| eproc-copia-integral | EPROC |
| esaj-copia-integral | ESAJ |
| pje-trt-copia-integral | PJE |
| pje-copia-integral | PJE |
| pje-trf5-copia-integral | PJE |
| projudi-go-copia-integral | PROJUDI |
| projudi-pr-copia-integral | PROJUDI |
| stj-jus-copia-integral | PROPRIO |
| tj-rj-copia-integral | PROPRIO |

## 1.3 - INFO

| ID ROBÔ | SISTEMA |
|---|---|
| esaj-info | ESAJ |
| eproc-info | EPROC |
| projudi-go-info | PROJUDI |

| | |
|---|---|
| projudi-ba-info | PROJUDI |
| projudi-mt-info | PROJUDI |
| projudi-pr-info | PROJUDI |
| pje-info | PJE |
| pje-tj-rj-publico-informacoes | PJE |

# 2 - SISTEMAS

## 2.1 - BBJUR

| ID ROBÔ | FUNÇÃO |
|---|---|
| bbjur-distribution-search | |
| bbjur-event-update-api | |
| bbjur-distribuicao | |

## 2.2 - LEGALONE

| ID ROBÔ | FUNÇÃO |
|---|---|
| legalone-tarefa | CADASTRO DE TAREFAS |

## 2.2 - SISJUR

| ID ROBÔ | FUNÇÃO |
|---|---|
| sisjur-pex-andamentos-documentos | INSERE ANDMENTOS E DOCUMENTOS - PEX |
| sisjur-andamentos-inserir-cc | INSERE ANDMENTOS E DOCUMENTOS - CC |
| sisjur-andamentos-inserir-jec | INSERE ANDMENTOS E DOCUMENTOS - JEC |

# Intimações

# App de Intimações

*Referência da API intimações, para automação com acionamento de hooks por API para notificação de eventos.*

O APP de intimações é um serviço de agendamento para execução recorrente dos robôs de intimação da Oystr. Ele executa os robôs de intimação nos dias e horários agendados com as configurações definidas e, ao fim da execução, pode disparar um **webhook** contendo o endpoint para obtenção dos resultados para uma url definida.

## O que são Webhooks e o seu papel no App de intimações?

Os Webhooks são uma forma de um software avisar outro software através de gatilhos e disparos. A melhor maneira de entender isso é pensar nos webhooks como notificações de eventos. Na nossa aplicação de intimações, sempre que um perfil configurado terminar de carregar ou abrir as intimações no tribunal, um disparo de evento será feito através da nossa API para o webhook configurado no software integrado. É desta maneira que as integrações serão feitas no App de Intimações Oystr.




## Perfis de Abertura de Intimações

O App de Intimações opera a partir de perfis. Um perfil representa uma configuração de credenciais, opções, filtros, etc. usada para agendar e iniciar os robôs de intimação. Quando dia e horário especificados no perfil forem cumpridos, o App de intimações executará as seguintes ações:

1. `Listagem`: acionará o robô com as credenciais e opções definidas para obter uma lista das intimações disponíveis. Erros nessa etapa são marcados como `Erro ao listar` na interface do App e uma nova tentativa será realizada no próximo ciclo de 30 minutos contanto que o horário limite configurado não tenha sido atingido. Caso não tenha sido concluída em até 3 horas, será interrompida automaticamente. Para cada cliente são listados até 2 perfis simultâneos no momento.

2. `Filtragem`: aplicará os filtros automáticos definidos sobre a lista de intimações recebida para decidir quais deve abrir
3. `Disparo do webhook opcional`: se definido, enviará um webhook contendo a lista das intimações que serão abertas pelo robô e a lista das intimações que *não* serão abertas pelo robô. Caso ocorra erro ou seja recebido qualquer status de retorno fora da lista `200, 201, 202, 203, 204`, uma nova tentativa será feita 30 minutos depois com um limite de 10 tentativas. O estado atual de envio e consulta fica disponível na aba `Auditoria` seção `Integração` da interface do App.
4. `Abertura`: acionará o robô com as credenciais e opções definidas para realizar a captura das intimações escolhidas na `Filtragem`. As execuções criadas são disponibilizadas na tela inicial do console para a conta que tiver criado a chave de API cadastrada no App de intimações (a chave de API da integradora não influencia esse comportamento). **Somente as execuções criadas pelo App de intimações são passíveis de integração**. Não há limite de execuções simultâneas por parte do App de intimações. A conclusão das execuções é verificada a cada 5 minutos
5. `Obtenção do registro`: recebe o registro de execução do robô e realiza o download dos anexos para integração
6. `Processamento`: se definido, aplicará processamentos extras no registro de execução do robô. Se não especificado, aplicará a padronização dos campos de data com a máscara `DD/MM/YYYY`. A padronização é aplicada apenas aos campos que puderem ser identificados, caso contrário o conteúdo é deixado intacto
7. `Disparo do webhook principal`: enviará um webhook contendo o endpoint de consulta do registro. Caso ocorra erro ou seja recebido qualquer status de retorno fora da lista `200, 201, 202, 203, 204`, uma nova tentativa será feita 30 minutos depois com um limite de 10 tentativas. O estado atual de envio e consulta fica disponível na aba `Auditoria` da interface do App.

# Configurações Gerais

## IMPORTANTE: Antes de Definir o Primeiro Perfil

Antes de começar, é necessário realizar duas configurações/ajustes pela primeira (e única) vez:

* Será necessário criar uma chave de API, caso ainda não tenha sido feito. Veja neste link como fazer.
* Será necessário realizar uma configuração padrão que servirá como modelo base para novos perfis. A configuração padrão deve ser definida em **Perfis**->**Configurações**.

## Configurações Gerais

*Link para cadastrar perfis no painel de Intimações.*

*Link das configurações gerais.*

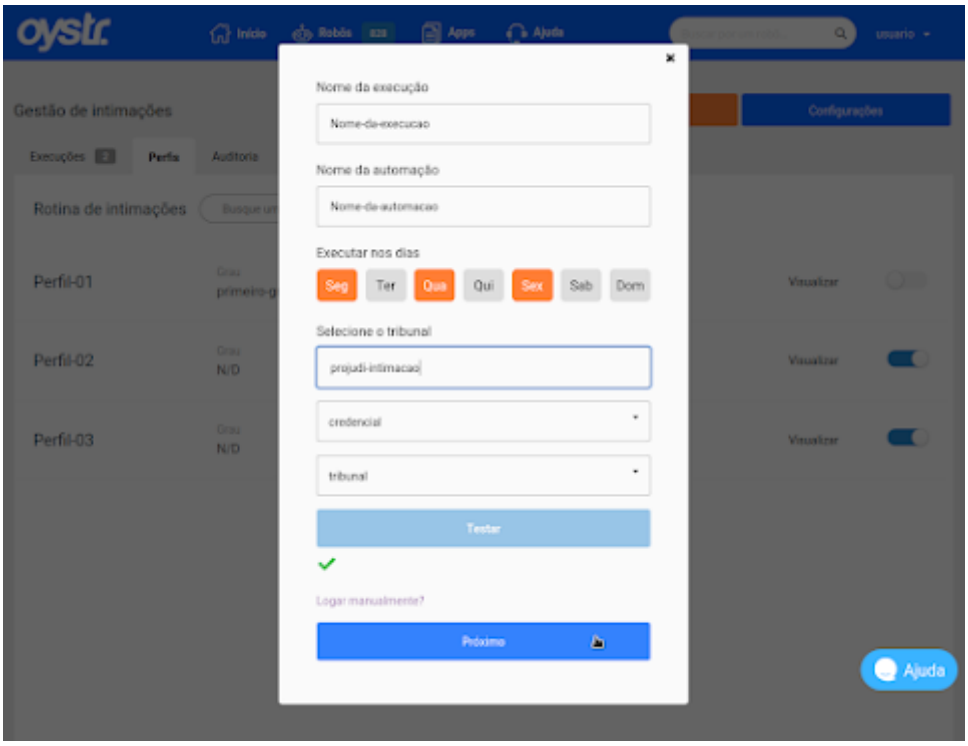
*Formulário das configurações gerais da conta.*

| CAMPO | DESCRIÇÃO |
|---|---|
| **Webhook** | Endpoint de envio do hook principal. Se essa configuração ficar em branco, não haverá disparo no webhook. |
| **JSON Header Webhook** | Headers adicionais a ser incluídos na requisição do hook principal em formato JSON. Isso é usado normalmente para realizar autenticações e garantir a segurança da sua aplicação. |
| **WebhookProto** | Endpoint de envio do hook opcional com a listagem dos itens que serão abertos. Se essa configuração ficar em branco, não haverá disparo no webhook. |
| **JSON Header WebhookProto** | Headers adicionais a ser incluídos na requisição do hook opcional em formato JSON. |
| **Enviar Protohook** | Habilita/desabilita o envio do hook opcional. |
| **Agendado para** | Horário de agendamento geral, quando os perfis irão iniciar suas execuções. |

# Criando um Perfil

## Criando um Perfil de Execução de Intimações

Após realizar as configurações acima, os perfis podem ser criados em **Perfis**->**Criar novo perfil**




*Formulário de novo perfil.*

| CAMPO | DESCRIÇÃO |
|---|---|
| **Nome da execução** | Prefixo que será atribuído as execuções que forem criadas a partir desse perfil. **ATENÇÃO:** Evite usar espaços, caracteres especiais e símbolos. |
| **Nome da automação** | Nome que será atribuído ao perfil/automação que está sendo criado. **ATENÇÃO:** Evite usar espaços, caracteres especiais e símbolos. |
| **Executar nos dias** | Dias da semana onde o perfil deve ser executado. |
| **Selecione o tribunal** | Configurações do robô e credenciais. |
| **Rotina** | Filtro de data que será aplicado às intimações listadas. **Veja na tabela a baixo** como estes valores se aplicam. |
| **Horário limite** | Com essa configuração, o App não iniciará perfis após o horário determinado. O valor padrão é 21h00 do horário de Brasília. |
| **Formatação dos campos de data** | Com essa configuração, o App aplicará um processamento no registro do robô para definir um formato padronizado. A formatação é aplicada aos campos `postDate`, `date`, `assignment`, `periodStart` e `periodEnd`. Caso não especificado, será aplicada a máscara `DD/MM/YYYY` por padrão nos campos que puderem ser identificados. |

| ROTINA | DESCRIÇÃO |
|---|---|
| **Últimos 3 dias** | Apenas as intimações postadas nos últimos 3 dias serão filtradas (selecionadas). Por exemplo, no dia 10/11/2021, apenas as intimações postadas no dia 09/11/2021, 08/11/2021 e 07/11/2021 serão filtradas. |
| **Ontem** | Apenas as intimações postadas no dia anterior (ontem) serão filtradas. Por exemplo, no dia 10/11/2021, apenas as intimações postadas no dia 09/11/2021 serão filtradas. |

| ROTINA | DESCRIÇÃO |
| --- | --- |
| **Ontem e antes** | Apenas as intimações postadas nos dias posteriores (ontem para trás) serão filtradas. Por exemplo, no dia 10/11/2021, as intimações postadas no dia 09/11/2021 para trás serão filtradas. |
| **Últimos X dias** | Apenas as intimações postadas nos últimos X dias serão selecionadas. Por exemplo, no dia 10/11/2021 com X = 2, apenas as intimações postadas nos dias 09/11/2021 e 08/11/2021 serão filtradas. |
| **X dias e antes** | Apenas as intimações postadas há X dias ou mais serão selecionadas. Por exemplo, no dia 10/11/2021 com X = 2, apenas as intimações postadas antes do dia 08/11/2021 serão filtradas. |
| **Todos** | O filtro irá pegar tudo, não levando em consideração a data da postagem. |

**A inicialização do robô pode atrasar em relação ao horário agendado em situações onde uma grande quantidade de perfis está agendada para o mesmo horário ou perfis iniciados anteriormente ainda não tenham finalizado a execução. O disparo dos perfis ocorre num modelo de fila por cliente, sendo executados no máximo 2 perfis por vez.**

# Webhooks

Um webhook é uma requisição `HTTP` enviada automaticamente quando um evento acontece. Isso permite que o próprio servidor notifique o cliente do evento em vez de o cliente perguntar repetidamente. No App de intimações, webhooks são usados para notificar o cliente em dois momentos: após a filtragem da lista de intimações e após o término do processamento dos resultados.

**Para receber webhooks do App de intimações é necessário disponibilizar um endpoint para receber os disparos, que são requisições `POST`.**

## Webhook Principal (conteúdo das intimações)

Para o hook principal (retorno das intimações), o `body` terá o seguinte formato:

```
{
   "configName": "nome-perfil",
   "executionId": "id-execucao",
   "bot": "pje-tj-intimation",
   "credentials": {
      "username": "username",
      "credentialType": "password"
   },
   "resultEndpoint": "/results/id-execucao"
}
```

| CAMPO | DESCRIÇÃO |
| --- | --- |

| | |
|---|---|
| configName | Nome da configuração. |
| executionId | ID da execução. |
| bot | ID do robô para conferência no painel da Oystr. |
| credentials | Resumo das credenciais utilizadas. |
| resultEndpoint | Endpoint para consulta dos resultados. |

O endpoint deve responder com um dos status da lista `200`, `201`, `202`, `203`, `204` em caso de sucesso. Em caso de falha, por qualquer razão, serão realizadas mais 9 tentativas com 30 minutos de espera entre elas. Caso seja preciso reenviar algum webhook basta entrar em contato com o suporte.

**É importante verificar se o endpoint é acessível pelo servidor do App, é comum que requisições partindo de faixas de IP de datacenter sejam bloqueadas.**

# Webhook Opcional (listagem das intimações)

Já para o opcional (listagem das intimações), o **body** terá o formato:

```
{
   "configName": "nome-perfil",
   "bot": "projudi-intimacao",
   "credentials": {
      "username": "username",
      "credentialType": "password"
   },
   "total": 1,
   "filtered": 1,
   "remaining": 0,
   "filteredItems": [{
      "valid": true,
      "data": [...],
      "files": null
   }],
   "remainingItems": [{
      "valid": true,
      "data": [...],
      "files": null
   }]
}
```

| CAMPO | DESCRIÇÃO |
| --- | --- |
| configName | Nome da configuração |
| bot | ID do robô para conferência no painel da Oystr |
| credentials | Resumo das credenciais utilizadas |
| filteredItems | Itens filtrados de acordo com o filtro selecionado na configuração do perfil. Esses são os itens que serão executados pela automação |
| remainingItems | Itens remanescentes após o filtro ser aplicado. Esses itens não serão executados pela automação |

O endpoint deve responder com um dos status da lista `200`, `201`, `202`, `203`, `204` em caso de sucesso. Em caso de falha, por qualquer razão, serão realizadas mais 9 tentativas com 30 minutos de espera entre elas. Caso seja preciso reenviar algum webhook basta entrar em contato com o suporte.

**É importante verificar se o endpoint é acessível pelo servidor do App, é comum que requisições partindo de faixas de IP de datacenter sejam bloqueadas.**

# Consulta dos resultados

```
GET: https://console4.oystr.com.br/api/v1/service/intimacoes/results/id-execucao
```

* **Lembrando que é necessário fazer essa requisição autenticada com o header X-Oystr-Auth**, veja neste link como fazer.

Ao consultar os resultados através do endpoint enviado no webhook, o retorno esperado é o seguinte:

```
{
   credentialsOption: "UF",
   data: {
      entries: [
         [{
            "offset": [posicao do item],
            "sent": [indicador de o item foi processado],
            "received": [indicador se o item teve resposta],
            "type": [TIPO DA RESPOSTA],
            "start": [data em millis],
```

```
            "duration": [timestamp],
            "request": {
               /** DADOS DE ENTRADA DO ROBÔ **/
            },
            "response": {
               /** DADOS DE SAÍDA DO ROBÔ **/
            }
         } /** ... N entradas **/ ]
      ],
   }
}
```

# "type": "RES"

Abaixo o modelo dos dados contendo uma intimação:

```
{
   credentialsOption: "UF",
   data:
   {
      entries:
      [
         [{
            "offset": 0,
            "sent": true,
            "received": true,
            "type": "RES",
            "start": 1613484436618,
            "duration": 19831,
            "request":
            {
               "id": "[string]",
               "integration": "[string]",
               "process": "[número do processo]",
               "type": "[string]",
               "withText": "1",
               "removeDigitalCopy": "1",
               "grau": "[string]",
               "url": "[string]",
```

```
    "date": "[data da intimação]",
    "readerDate": "[string]",
    "tribunal": "[string]",
    "party": "[parte intimada]",
    "event": "[evento/tipo do movimento]",
    "postDate": "[string]",
    "lastDate": "[string]",
    "typeReader": "01-1",
    "owner": "[id]",
    "codAdv": "[advogado]",
    "autor": "[string]",
    "reu": "[string]",
    "idPendencia": "[string]",
    "codigoPendencia": "[string]",
    "abrirIntimacao": "[string]",
    "dataHora": "[string]",
    "erroComarca": "[string]",
    "meioComunicacao": "[string]",
    "leitor": "[string]"
},
"response":
{
    "notices":
    [{
        "id":"",                        // Identificação única no site do tribunal, se houver
        "processNumber":"[string]",     // Número do processo
        "origin":"[string]",            // Diário Eletronico de origem
        "author":"[string]",            // Autor
        "defendant":"[string]",         // Réu
        "postDate":"[string]",          // Data de postagem
        "automatic":"[string]",         // Data de leitura automatica
        "date":"[string]",              // Data da intimação
        "period":"[string]",            // prazo
        "processType":"[string]",       // Classe processual
        "periodStart":"[string]",       // Primeiro dia de prazo
        "periodEnd":"[string]",         // Último dia de prazo
        "type":"[string]",              // Tipo do diário
        "assignment":"[string]",        // Data da distribuição
        "personal":[boolean],           // Pessoal
        "juizo":"[string]",             // Juízo
```

```
                "fulfillmentDate":"[string]",   // Data de cumprimento
                "reader":""[string],            // Leitor
                "status":"[string]",            // Situação/Status
                "document":"[string]",          // Título documento da publicação
                "update":"[string]",            // Movimento/Publicação
                "lawyer":"[string]",            // Advogado
                "link":"[string]",              // Link
                "attachments":                  // Array de anexos,
                [{
                    "error":false,          // Indicação de erro
                    "name":"file-vWz.pdf",  // Nome do arquivo
                    "data":"[string]",      // Conteúdo do arquivo em base64
                    "text":"[string]",      // Texto extraído do arquivo
                    "description":null      // Descrição do arquivo
                }],
                "singleFile":null      // Uso interno, ignorar
            }],
            "total": 1,
            "withText": "1",
            "notFound": false,
            "error": false,
            "errorMessage": null,
            "exp": null
        }
      }]
    ],
  }
}
```


| CAMPO | DESCRIÇÃO |
| --- | --- |
| data->entries[][] | Array contendo todos os itens de intimação que o robô tentou capturar, com todas as informações para recuperar a entrada (intimação a capturar) e saída do robô (resultado) |

| CAMPO | DESCRIÇÃO |
| --- | --- |
| data->entries[x][x]->type | Tipo da resposta que estará contido no objeto response deste item. Para cada tipo, campos diferentes serão trazidos:<br><br>- RES - Resposta padrão com os dados de uma intimaçao. **Todos os outros tipos de resposta representam erros (e subsequente ausência do objeto `response` ****, sendo os mais comuns ERR e REQ***<br>- ERR - Resposta com erro, onde o robô não conseguiu capturar a intimação. Neste caso o "response" será nulo. Existirá um objeto "error" com as informações tecnicas sobre o erro.<br>- REQ - Item sem resposta por parte do robô, neste caso o objeto `response` será nulo. |

# Modelo de objeto resposta com erros/exceções

**IMPORTANTE**: Estes cenários onde a resposta não é **RES** podem, ou não, representar problemas para o cliente dependendo do erro e da configuração dos seus perfis (se houve ciência dada na intimação, qual configuração de filtro e agendamento o perfil possui, se mais perfis podem vir a abrir a intimação, etc.), **sendo recomendada a verificação com o suporte para determinar se a situação representa algum risco de perda de prazo**. Dentro do objeto *request* você sempre irá encontrar as informações preliminares da intimação, sendo possível assim saber qual foi o processo, evento e data da intimação com erro no robô.

# "type": "ERR"

Quando o type for "**ERR**", o robô encontrou um erro ao tentar ler/abrir a intimação indicada. **Veja que o retorno não terá o objeto `response` mas sim terá um objeto `error`**.


```
{
  "offset": 10,
  "sent": true,
  "received": true,
  "type": "ERR",
  "start": 1626715329826,
  "duration": 25537,
  "request": {
    "id": null,
    "integration": null,
    "process": "[número do processo]",
    "type": "novas",
```

```
      "withText": "1",
      "removeDigitalCopy": "1",
      "grau": "primeiro-grau",
      "url": null,
      "date": "[data da intimação]",
      "readerDate": null,
      "tribunal": "JFRS",
      "party": "[parte intimada]",
      "event": "Expedida/certificada a intimação eletrônica",
      "postDate": "Abrir Prazo",
      "lastDate": "",
      "typeReader": "01-1",
      "owner": "[id]",
      "codAdv": "[advogado]",
      "autor": null,
      "reu": null,
      "idPendencia": null,
      "codigoPendencia": null,
      "abrirIntimacao": null,
      "dataHora": null,
      "erroComarca": null,
      "meioComunicacao": null,
      "leitor": null
    },
    "error": {
      "message": "curl error: 16",
      "stackTrace": "xingu.http.client.HttpException: curl error: 16\n\tat
xingu.http.client.impl.SimpleHttpRequest.exec(SimpleHttpRequest.java:63)\n\tat
xingu.http.client.impl.StatefullRequest.exec(StatefullRequest.java:35)\n\tat
oystr.eproc.intimacao.controllers.EprocController.pesquisarProcesso(EprocController.java:283)\n\tat
oystr.eproc.intimacao.controllers.EprocPendentesWithoutReaderController.execute(EprocPendentesWithoutRead
erController.java:57)\n\tat oystr.eproc.intimacao.EprocIntimacaoBot.execute(EprocIntimacaoBot.java:152)\n\tat
oystr.eproc.intimacao.EprocIntimacaoBot.execute(EprocIntimacaoBot.java:37)\n\tat
oystr.broker.client.MessageHandler.onRequest(MessageHandler.java:233)\n\tat
oystr.broker.client.MessageHandler.processAsync(MessageHandler.java:139)\n\tat
oystr.broker.client.MessageHandler.access$000(MessageHandler.java:50)\n\tat
oystr.broker.client.MessageHandler$1.run(MessageHandler.java:95)\n\tat
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)\n\tat
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)\n\tat
java.lang.Thread.run(Thread.java:748)\n"
```

```
  }
}
```

# "type": "REQ"

Quando o type for "**REQ**", o robô não executou o item em questão. Estes casos server para apontar problemas de configuração ou comunicação com o tribunal. Caso esse problema seja recorrente, indica-se que seja verificado a situação da configuração/perfil no App para evitar perda de prazos. **Veja que o retorno não terá o objeto `response` e que o valor de `received` é `false`.**


```
{
  "offset": 8,
  "sent": true,
  "received": false,
  "type": "REQ",
  "start": 1626775895441,
  "request": {
    "id": null,
    "integration": null,
    "process": "[número do processo]",
    "type": "novas",
    "withText": "1",
    "removeDigitalCopy": "1",
    "grau": "primeiro-grau",
    "url": "[url...]",
    "date": "[data da intimação]",
    "readerDate": null,
    "tribunal": "PR",
    "party": "[parte intimada]",
    "event": "JUNTADA DE ATO ORDINATÓRIO (10 dias úteis)",
    "postDate": "-",
    "lastDate": "-",
    "typeReader": "novas",
    "owner": null,
    "codAdv": null,
    "autor": null,
    "reu": null,
    "idPendencia": null,
    "codigoPendencia": null,
```

```
    "abrirIntimacao": null,
    "dataHora": null,
    "erroComarca": null,
    "meioComunicacao": null,
    "leitor": null
  }
}
```


# Outras observações

- Dependendo das configurações dos perfis do cliente pode haver a abertura da mesma intimação múltiplas vezes. O cliente pode acionar o suporte para fazer a identificação da(s) causa(s) e as mudanças necessárias mas, caso seja necessária a implementação de um mecanismo de controle por parte da integradora, recomendamos que a intimação seja identificada através do hash de certas informações do elemento do array `notices`.
- O registro da execução do robô é sempre disponibilizado para integração, incluindo o envio do webhook, ainda que a execução tenha 100% de erro. A integradora pode notificar o cliente das ocorrências de erro (recomendamos a inclusão do id de execução na notificação, caso ocorra) mas o ideal é que o cliente acompanhe o painel da Oystr com frequência para o acompanhamento desses casos e acionamento do suporte caso necessário.
- Os registros das execuções ficam disponíveis mesmo após a consulta. O App de intimações armazena o status de envio e consulta dos registros mas **não** impede o acesso após o registro da consulta. Caso um registro esteja indisponível ele pode ter sido movido para armazenamento frio e podemos recuperar esses registros caso necessário.

# Novo formato de anexos (beta)

Devido à inclusão de anexos inteiros em base64 no documento de integração o tamanho do arquivo pode atingir valores exorbitantes. Para contornar esse problema, um novo formato de anexos foi implementado em que são fornecidos endpoints para download posterior dos documentos, permitindo separar o processamento desses itens para um momento oportuno.

Todos os documentos seguem tendo o mesmo formato mas com pequenas mudanças. O conteúdo do webhook passará a ter as seguintes informações:

```
{
   "configName": "nome-perfil",
   "executionId": "id-execucao",
   "bot": "pje-tj-intimation",
   "credentials": {
      "username": "username",
      "credentialType": "password"
   },
   "resultEndpoint": "/results/id-execucao", // Ainda contém o arquivo no modelo anterior sem nenhuma mudança
   "testingResultEndpoint": "/results/testing-id-execucao" // Campo temporário com o journal no novo modelo, mais adiante passaremos a disponibilizar em resultEndpoint
}
```

A integradora pode então realizar a requisição do documento no endpoint em `testingResultEndpoint`. Caso haja qualquer problema, o arquivo no modelo anterior estará disponível no endpoint em `resultEndpoint`. Já os anexos em `attachments` passarão a ter o seguinte modelo:

```
[{
   "type": "link|raw",      // será "raw" caso o arquivo esteja presente em base64 no campo "data"; ou "link" caso o endpoint de download esteja presente no campo "link"
   "error":false,           // Indicação de erro
   "name":"file-vWz.pdf",   // Nome final do arquivo (pode ser diferente da extensão indicada no link pois, caso sejam arquivos html, convertidos para pdf antes do download)
   "data":"[string]|null",  // Conteúdo do arquivo em base64, somente presente caso "type" seja "raw", do contrário será null
   "text":"[string]",       // Texto extraído do arquivo
   "link": "/document/id-execucao/0/file-vWz.html|null", // Endpoint de download, somente presente caso "type" seja "link", do contrário será null
   "description":null       // Descrição do arquivo
}],
```

Caso o campo `type` seja `raw`, o anexo é compatível com o formato anterior e o documento completo estará em `data` em base64. Caso seja `link`, `data` será `null` e o documento em si deverá ser obtido posteriormente no endpoint em `link`.

# Autenticação 2 Fatores

## Cadastro de uma nova chave A2F

POST: https://api4.oystr.com.br/v1/service/vault/totp


```
{
   "bot": "sample",
   "name": "chavedeexemplo",
   "url":
"otpauth://totp/chavedeexemplo?algorithm=SHA1&digits=6&period=30&secret=GQ7FWN3CIM4D6OKRF53TKTZ
FMI7HGSSQ",
   "username": "12312312312"
}
```


Este endpoint irá cadastrar uma chave de autenticação 2 fatores para um usuário.

- **Tipo requisição:** POST;
- **Tipo resposta:** SÍNCRONA;
- **Body:** json

| Parâmetro | Tipo | Descrição |
|---|---|---|
| **bot** | *String (obrigatório)* | Identificador interno do bot. Para cadastro de A2F do Projudi, utilize `projudi-intimacao` |
| **name** | *String (obrigatório)* | Nome que será atribuído a chave. **Campo restringido a apenas** caractéres de `a - z` (em minúsculo) e com tamanho máximo de 24 caractéres. |
| **url** | *String (obrigatório)* | URL no padrão otpauth para registro da chave secreta. A composição desta chave irá contemplar o nome da chave informada no campo `name` e também a chave secreta no padrão ( **todos os caractéres em maiúsculo e sem espaçamento**, por exemplo: `GQ7FWN3CIM4D6OKRF53TKTZFMI7HGSSQ`) |

| Parâmetro | Tipo | Descrição |
| --- | --- | --- |
| **username** | *String (obrigatório)* | Identificador do usuário na plataforma que irá utilizar esse cadastro A2F. **Em caso de tribunal como Projudi**, normalmente é o CPF contendo apenas digitos, sem espaçamento e `.` `-`. |

# Listando todas as chaves A2F

```
GET: https://console4.oystr.com.br/api/v1/service/vault/totp
```

Este endpoint irá listar todas as chaves de autenticação 2 fatores para uma conta.

- **Tipo requisição:** GET;
- **Tipo resposta:** SÍNCRONA;
- **Resposta:** json

# Distribuição

## Webhook

Um webhook é uma requisição HTTP enviada automaticamente quando um evento acontece. Isso permite que o próprio servidor notifique o cliente do evento em vez de o cliente perguntar repetidamente. No App de distribuição, webhooks são usados para notificar o cliente quando os mesmos recebem novas distribuições.

**Para receber webhooks do App de distribuição é necessário disponibilizar um endpoint para receber os disparos, que são requisições** POST

**Todas requisições internas precisam ter a propriedade** X-Oystr-Auth **no** headers, **recebendo o token que pode ser adquirido nas configurações do painel do Console como chave de API.**

Ao receber novas distribuições, os webhooks notificarão a url cadastrada com o seguinte corpo.


```
{
  "webhookId": "id",
  "entity": {
    "type": "user",
    "id": "id",
    "username": "username"
  },
  "lastUpdateAt": null,
  "lastViewedAt": null,
  "endpoints": {
    "result": "/webhook/:id/results",
    "callback": "/webhook/:id/callback"
  }
}
```


| CAMPO | DESCRIÇÃO |
| --- | --- |

| | |
|---|---|
| webhookId | Id do webhook |
| entity | Dados internos |
| lastUpdateAt | Última notificação realizada |
| lastViewedAt | Última visualização de novas distribuições |
| endpoints | Endpoints de auxílio para visualização dos resultados |

# Ajuda

Para ter acesso às rotas da integração, é possível fazer uma requisição `<span data-slate-string="true">GET</span>` com final `<span data-slate-string="true">/help</span>`

GET: https://console4.oystr.com.br/api/v1/service/disthook/webhook/help

**RESPOSTA**


```
[
  {
    "method": "GET",
    "endpoint": "/webhook/help",
    "description": "Descrição de cada rota"
  },
  {
    "method": "GET",
    "endpoint": "/webhook/:id",
    "description": "Informações do webhook"
  },
  {
    "method": "POST",
    "endpoint": "/webhook/:id/results",
    "description": "Resultado do webhook e confirmação de visualização"
  },
  {
    "method": "GET",
    "endpoint": "/webhook/:id/callback",
    "description": "Verificação do webhook"
  },
  {
    "method": "GET",
```

```
    "endpoint": "/webhook",
    "description": "Busca de webhooks"
  },
  {
    "method": "POST",
    "endpoint": "/webhook",
    "description": "Criação de webhook"
  },
  {
    "method": "PUT",
    "endpoint": "/webhook/:id",
    "description": "Modificação de webhook"
  },
  {
    "method": "DELETE",
    "endpoint": "/webhook/:id",
    "description": "Remoção de webhook"
  }
]
```


## Informações

```
GET: https://console4.oystr.com.br/api/v1/service/disthook/webhook/:id
```

Para ter acesso às informações de um webhook já cadastrado, é possível fazer uma requisição GET com final /:id. O retorno trará as seguintes informações:


```
{
  "id": "id",
  "carteiraId": null,
  "userId": "id",
  "needView": false,
  "needNotify": false,
  "lastStatusCode": null,
  "lastResponse": null,
  "lastUpdateAt": null,
  "lastViewedAt": null,
```

```
  "url": "url",
  "headers": "{}",
  "createdAt": "2024-01-04T19:06:40.870Z",
  "updatedAt": "2024-01-04T19:06:40.870Z",
  "deletedAt": null,
  "entity": {
    "type": "user",
    "id": "id",
    "username": "username"
  }
}
```


| CAMPO | DESCRIÇÃO |
| --- | --- |
| id | Id do webhook |
| carteiraId | Id da carteira vinculada |
| userId | Id do usuário vinculado |
| needView | Booleano para a necessidade de visualização de novos dados |
| needNotify | Booleano para a necessidade de notificação no endpoint definido |
| lastStatusCode | Último statusCode da requisição de notificação feita |
| lastResponse | Última resposta da requisição de notificação feita |
| lastUpdateAt | Última notificação realizada |
| lastViewedAt | Última visualização de novas distribuições |
| url | Url `POST` de disparo de notificação |
| headers | Parâmetros para a url de disparo de notificação |
| createdAt | Data de criação do webhook |
| updatedAt | Data de atualização do webhook |
| deletedAt | Data de deleção do webhook |
| entity | Dados internos |

## Resultados (novas distribuições)

POST: https://console4.oystr.com.br/api/v1/service/disthook/webhook/:id/results

Para ter acesso às novas distribuições que foi notificado por um webhook, é possível fazer uma requisição `<span data-slate-string="true">POST</span>` com final `<span data-slate-string="true">/:id/results</span>`. **Apenas serão mostradas distribuições recebidas após a última notificação, distribuições notificadas anteriormente e que não foram vistas não aparecerão.** O retorno trará as seguintes informações:


```
{
  "id": "id",
  "dataDistribuicao": "2024-01-10T11:23:24-03:00",
  "termo": "Lorem",
  "tribunal": "LI",
  "numeroProcesso": "0000000-00.0000.0.00.0000",
  "tipoOcorrencia": "Lorem",
  "reu": "Lorem Ipsum",
  "autor": "Lorem Ipsum",
  "forum": "LI-4",
  "vara": "2ª Lorem Ipsum",
  "cidade": "LOREM",
  "uf": "LI",
  "valorOcorrencia": "20000.000",
  "advogadoautor": "Lorem Ipsum",
  "publicacao": null,
  "linkDocumentosIniciais": "https://loremipsum.com.br/lorem.pdf",
  "processoEletronico": "0000000-00.0000.0.00.0000",
  "tipoProcesso": "Lorem Ipsum",
  "instancia": "1",
  "assunto": "Lorem Ipsum",
  "juiz": null,
  "createdAt": "2024-01-11T13:35:21.420Z"
}
```


Sem o parâmetro `all` passado via query, direto pela url, esse endpoint apenas retornará as distribuições novas. Passando o parâmetro citado, o endpoint passará a retornar todas as distribuições do usuário ou da carteira, dependendo do tipo de vínculo do webhook, dessa forma, é possível filtrar elas com os seguintes parâmetros, esses passados pelo `body`.

```
{
  "from": "0",
  "size": "20",
  "calendarInterval": "day",
```

```
    "year": "2024",
    "month": "1",
    "day": "1"
}
```

| CAMPO | DESCRIÇÃO | **Valor padrão** |
|---|---|---|
| from | Índice inicial de corte para retorno das distribuições (paginador) | 0 |
| size | Índice final de corte para retorno das distribuições (paginador) | 20 |
| calendarInterval | Tipo de data de filtro das distribuições (year, month, day) | year |
| year | Ano da distribuição | null |
| month | Mês da distribuição | null |
| day | Dia da distribuição | null |

## Verificação

```
GET: https://console4.oystr.com.br/api/v1/service/disthook/webhook/:id/callback
```

Para saber se há novas distribuições a serem visualizadas, é possível fazer uma requisição GET com final /:id/callback. O retorno trará a seguinte informação:

```
{
  "needView": false
}
```

## Busca

```
GET: https://console4.oystr.com.br/api/v1/service/disthook/webhook
```

Para buscar por webhooks já cadastrados na conta, é possível fazer uma requisição GET com final /.

## Criação

```
POST: https://console4.oystr.com.br/api/v1/service/disthook/webhook
```

Para criar um webhook, é possível fazer uma requisição `<span data-slate-string="true">POST</span>` com final `<span data-slate-string="true">/</span>`. O `<span data-slate-string="true">body</span>` terá o

seguinte formato:

```
{
  "url": "url",
  "headers": {},
  "carteiraId": "id"
}
```

| CAMPO | DESCRIÇÃO |
| --- | --- |
| url | Url `POST` de disparo de notificação (obrigatório) |
| headers | Parâmetros para a url de disparo de notificação (opcional) |
| carteiraId | ID interno da carteira que deseja notificação |

## Atualização

```
PUT: https://console4.oystr.com.br/api/v1/service/disthook/webhook/:id
```

Para atualizar um webhook, é possível fazer uma requisição PUT com final /:id. O body terá o seguinte formato:

```
{
  "url": "url",
  "headers": {}
}
```

## Deleção

```
DELETE: https://console4.oystr.com.br/api/v1/service/disthook/webhook/:id
```

Para deletar um webhook, é possível fazer uma requisição `DELETE` com final `/:id`