INTERACT CTI

Manual de Integração

Dígitro

INTELIGÊNCIA · TI · TELECOM

Manual de Integração Interact CTI
Release: 2.18

© 2021

por
DÍGITRO TECNOLOGIA S.A.
Seção de Documentação - Departamento Técnico
Rua Profª Sofia Quint de Souza, 167 - Capoeiras
CEP 88085-040 - Florianópolis - SC
www.digitro.com

Florianópolis, setembro de 2021.

SUMÁRIO

1

SOBRE ESTE DOCUMENTO

BEM-VINDO

Este manual descreve o **Interact CTI**, uma ferramenta de integração para controle e monitoração dos recursos do **Interact**.

OBSERVAÇÕES IMPORTANTES

1. Ficará a critério da Dígitro disponibilizar, através de proposta de fornecimento ou contrato de suporte específico, facilidades adicionais que venham a ser criadas.
2. Serviços solicitados pelo cliente, que impliquem em alterações de características específicas, funções adicionais ou outros itens não especificados, serão considerados como adicionais, e serão efetuados conforme cronograma de execução e alocação de recursos, elaborados pela Dígitro e aprovados pelo cliente, através de proposta comercial.
3. Toda funcionalidade identificada com a palavra Opcional, não faz parte da solução. Seu fornecimento depende de proposta específica.
4. A Dígitro, como qualquer empresa desenvolvedora, não pode garantir que softwares não contenham erros ou que o cliente seja capaz de operá-los sem problemas ou interrupções, por isso, não assume eventuais prejuízos financeiros decorrentes dessas falhas ou de problemas de responsabilidade de terceiros.
5. Devido ao desenvolvimento contínuo de técnicas de invasão e ataques à rede, não é possível garantir que o equipamento (hardware e software) esteja livre da vulnerabilidade da invasão/ação externa.
6. Após o aceite ou a entrada em operação do sistema, se ocorrerem erros ou falhas, estes somente serão avaliados e/ou corrigidas mediante contrato de suporte ou autorização para pagamento de suporte avulso, conforme a tabela de preços vigente na data da solicitação.
7. A Dígitro não atualizará este produto em função de novas versões, sendo necessária, para isso, negociação comercial.
8. As informações preenchidas nos campos das janelas exibidas e descritas nesse manual são apenas para ilustração.

9. A configuração do aplicativo depende dos itens adquiridos pelo cliente. O manual descreve a versão mais atual do aplicativo. Assim, poderão existir versões de aplicativos diferentes da versão descrita nesse manual.
10. O Java Runtime Environment, de distribuição gratuita através da Internet, está disponível no site da Sun MicroSystems, e possui um acordo com suas normas de utilização. Custos referentes a alterações nas normas de utilização do aplicativo Java (ou substituto) não serão em hipótese alguma repassados à Dígitro, ficando a cargo do cliente a aceitação ou não do novo acordo de utilização.
11. A Dígitro não se responsabiliza por perdas de informações, devido a não observação, por parte do cliente, de procedimentos de backup, orientando para que, regularmente, armazene os dados também em mídia eletrônica (CD, DVD etc.) de forma a possuir contingência externa.
12. As senhas de acesso são estabelecidas pelo administrador e de sua inteira e exclusiva responsabilidade.
13. A Dígitro não assume qualquer responsabilidade por alterações promovidas por terceiros, autorizados pelo administrador ou não, por falta de cuidado na seleção dos procedimentos de segurança, por vazamento de senhas ou de qualquer outro procedimento operacional do administrador.
14. A Dígitro mantém um processo de ciclo de vida de seus produtos devido a inovações tecnológicas, necessidade de mercado ou outro motivo. Para mais informações, acesse o ambiente exclusivo para clientes em www.digitro.com.br.

2

INTRODUÇÃO

O **Interact CTI** é uma ferramenta de integração para controle e monitoração dos recursos do **Interact** e, nesta versão, provê interface para:

- Sincronismo de tela de agentes.
- Controle de chamadas.
- Gravação.
- Supervisão de estatísticas e alarmes.
- Discador.

NOTA

Esses recursos necessitam de licença para operar.

ARQUITETURA BASEADA EM REST

O **Interact CTI** foi desenvolvido com base em um modelo REST (*Representational State Transfer*).

O modelo REST pode ser considerado um conjunto de princípios e recomendações para a arquitetura de disponibilização de serviços em rede. A abordagem REST possui uma série de vantagens, como simplicidade de uso, compreensibilidade, minimização do tamanho das requisições e melhor aproveitamento do protocolo HTTP.

Soluções totalmente aderentes aos princípios REST são comumente chamadas de RESTFul. Uma das recomendações da abordagem REST é a utilização dos métodos PUT e DELETE do protocolo HTTP para, respectivamente, realizar funções de alteração e exclusão de recursos.

Entretanto, tais verbos podem ser bloqueados por alguns servidores e, também, não são diretamente permitidos em alguns navegadores (*browsers*). Para evitar problemas, as funcionalidades do **Interact CTI** utilizam apenas os verbos GET e POST, não sendo, portanto RESTFul.

ASSINATURA DE EVENTOS

O **Interact CTI** implementa um modelo em que um cliente assina os eventos operacionais que deseja monitorar. As assinaturas podem, também, incluir filtros para agentes, serviços e tipos de mídia. Assim, é possível, por exemplo, criar uma assinatura para monitorar apenas eventos que ocorrerem na mídia de voz ou, ainda, monitorar apenas eventos operacionais relacionados a um determinado serviço, ou para um determinado conjunto específico de agentes.

As assinaturas podem ser feitas para eventos de agentes (*login*, *logout*, mudança de estado ou de configuração), eventos de chamadas (recepção, atendimento, colocação/retirada de espera ou liberação, entre outros), eventos de alarme ou eventos de supervisão de estatísticas de agentes e serviços.

Existem duas formas de receber os eventos:

- por conexão HTTP persistente (canal).
- ou por meio de um webhook.

Monitoração de eventos por canal

Neste modo, os eventos de uma assinatura são transmitidos por um canal. Cada assinatura deve indicar um canal para o envio dos eventos. É possível ter múltiplas assinaturas indicando o mesmo canal.

Os canais são estabelecidos por meio de conexões persistentes. Os eventos são enviados nessas conexões segundo o modelo de Server Sent Events (SSE), cuja especificação pode ser encontrada em http://www.w3.org/TR/eventsource.

NOTA

Basicamente, o SSE especifica uma forma de se manter um canal permanentemente conectado entre um servidor HTTP e um cliente, o qual pode ser um navegador (browser). Por meio desse canal, o servidor envia pacotes de dados onde constam a identificação do tipo de evento e um conjunto de dados associados. Os navegadores mais novos já suportam SSE em um objeto javascript denominado EventSource, que permite associação direta de métodos (listeners) para tratar os eventos recebidos.

A figura a seguir ilustra como uma aplicação cliente deve proceder para realizar a assinatura e monitoração de eventos por canal.

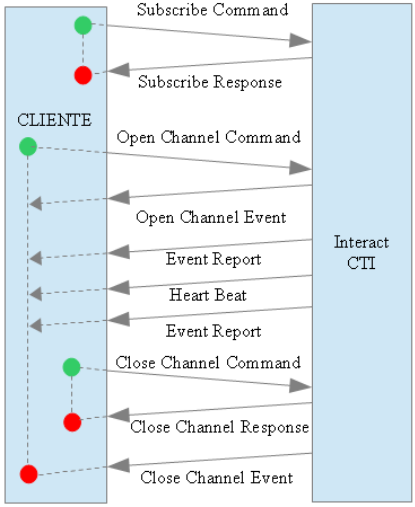

Figura 1. Assinatura e monitoração de eventos

Onde:

- Abertura de conexão.
- Fechamento de conexão.

Inicialmente a aplicação cliente deve enviar um comando de assinatura (Subscribe Command), no qual informará um número de canal para recepção de eventos. Esse é um número que o cliente pode determinar dentro do intervalo de 1 a 9999. Nesse comando de assinatura, o cliente pode informar filtros de quais eventos operacionais

deseja receber, bem como informar quais agentes, serviços, estatísticas, alarmes e tipos de mídia deseja monitorar.

Após um comando de assinatura bem sucedido, a aplicação cliente deve enviar um comando de abertura de canal (Open Channel Command) com o número informado no comando de assinatura. Será, então, enviado um primeiro evento (Open Channel Event) para sinalizar a abertura bem sucedida do canal. A partir desse momento, todos os eventos que passarem nos filtros da assinatura serão enviados à aplicação cliente.

Caso não haja envio de eventos por mais de 15 segundos (tempo *default*), será enviado um pacote de dados vazio com o objetivo único de preservar a conexão aberta. Esse pacote é denominado **Heart Beat** (batida de coração) e está representado na Figura 1.

A conexão será fechada quando a aplicação cliente enviar um comando de fechamento de canal (Close Channel Command), conforme representado na Figura 1. O canal também será fechado se ficar mais de 1 minuto sem assinatura associada.

Alternativamente, é possível delegar a gestão do identificador de canal (channel_id) ao **Interact CTI.** Neste caso, deve-se primeiramente abrir a conexão sem informar o identificador de canal (channel_id). Com isso, o **Interact CTI** alocará automaticamente um identificador disponível na faixa de 1 a 9999 e o informará no atributo *channel_id* no evento de abertura de canal (on_open_channel). De posse desse indentificador, deve-se realizar a assinatura informando-o no atributo *channel_id* para que os eventos possam ser devidamente direcionados ao canal.

Monitoração de eventos por webhook

Neste modo, os eventos de uma assinatura são remetidos para um serviço de responsabilidade do cliente por meio do método HTTP POST. Para tanto, deve-se informar, em um atributo *webhook* na assinatura, a URL completa onde o serviço do cliente espera receber os eventos como, por exemplo: http://webserviceclient:8080/update_agent_info. Para cada evento será enviado um HTTP POST.

NOTA

*Para que o envio de webhook funcione é necessário que a URL informada na assinatura seja acessível para a plataforma do **Interact CTI**. Para tanto pode ser necessário contatar o suporte da Dígitro, para que as devidas permissões de rede sejam disponibilizadas.*

INSERÇÃO DE DADOS DO CLIENTE

Em todos os comandos enviados ao **Interact CTI** é possível informar um parâmetro na URL denominado ***client_data***. Esse comando possibilita que as informações enviadas pelo cliente sejam retornadas em um atributo na resposta do comando.

NOTA

*O atributo **client_data** será retornado sempre que informado como parâmetro na chamada do comando e a execução deste comando resultar em resposta HTTP 200 OK.*

Exemplo em JSON:

http://IP:PORTA/interact_cti/v1.0/agent/get_agents_list?format=json&client_data=exemplo

```
{"response":
   { "resource":"agent",
      "command":"get_agents_list",
       "status":"ok",
       …
```

```
        "client_data":"exemplo"
    }
}
```

Exemplo em XML:

http://IP:PORTA/interact_cti/v1.0/agent/get_agents_list?format=xml&client_data=exemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
    <resource>agent</resource>
  <command>get_agents_list</command>
  <status>ok</status>
   ...
  <client_data>exemplo</client_data>
</response>
```

3

MONITORAÇÃO DE AGENTES E CHAMADAS

Os serviços disponibilizados pelo **Interact CTI** para monitoração de agentes e chamadas (recursos para sincronismo de tela) por meio de REST têm o seguinte formato geral:

interact_cti/v1.0/agent/monitor/metodo?parametros[&format= (json|xml)]

Onde:

- **método:** nome da requisição (comando);
- **parâmetros:** conjunto de pares nome-valor na forma prm=valor, separados por & (E comercial - ampersand);
- **format:** define o formato desejado para a resposta, que pode ser no formato JSON (.json) ou no formato (.xml). Na ausência desse parâmetro, será utilizado o mesmo formato informado no header Content-type. Na ausência do header Content-type, será utilizado XML por *default*.

NOTA

*Na ausência de parâmetros e de formato, o sinal de interrogação após **metodo** é desnecessário.*

As requisições aos serviços serão sempre por meio de métodos **HTTP GET** ou **POST**. O método GET é utilizado em todas as requisições que não enviam apenas parâmetros. O método POST é utilizado em todas as requisições que enviam pacotes de dados (além dos parâmetros). Os parâmetros enviados na URL deverão ser codificados no padrão URL encode. Os dados nos pacotes POST deverão ser codificados em UTF-8.

O acesso se dará na porta específica **8582**, conforme exemplos a seguir:

Comando de assinatura

http://XXX.XXX.XXX.XXX:YYYY/interact_cti/v1.0/agent/monitor/subscribe

Comando de cancelamento de assinatura:

http://XXX.XXX.XXX.XXX:YYYY/interact_cti/v1.0/agent/monitor/unsubscribe

Comando de abertura de canal

http://XXX.XXX.XXX.XXX:YYYY/interact_cti/v1.0/agent/monitor/open_channel

Comando de fechamento de canal

http://XXX.XXX.XXX.XXX:YYYY/interact_cti/v1.0/agent/monitor/close_channel

Onde:

- **XXX.XXX.XXX.XXX:** endereço IP do servidor **Interact CTI**;
- **YYYY:** porta disponibilizada pelo servidor **Interact CTI** (normalmente a porta 8582);

IMPORTANTE

*Caso haja algum erro sintático, será retornado o erro **HTTP 400 Bad Request**. Caso o analisador sintático do **Interact CTI** consiga identificar a posição em que há um erro de sintaxe, será enviado um texto indicando a posição do pacote enviado em que foi detectado o erro (tanto para XML como para JSON).*

Caso o **Interact CTI** não esteja conectado ao processo **Interact**, todos os comandos serão recusados, retornando o erro **HTTP 503 Service Unavailable**.

Caso haja um erro no endereçamento do caminho da URL, será retornado um erro **HTTP 404 Not Found**. Isso ocorre, por exemplo, ao se realizar uma chamada com um erro no nome do recurso **agent**, como a seguir:

interact_cti/v1.0/agente/monitor/unsubscribe?id=1&format=xml

Nesse caso, sinalizando que o recurso endereçado (agente, no caso) não foi encontrado.

ASSINATURA DE EVENTOS PARA MONITORAÇÃO (COMANDO SUBSCRIBE)

O comando **subscribe** do recurso agente possibilita que eventos operacionais sejam assinados. O conceito de assinatura existe para que uma aplicação possa receber apenas os eventos que necessitar.

Método HTTP POST

interact_cti/v1.0/agent/monitor/subscribe?format=[xml | json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
      "subscription" : { "id" : nnnn,
                         "agents": [ lista_de_agentes ],
                         "events": [ lista_de_eventos ],
                         "services": [ lista_de_servicos ],
                         "media_types": [ lista_de_midias ],
                         "expires": tempo_de_expiracao,
                         "channel_id": identificador_do_canal,
                         "webhook": url_do_cliente,
                         "agent_control": controle_de_agente
```

```
                                }
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
   <subscription>
          <id> identificador_da_assinatura </id>
          <agents> lista_de_agentes </agents>
          <events> lista_de_eventos </events>
          <services> lista_de_servicos </services>
          <media_types> lista_de_midias </media_types>
          <channel_id> identificador_do_canal </channel_id>
          <webhook> url_do_cliente </webhook>
          <expires> tempo_de_expiracao </expires>
          <agent_control> controle_de_agente </agent_control>
      </subscription>
</request>
```

Onde:

- **identificador_da_assinatura (opcional):** identificador da assinatura. Deve ser enviado apenas caso se deseje modificar uma assinatura já existente;
- **lista_de_agentes (opcional)**: lista com os nomes dos agentes para monitoração (a ausência desse parâmetro ou o uso do valor **"all" (ou *)** indica monitoração de todos os agentes cadastrados no **Interact**. Será consumida uma licença de sincronismo de tela para cada agente se o

parâmetro **agent_control** for ***false***, ou uma licença de controle de agente se o parâmetro **agent_control** for ***true***);

- **lista_de_eventos (opcional):** lista com os nomes dos eventos de operação para monitoração (a ausência desse parâmetro ou o uso do valor **"all" (ou *)** indica monitoração de todos os eventos). Apenas os eventos de operação (aqueles relacionados a agentes e a chamadas) é que podem ser assinados. Eventos de controle são sempre enviados (Veja nota a seguir);
- **lista_de_servicos (opcional):** lista com o nome dos serviços para monitoração (a ausência desse parâmetro ou o uso do valor **"all" (ou *)** indica monitoração de todos os serviços);
- **lista_de_midias (opcional):** lista de mídias para monitoração (a ausência desse parâmetro ou o uso do valor **"all" (ou *)** indica monitoração de todas as mídias);
- **identificador_de_canal:** identificador do canal de eventos pelo qual os eventos dessa assinatura deverão ser enviados. Os canais podem ter identificadores dentro da faixa numérica desde 1 até 9999;
- **url_do_cliente:** endereço do webservice do cliente que espera receber eventos por HTTP POST;
- **tempo_de_expiracao:** tempo, em segundos, durante o qual a assinatura permanecerá ativa. O valor máximo é 86400, o que corresponde a um dia completo. Esse parâmetro é obrigatório. Valores aceitos entre 1 e 86400;
- **controle_de_agente:** pode ter os valores *true* ou *false*. Se *true* indica que as operações de controle de agente e chamada estarão disponíveis. Nesse caso, será consumida uma licença de controle de agentes CTI por nome na lista de agentes. Se *false* ou ausente indica que não há controle de agente e chamada na assinatura.

NOTA

*Eventos de controle do canal (**on_open_channel** e **on_close_channel**), de disponibilidade do servidor (**on_server_state_change**) e de expiração de assinatura (**on_subscription_expire** e **on_subscription_expire_warning**) são denominados de **Eventos de Controle** e não são passíveis de serem assinados e sempre serão enviados aos clientes.*

Exemplo em JSON

```
{
   "subscription": {     "agents": [ "Jose", "Maria" ],
                         "services":["Atendimento_A",  "Atendimento_B" ],
                         "media_types": ["voice","chat" ],
                         "events":["on_login", "on_logout",
"on_call_receive"],
                         "expires": 3600,
                         "channel_id": 1234
                         },
}
```

Exemplo em XML

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
  <subscription>
      <agents>
```

```
            <member>Jose</member>
            <member>Maria</member>
         </agents>
         <services>
            <member>Atendimento_A</member>
            <member>Atendimento_B</member>
         </services>
         <media_types>
            <member>voice</member>
            <member>chat</member>
         </media_types>
         <events>
            <member>on_login</member>
            <member>on_logout</member>
            <member>on_call_receive</member>
         </events>
         <expires>3600</expires>
         <channel_id>1234</channel_id>
   </subscription>
</request>
```

Descrição do exemplo

A Interface CTI assina a recepção de eventos de **Login** (on_login), **Logout** (on_logout) ou **Recepção de Chamadas** (on_call_receive), que ocorram para os agentes Jose ou

Maria, nas mídias de **Voz** ou **Chat**, para os serviços **AtendimentoClientes_A** ou **AtendimentoClientes_B**.

Em princípio, a assinatura permanecerá válida por **3600 segundos (1 hora)**, caso não seja feita a renovação da assinatura. Isso significa que durante esse período, o **canal 1234**, **estando aberto**, receberá os eventos de operação monitorados e, também, os eventos de controle.

Resposta ao Comando de Assinatura

A resposta para a assinatura de eventos possui o seguinte formato:

RESPONSE DATA em JSON:

```
{ "response":
            { "command": "subscribe",
              "id": identificador_da_assinatura,
              "status": status,
              "error_type": tipo_de_erro,
              "error_items": listas_de_items_com_erro
            }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>subscribe</command>
   <id>identificador_da_assinatura</id>
```

```
   <status>status</status>
   <error_type>tipo_de_erro</error_type>
   <error_items>listas_de_itens_com_erro</error_items>
</response>
```

Onde:

- **identificador_da_assinatura:** identificador numérico único para a assinatura;
- **status:** estado da execução do comando (pode ser **ok** em caso de sucesso, ou **error** em caso de erro);
- **tipo_de_erro:** enviado apenas quando *status* for **error** e pode ser:
  - **empty_data:** enviado quando o pacote de dados estiver vazio;
  - **invalid_data:** enviado quando o pacote de dados estiver em formato inválido;
  - **not_found:** enviado em resposta a um comando de atualização com campo de id inexistente no servidor. Pode ocorrer no caso da tentativa de atualização de uma assinatura que já expirou;
  - **invalid_item:** enviado quando o pacote de dados contiver algum item inválido (como o nome de um agente, o nome de um serviço, o nome de um evento ou o nome de um tipo de mídia);
  - **no_license:** enviado quando não há licenças disponíveis em número suficiente para atender à assinatura.
- **lista_de_itens_com_erro:** enviado apenas quando o *status* for **error** e o **tipo_de_erro** for **invalid_item**. O conteúdo de **error_items** será uma lista de todos os itens que apresentaram erro na assinatura, separados nas tags **agents**, **services**, **events** e **media_*types***.

Exemplo de Resposta para o Caso de Sucesso

RESPONSE DATA em JSON:

```
{
   "response":  {
                        "command":"subscribe",
                        "id": 10,
                        "status":"ok",
                }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>subscribe</command>
   <id>10</id>
   <status>ok</status>
</response>
```

Exemplos de Resposta para Casos de Insucesso

Exemplo 1: RESPONSE DATA em JSON:

```
{
   "response":{
                        "command":"subscribe",
```

```
                    "status":"error",
                    "error_type":"invalid_data"
                }
}
```

Exemplo 2: RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>subscribe</command>
   <status>error</status>
   <error_type>invalid_data</error_type>
</response>
```

Exemplo 3: RESPONSE DATA em JSON:

```
{
   "response":{
                    "command":"subscribe",
                    "status":"error",
                    "error_type":"invalid_item",
                    "error_items":      { "agents":
["erro1","erro2"] },
                                        { "services":
["erro3","erro4"] }
                }
}
```

Exemplo 4: RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>subscribe</command>
   <status>error</status>
   <error_type>invalid_item</error_type>
   <error_items>
           <agents>
               <member>erro1</member>
               <member>erro2</member>
           </agents>
           <services>
               <member>erro3</member>
               <member>erro4</member>
           </services>
      </error_items>
</response>
```

ALTERAÇÃO E RENOVAÇÃO DE ASSINATURA (COMANDO SUBSCRIBE)

Uma assinatura pode ser alterada com o envio de um novo comando de assinatura passando o código identificador da assinatura (**id**). A assinatura anterior será cancelada e a nova ficará ativa em seu lugar.

Para renovação da assinatura, basta que sejam enviados os mesmos parâmetros da assinatura original adicionados do código identificador da assinatura (**id**).

O tempo de expiração será reiniciado para o valor que for informado no campo **expires**. Todos os parâmetros da assinatura podem ser alterados, inclusive o número do canal para envio de eventos.

CANCELAMENTO DE ASSINATURA (COMANDO UNSUBSCRIBE)

Uma assinatura poderá ser cancelada ao se enviar um comando de **unsubscribe** informando o código identificador da assinatura (**id**). Havendo uma assinatura com o **id** informado, ela será cancelada. Caso o id não exista, será retornado um erro **not found**.

Método HTTP POST

interact_cti/v1.0/agent/monitor/unsubscribe?format=[xml | json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
   “id”: identificador_da_assinatura
}
```

POST DATA em XML:

```
<request>
  <id>identificador_da_assinatura</id>
</request>
```

Onde:

- **identificador_da_assinatura**: número idenficador da assinatura recebido na resposta de um comando de assinatura (**subscribe**) bem sucedido.

Resposta ao Comando de Cancelamento de Assinatura

A resposta para o cancelamento da assinatura será uma das formas a seguir:

RESPONSE DATA em JSON:

```
{ "response":
            { "command": "unsubscribe",
              "id": identificador_da_assinatura,
              "status": status,
              "error_type": tipo_de_erro
            }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
```

```
   <command>unsubscribe</command>
   <id>identificador_da_assinatura</id>
   <status>status</status>
   <error_type>tipo_de_erro</error_type>
</response>
```

Onde:

- **identificador_da_assinatura:** identificador numérico para a assinatura;
- **status:** estado da execução do comando (pode ser **ok** em caso de sucesso, ou **error** em caso de erro);
- **tipo_de_erro:** enviado apenas quando *status* for **error** e pode ser:
  - empty_data: enviado quando não forem passados parâmetros na URL do comando;
  - not_found: enviado em resposta a um comando de atualização com campo de id inexistente no servidor. Pode ocorrer no caso da tentativa de cancelamento de uma assinatura que já expirou ou que já foi removida anteriormente;
  - missing_data: enviado quando não for encontrado parâmetro id na URL do comando.

Exemplo de Resposta para o Caso de Sucesso

RESPONSE DATA em JSON:

```
{
   "response":  {
                        "command":"unsubscribe",
```

```
                    "id": 10,
                    "status":"ok",
                }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>unsubscribe</command>
   <id>10</id>
   <status>ok</status>
</response>
```

Exemplo de Resposta para o Caso de Insucesso

RESPONSE DATA em JSON:

```
{
   "response":{
                    "command":"unsubscribe",
                    "id":10,
                    "status":"error",
                    "error_type":"not_found"
                }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>unsubscribe</command>
   <id>10</id>
   <status>error</status>
   <error_type>not_found</error_type>
</response>
```

CONSULTA DE ASSINATURAS (COMANDO GET_SUBSCRIPTIONS)

As assinaturas ativas podem ser consultadas ao se enviar um comando **get_subscriptions**. Esse comando tem dois parâmetros opcionais, um para identificar uma assinatura (**id**) e outro para identificar um canal (**channel_id**).

Informando-se o código identificador da assinatura (**id**), serão retornados os dados da respectiva assinatura. Informando-se o código identificador do canal (**channel_id**), serão retornados os dados das assinaturas existentes para aquele canal. Caso o **id** ou o **channel_id** informados não encontrem correspondências nas assinaturas ativas, será retornado um erro ***not found***.

Método HTTP GET

interact_cti/v1.0/agent/monitor/get_subscriptions?id=nnnn&channel_id=kkkk&format=[xml | json]

O campo **nnnn** deve ser substituído pelo código identificador da assinatura (**id**) que se deseja consultar.

O campo **kkkk** deve ser substituído pelo código identificador do canal (**channel_id**), configurado para a(s) assinatura(s) que se deseja consultar.

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

Resposta ao Comando de Consulta de Assinatura

A resposta para a consulta bem sucedida de assinaturas será uma das formas a seguir:

RESPONSE DATA em JSON:

```
{ "response":
            { "command": "get_subscriptions",
              "status": status,
              "subscriptions": lista_de_assinaturas
            }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>get_subscriptions</command>
   <status>status</status>
   <subscriptions>lista_de_assinaturas</subscriptions>
</response>
```

Onde:

- **status:** estado da execução do comando (pode ser **ok** em caso de sucesso, ou **error** em caso de erro);
- **lista_de_assinaturas:** um conjunto de dados que atendem aos parâmetros de consulta e têm o seguinte formato:

Formato em JSON:

```
[ { dados_assinatura_1 } , { dados_assinatura_2}, … { dados_assinatura_n } ]
```

Formato em XML:

```
<member>dados_assinatura_1</member>
<member>dados_assinatura_2</member>
...
<member>dados_assinatura_n</member>
```

Onde:

- **dados_assinatura_x:** possui a seguinte estrutura:

Em JSON:

```
{ "id" : nnnn,
  "agents": [ lista_de_agentes ],
  "events": [ lista_de_eventos ],
  "services": [ lista_de_servicos ],
  "media_types": [ lista_de_midias ],
  "expires": tempo_de_expiracao,
```

```
  "channel_id": identificador_do_canal,
  "webhook": url_do_cliente,
}
```

Em XML:

```
<?xml version="1.0" encoding="UTF-8">
<id> identificador_da_assinatura </id>
<agents> lista_de_agentes </agents>
<events> lista_de_eventos </events>
<services> lista_de_servicos </services>
<media_types> lista_de_midias </media_types>
<channel_id> identificador_do_canal </channel_id>
<webhook> url_do_cliente </webhook>
<expires> tempo_de_expiracao </expires>
```

Onde:

- **identificador_da_assinatura:** identificador da assinatura;
- **lista_de_agentes:** lista com os nomes dos agentes para monitoração configurados na assinatura;
- **lista_de_eventos:** lista com os nomes dos eventos de operação para monitoração configurados na assinatura;
- **lista_de_servicos**: lista com o nome dos serviços para monitoração configurados na assinatura;
- **lista_de_midias:** lista de mídias para monitoração configuradas na assinatura;

- **identificador_do_canal:** identificador do canal de eventos configurado para a assinatura;
- **url_do_cliente:** endereço do webservice do cliente que espera receber eventos por HTTP POST;
- **tempo_de_expiracao:** tempo em segundos restante para a assinatura expirar.

A resposta para a consulta **mal sucedida** de assinaturas será uma das formas a seguir:

RESPONSE DATA em JSON:

```
{ "response":
            { "command": "get_subscriptions",
              "status": error,
              "error_type": tipo_de_erro
            }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>get_subscriptions</command>
   <status>error</status>
   <error_type>tipo_de_erro</error_type>
</response>
```

Onde:

- **tipo_de_erro:** enviado apenas quando *status* for **error**, e pode ser:
  - *not_found*: enviado em resposta a um comando de consulta de assinaturas com parâmetros que não resultaram em nenhuma correspondência.
  - *missing_data*: enviado quando todos os parâmetros enviados não puderem ser reconhecidos pelo comando de consulta.

Exemplo de Resposta para o Caso de Sucesso

RESPONSE DATA em JSON:

```
{
    "response": {
        "command": "get_subscriptions",
        "status": "ok",
        "subscriptions": [
            {
                "id": 5,
                "events": [
                    "on_login",
                    "on_logout",
                    "on_state_change"
                ],
                "media_types": [
                    "voice",
                    "email"
```

```
                ],
                "expires": 3589,
                "channel_id": 1234
            }
        ]
    }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <command>get_subscriptions</command>
  <status>ok</status>
  <subscriptions>
    <member>
      <id>5</id>
      <events>
        <member>on_login</member>
        <member>on_logout</member>
        <member>on_state_change</member>
      </events>
<media_types>
        <member>voice</member>
        <member>email</member>
      </media_types>
```

```
      <expires>3589</expires>
      <channel_id>1234</channel_id>
    </member>
   </subscriptions>
</response>
```

Exemplo de Resposta para o Caso de Insucesso

RESPONSE DATA em JSON:

```
{
   "response":{
                        "command":"get_subscriptions",
                        "status":"error",
                        "error_type":"not_found"
                  }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>get_subscriptions</command>
   <status>error</status>
   <error_type>not_found</error_type>
</response>
```

ABERTURA DE CANAL PARA RECEPÇÃO DE EVENTOS (COMANDO OPEN_CHANNEL)

A recepção de eventos será sempre realizada por meio de canais. Um canal é uma conexão HTTP persistente, que deverá ser aberta por meio do comando **open_channel**.

Cada canal possui um identificador único (**channel_id**). Esse identificador deve ser informado como um parâmetro passando no comando **open_channel** ou, se não informado, será atribuído automaticamente pelo **Interact CTI**. Em qualquer caso, o **channel_id** é indicado no evento **on_open_channel**.

Na abertura do canal deverá ser informado um formato para a recepção dos dados (XML ou JSON).

O identificador do canal deverá ser um valor na faixa **entre 1 e 9999**, e poderá ser associado a várias assinaturas.

Um canal é fechado automaticamente depois de 1 minuto sem alguma assinatura associada. Isso vale também para o caso de se abrir um canal sem nenhuma assinatura associada. Ele permanecerá aberto por 1 minuto e, caso nenhuma assinatura seja feita nesse intervalo, ele será automaticamente fechado.

Método HTTP POST

interact_cti/v1.0/agent/monitor/open_channel?channel_id=nnnn&format=[xml | json]

POST DATA em JSON:

```
{
   “channel_id”: identificador_do_canal
```

```
}
```

POST DATA em XML:

```
<request>
  <channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

- **identificador_do_canal**: número idenficador do canal que se deseja abrir.

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

NOTA

*Caso se deseje abrir um canal com um identificador (**channel_id**) que já existe ou fora da faixa válida (**entre 1 e 9999**), será retornado o erro **HTTP 403 Forbidden**.*

FECHAMENTO DE CANAIS DE RECEPÇÃO DE EVENTOS (COMANDO CLOSE_CHANNEL)

Um canal pode ser fechado por meio do envio do comando **close_channel**, informando o seu identificador numérico (**channel_id**) e um formato para a recepção dos dados (XML ou JSON).

Ressalta-se que um canal será automaticamente fechado depois de 1 minuto sem alguma assinatura associada, independente do envio do comando **close_channel**.

Método HTTP POST

interact_cti/v1.0/agent/monitor/close_channel?format=[xml| json]

POST DATA em JSON:

```
{
   “channel_id”: identificador_do_canal
}
```

POST DATA em XML:

```
<request>
  <channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

- **identificador_do_canal**: número idenficador do canal que se deseja fechar.

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

Resposta ao Comando de Fechamento de Canal

A resposta para o cancelamento da assinatura será uma das formas a seguir:

RESPONSE DATA em JSON:

```
{ "response":
            { "command": "close_channel",
              "channel_id": identificador_do_canal,
              "status": status,
              "error_type": tipo_de_erro
            }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>close_channel</command>
   <channel_id>identificador_do_canal</channel_id>
   <status>status</status>
   <error_type>tipo_de_erro</error_type>
</response>
```

Onde:

- **identificador_do_canal:** identificador numérico para a assinatura.

- **Status:** estado da execução do comando (pode ser **ok** em caso de sucesso, ou **error** em caso de erro).
- **tipo_de_erro:** enviado apenas quando *status* for **error**, e pode ser:
  - *not_found:* enviado em resposta a um comando de fechamento com campo de **channel_id** inexistente no servidor. Ocorrerá nos casos de tentativa de fechamento de um canal não aberto ou que já foi fechado anteriormente.
  - *empty_data:* enviado quando não forem passados parâmetros na URL do comando.
  - *missing_data:* enviado quando não for encontrado parâmetro **channel_id** na URL do comando.

Exemplo de Resposta para o Caso de Sucesso

RESPONSE DATA em JSON:

```
{
   "response":  {
                        "command":"close_channel",
                        "channel_id": 10,
                        "status":"ok"
                }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
```

```
   <command>close_channel</command>
   <channel_id>10</channel_id>
   <status>ok</status>
</response>
```

Exemplo de Resposta para o Caso de Insucesso

RESPONSE DATA em JSON:

```
{
   "response":{
                        "command":"close_channel",
                        "id":10,
                        "status":"error",
                        "error_type":"not_found"
                  }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>close_channel</command>
   <id>10</id>
   <status>error</status>
   <error_type>not_found</error_type>
</response>
```

MANUTENÇÃO DO CANAL ABERTO

Para que uma conexão HTTP se mantenha de forma persistente é necessário que haja tráfego de pacotes desde o servidor até o cliente (autor da requisição) com certa frequência, caso contrário, a conexão será derrubada por *time-out* (que pode ocorrer tanto em um *browser* como em um servidor intermediário, entre o servidor do **Interact CTI** e a aplicação cliente).

O **Interact CTI** enviará um pacote de dados vazio apenas para sinalizar a atividade do canal e manter a conexão estabelecida.

O pacote de “heart beat” possui o seguinte formato: data: {}

RECEPÇÃO DOS EVENTOS PELO CANAL

Os eventos serão enviados dentro do padrão de Server **Send Events da W3C (http://www.w3.org/TR/eventsource).**

No **Interact CTI**, cada evento será enviado em um pacote de dados no seguinte formato:

```
event: nome_do_evento
data: dados_do_evento
```

Onde:

- *event*: nome do evento;

Para controle do estado do canal:

- on_open_channel;
- on_close_channel;

Para controle da assinatura:

- on_subscription_expire;

Para controle do estado do servidor **Interact**:

- on_server_state_change;

Para o recurso **agent** será um dos seguintes eventos (eventos operacionais que podem ser assinados):

- on_login;
- on_logout;
- on_change_state;
- on_config_change;
- on_queue_change;
- on_call_receive;
- on_call_accept;
- on_call_generate;
- on_call_hold;
- on_call_retrieve;
- on_call_release;
- on_call_consultation;
- on_call_transfer;
- on_call_end;
- on_call_record;
- on_call_associated_data;
- on_call_conference;
- on_chat_message;
- on_typing.

- data: pacote de dados recebido e que estará no formato especificado na abertura do canal (XML ou JSON).

Para os eventos **on_open_channel** e **on_close_channel**, o pacote de dados enviado conterá apenas o identificador do canal (**channel_id**) conforme o exemplo abaixo:

Evento em XML:

```
event: open_channel
data: <channel_id>1234</channel_id>
```

Evento em JSON:

```
event: open_channel
data: {"channel_id":1234}
```

Os eventos do recurso **agent** são apresentados e discutidos em detalhes na seção Eventos Operacionais de Agentes e Chamadas.

EVENTOS OPERACIONAIS DE AGENTES E CHAMADAS

Eventos operacionais são eventos relacionados aos agentes e às chamadas que transitam pelo **Interact** e que podem ser roteados pelo **Interact CTI**. Esse tipo de evento pode ser assinado, ou seja, só será recebido se o integrador assim o desejar. A seguir é apresentada a relação desses eventos:

Evento	Descrição
on_login	Ocorre quando um agente faz o *login.*

Evento	Descrição
on_logout	Ocorre quando um agente faz o *logout*.
on_agent_status	Ocorre quando é feita uma assinatura ou aberto um canal.
on_state_change	Ocorre quando há mudança de estado do agente.
on_config_change	Ocorre quando há mudança de configuração do agente.
on_queue_change	Ocorre na entrada de chamada na fila ou quando há alteração da posição de fila do serviço para qual o agente tem perfil mínimo de atendimento.
on_call_receive	Ocorre quando uma chamada é encaminhada para um agente.
on_call_accept	Ocorre quando uma chamada é aceita/atendida pelo agente.
on_call_generate	Ocorre quando uma chamada é gerada pelo agente.
on_call_release	Ocorre quando uma chamada é finalizada, porém, o atendimento pode continuar (Classificação e/ou Pós-Atendimento).
on_call_hold	Ocorre quando uma chamada é colocada em espera.
on_call_retrieve	Ocorre quando uma chamada é retirada da espera.
on_call_consultation	Ocorre quando é realizada uma consulta sobre uma chamada.
on_call_transfer	Ocorre quando uma chamada é transferida.

Evento	Descrição
on_call_end	Ocorre quando o atendimento é finalizado.
on_call_record	Ocorre quando a gravação é finalizada (após finalização do atendimento). Retorna o identificador da gravação.
on_call_associated_data	Ocorre quando um novo dado é associado à chamada.
on_call_conference	Evento gerado sempre que há uma mudança nos interlocutores de uma conferência (início da conferência, entrada ou saída de novos participantes).
on_chat_message	Ocorre quando chega uma nova mensagem (exclusivo para a mídia *Chat*).
on_typing	Evento gerado sempre que um agente ou interlocutor de *Chat* inicia ou encerra a ação de digitação (exclusivo para mídia *Chat* e dependente de configuração).
on_file_transfer	Evento gerado sempre que ocorre uma ação de transferência de arquivo em uma chamada de *Chat*.
on_video_action	Evento gerado sempre que uma ação de vídeo ocorre uma chamada de *Chat*.
on_inactivity_timer	Evento gerado sempre que é ligado ou deslgado o timer de inatividade do agente para uma determinada chamada de *Chat*.

Todos estes eventos enviam informações comuns sobre o agente, nas formas a seguir:

Em JSON:

```
"agent": nome_do_agente,
```

```
"dt_login": data_hora_de_login,
"state": estado_do_agente,
"state_id": id_do_estado,
"pause_name": nome_da_pausa,
"pause_trigger": gatilho_da_pausa,
"pause_timeout": timeout_da_pausa,
"pre_pause":{
"id":id_da_pre_pausa,
"name": nome_da_prepausa,
"trigger": gatilho_da_prepausa,
"dt_begin":data_hora_entrada_pre_pausa
},
"branch": numero_do_ramal,
"contingency": estado_de_contingencia
```

Em XML:

```
<agent>nome_do_agente</agent>
<dt_login>data_hora_de_login</dt_login>
<state>estado_do_agente</state>
<state_id>id_do_estado</state_id>
<pause_name>nome_da_pausa</pause_name>
<pause_trigger>gatilho_da_pausa</pause_trigger>
<pause_timeout>timeout_da_pausa</pause_timeout>
<pre_pause>
<id>id_da_pre_pausa</id>
```

```
<name>nome_da_pre_pausa</name>
<trigger>gatilho_da_pre_pausa</trigger>
<dt_begin>data_hora_entrada_pre_pausa</dt_begin>
</pre_pause>
<branch>numero_do_ramal</branch>
<contingency>estado_de_contingencia</contingency>
```

Onde:

- **nome_do_agente:** nome de cadastro do agente no sistema;
- **data_hora_de_login:** data e horário de *login* do agente no **Interact**;
- **estado_do_agente:** estado do agente;
- **id_do_estado**: identificador do estado do agente;
- **nome_da_pausa**: nome da pausa cadastrada (enviado apenas se o estado do agente for uma pausa cadastrada – estado do agente será **custom_pause**);
- **gatilho_da_pausa**: identifica como a pausa foi disparada. Pode ser manual (selecionada manualmente por agente ou supervisor), *scheduled* (pausa automática configurada no sistema) ou *system* (pausa disparada pelo sistema). Este atributo somente é enviado se o agente estiver em alguma pausa;
- **timeout_da_pausa**: timestamp em que uma pausa automática está programada para se encerrar (atributo somente disponível quando o agente está em uma pausa automática – ou seja, gatilho da pausa será *scheduled*);
- **id_da_pre_pausa**: corresponde ao identificador da pausa em que o agente vai entrar após finalizar todas as chamadas em andamento.As informações da pré-pausa só serão enviadas quando o agente for colocado em pausa enquanto estiver ocupado (serviço ou particular);

- **nome_da_pre_pausa**: nome da pausa correspondente à pre-pausa;
- **gatilho_da_pre_pausa**: identifica como a pré-pausa foi disparada. Pode ser manual (selecionada manualmente por agente ou supervisor), *scheduled* (pausa automática configurada no sistema) ou *system* (pausa disparada pelo sistema). Este atributo somente é enviado se o agente estiver com pré-pausa automática;
- **data_hora_entrada_pre_pausa**: data e horário do momento em que o agente foi para pré-pausa;
- **numero_do_ramal**: número do ramal associado ao agente;
- **estado_de_contingencia**: informa se o agente está logado no modo de contingência (true) ou não (false).

O estado do agente pode ser:

- *invalid*: inválido;
- *not_logged_in*: não logado;
- *ready*: pronto (ou operando);
- *ready_free*: operando livre;
- *ready_service*: operando em serviço (com chamada de serviço);
- *ready_private*: operando em particular (com chamada particular);
- *pause*: em pausa;
- *fail_pause*: em pausa por falha;
- *no_handling_fail_pause*: em pausa por falha devido ao não atendimento da chamada (via **Interact Multiagent**);
- *timeout_fail_pause*: em pausa por falha devido ao não atendimento da chamada (via **Interact CTI**);

- *invalid_state_fail_pause*: em pausa por falha devido a não definição de estado do agente após *login*;
- *inactivity_fail_pause*: em pausa por falha devido a inatividade do agente;
- *call_rejected_fail_pause*: em pausa por falha devido a chamada ter sido rejeitada pelo agente;
- *timeout_scheduled_fail_pause*: em pausa por falha devido a *timeout* de saída de pausa automática;
- *follow_me_fail_pause*: em pausa por falha devido a programação de siga-me no ramal do agente para o ramal de outro agente;
- *custom_pause:* em pausa personalizada*;*
- *unknown*: desconhecido.

O evento **on_stage_change** inclui o atributo **state_changed_by** que informa quem realizou a última alteração de estado do agente, identificado pelo nome de *login*. Em caso de alteração provocada pelo sistema, como as mudanças de estado para pausa por falha, o atributo terá o valor **"system"**.

Os eventos **on_call_receive, on_call_accept, on_call_generate, on_call_hold, on_call_retrieve, on_call_release, on_call_consultation, on_call_end** e **on_call_conference** incluem um conjunto de dados comuns relacionados a chamadas e apresentado a seguir.

Em JSON:

```
"media_type": nome_da_midia,
"interlocutor": identificador_do_interlocutor,
"service": nome_do_servico,
"call_id": identificador_da_chamada,
"duration": duracao_da_chamada,
```

```
"call_state": estado_da_chamada,
"call_type":tipo_da_chamada,
"call_source": origem_da_chamada,
"consultation_call_id": identificador_da_chamada_de_consulta,
"entry_point": identificador_de_ramal_de_entrada,
"entry_address": identificador_de_endereco_de_entrada_email,
"associated_data": dados_associados
```

Em XML:

```
<media_type>nome_da_midia</media_type>
<interlocutor>identificador_do_interlocutor</intercolutor>
<service>nome_do_servico</service>
<call_id>identificador_da_chamada</call_id>
<duration>duracao_da_chamada</duration>
<call_state>estado_da_chamada</call_state>
<call_type>tipo_da_chamada</call_type>
<call_source>origem_da_chamada</call_source>
<consultation_call_id>identificador_da_chamada_de_consulta</consultation_call_id>
<entry_point>identificador_de_ramal_de_entrada</entry_point>
<entry_address>identificador_de_endereco_de_entrada_email</entry_address>
<associated_data>dados_associados</associated_data>
```

Onde:

- **nome_da_midia:** nome do tipo de mídia da chamada (“voice”, “email” ou “chat”);
- **identificador_do_interlocutor:** identificador do interlocutor (pode ser um número de telefone para chamadas de voz, um endereço de *e-mail* para mensagens de *e-mail* ou um nome para um cliente de *Chat*). Em conferências, esse atributo traz uma lista de interlocutores separados por vírgula, os quais podem ser nomes de agentes ou números telefônicos;
- **nome_do_servico:** nome do serviço pelo qual a chamada chegou ao agente;
- **identificador_da_chamada:** identificador único da chamada que pode ser utilizado como chave para integração com outros produtos da empresa e cruzamento com bancos de dados;
- **duração_da_chamada:** duração da chamada (em segundos);
- **estado_da_chamada**: estado da chamada, o qual pode ser:
  - *ready: pronto;*
  - *dialing:processando chamada (“discando”);*
  - *ringing: tocando;*
  - *calling: chamando;*
  - *busy: ocupado;*
  - *conference: em conferência;*
  - *held: em espera;*
  - *paused: em pausa;*
  - *after_call:pós-atendimento;*
  - *classifying*: em classificação;
  - *unavailable*: indispónível;

  - *fail*: em falha;
  - *reserved*: reservado;
  - *queued*: em fila;
  - *waiting_authorization*: aguardando autorização;
  - *authorized*: autorizada;
  - *blocked*: bloqueada;
  - *monitoring*: em monitoração;
  - *unknown*: desconhecido.
- **tipo_da_chamada:** tipo da chamada, o qual pode ser:
  - *inbound*: receptiva (entrante);
  - *outbound*: ativa (sainte);
  - *unknown*: desconhecido;
- **origem_da_chamada:** origem da chamada, a qual pode ser:
  - *internal*: interna;
  - *external*: externa;
  - *internal consultation*: interna de consulta;
  - *external consultation*: externa de consulta;
  - *internal conference*: interna de conferência;
  - *external conference*: externa de conferência;
  - *internal callback*: interna de *Callback*;
  - *external callback*: externa de *Callback*;
  - *unknown*: desconhecida.
- **identificador_da_chamada_de_consulta:** identificador da chamada em consulta (esse campo só será enviado no caso da existência de uma chamada em consulta associada).

identificador_de_endereco_de_entrada_email: endereço de entrada de *e-mails* configurado para o serviço. Esse campo apenas é enviado no caso da chamada ser de *e-mail.*

- **identificador_de_ramal_de_entrada:** ramal de entrada de chamadas de voz configurado para o serviço (ponto de roteamento). Esse campo apenas é enviado no caso ser de voz.
- **dados_associados:** campo de dados associados à chamada.

O evento **on_call_release** inclui um atributo denominado **must_classify**, que pode ter os valores **true** ou **false**. Se este atributo tiver o valor **false**, indica que a chamada não precisa ser classificada. Se o valor do atributo for **true**, indica que a chamada deve ser classificada.

Nos eventos **on_call_release**, **on_call_generate** e **on_call_receive** são incluídos ainda os atributos **classification_time**, os quais contêm, respectivamente, os tempos limite (em segundos) para a classificação e para o pós atendimento da chamada.

Nos eventos **on_call_generate** e **on_call_receive** são incluídos também os atributos **expected_handle_time** e **permissions**. O primeiro informa o tempo esperado de atendimento (em segundos). O atributo **permissions** é um *array* com um conjunto de operações que podem ser realizadas sobre a chamada. As operações possíveis são:

- **consult:** permissão para realizar consulta;
- **transfer:** permissão para realizar transferência;
- **conference:** permissão para realizar conferência;
- **email_forward:** permissão para encaminhar mensagens de *e-mail* (somente para chamadas de *e-mail*)

O evento **on_call_receive** também informa o atributo **must_accept**. Quando este atributo tiver o valor **true**, é indicativo de que o agente está configurado com atendimento manual e será necessário enviar o comando de aceite de chamada (comando

call_accept) para que a chamada seja autorizada. Se o atributo **must_accept** tiver o valor **false**, é indicativo de que o agente etá configurado com atendimento automático.

O evento **on_call_transfer** inclui informações específicas sobre uma transferência, conforme apresentado a seguir:

Em JSON:

```
"transfer_from": origem_da_transferencia,
"transfer_to": destino_da_transferencia,
"transfer_datetime": data_hora_da_transferencia,
"service": nome_do_servico,
"previous_call_id": identificador_da_chamada_anterior,
"call_id": identificador_da_chamada
```

Em XML:

```
<transfer_from>origem_da_transferencia</transfer_from>
<transfer_to>destino_da_transferencia</transfer_to>
<transfer_datetime>data_hora_da_transferencia</transfer_datetim
e>
<service>nome_do_servico</service>
<previous_call_id>identificador_da_chamada_anterior</previous_c
all_id>
<call_id>identificador_da_chamada</call_id>
```

Onde:

- **origem_da_transferencia:** nome do agente de origem da transferência;

- **destino_da_transferencia:** nome do agente de destino da transferência;
- **data_hora_da_transferencia:** data e horário da execução da transferência;
- **nome_do_servico:** nome do serviço pelo qual a chamada chegou ao agente;
- **identificador_da_chamada_anterior:** identificador da chamada anterior, estabelecida antes da transferência (identifica a chamada original);
- **identificador_da_chamada:** identificador da nova chamada estabelecida para a transferência.

O evento **on_chat_message** inclui informações sobre a ocorrência de uma nova mensagem de *Chat* . Sua forma é apresentada a seguir:

Em JSON:

```
{
   "service": nome_do_servico,
   "call_id": identificador_da_chamada,
   "actor": nome_do_originador,
   "actor_alias": nome_de_exibicao_do_originador,
   "message": texto_da_mensagem,
   "message_id": identificador_da_mensagem,
   "timestamp": timestamp_da_mensagem,
}
```

Em XML:

```
   <service> nome_do_servico </service>
   <call_id> identificador_da_chamada </call_id>
   <actor> nome_do_originador </actor>
   <actor_alias> nome_de_exibicao_do_originador </actor_alias>
   <message>texto_da_mensagem </message>
```

```
<message_id> identificador_da_mensagem </message_id>
<timestamp>timestamp_da_mensagem</timestamp>
```

Onde:

- **nome_do_servico**: nome do serviço pelo qual a chamada chegou ao agente;
- **identificador_da_chamada**: identificador da chamada de *Chat*;
  nome_do_originador: nome da origem da mensagem (agente ou interlocutor de *Chat*);
- **nome_de_exibicao_do_originador**: nome de exibição da origem da mensagem (agente ou interlocutor de *Chat*);
- **texto_da_mensagem**: texto enviado na mensagem;
- **identificador_da_mensagem:** identificador numérico sequencial único de cada mensagem em uma determinada chamada de *Chat*;
- **timestamp_da_mensagem:** timestamp de envio da mensagem.

O evento **on_typing** inclui informações sobre o estado da digitação em uma determinada chamada de *Chat.*

Em JSON:

```
{
   "call_id": identificador_da_chamada,
   "actor": ator_da_acao_de_digitacao
,
   "actor_alias": nome_de_exbicao_do_ator,
   "status": estado_da_digitacao
```

```
}
```

Em XML

```
<call_id> identificador_da_chamada </call_id>
<actor> ator_da_digitacao </actor>
<actor_alias> nome_de_exibicao_do_ator </actor_alias>
<status>estado_da_digitacao</status>
```

Onde:

- **identificador_da_chamada**: identificador número da chamada;
- **ator_da_digitacao**: nome do agente ou interlocutor que está iniciando ou finalizando a ação de digitação;
- **nome_de_exibicao_do_ator**: nome de exibição do agente ou interlocutor que está iniciando ou finalizando a ação de digitação;
- **estado_da_digitacao**: estado da digitação. Pode ser **start** para indicar o início de digitação ou **stop** para indicar o término da digitação.

O evento **on_config_change** inclui informações sobre os dispositivos disponíveis para um agentes (incluindo tipo de mídia e quantidade):

Em JSON:

```
"devices":    [   {    "media_type": nome_da_midia_1,
                       "quantity": quantidade_de_dispositivos_1,
                       "handle_mode": modo_de_atendimento
                  },
                  {    "media_type": nome_da_midia_2,
                       "quantity": quantidade_de_dispositivos_2,
```

```
                    "handle_mode": modo_de_atendimento_2
                },
                ...
                {   "media_type": nome_da_midia_n,
                    "quantity": quantidade_de_dispositivos_n,
                    "handle_mode": modo_de_atendimento_n
                }
            ]
```

Em XML:

```
<devices>
    <member>
        <media_type>nome_da_midia_1</media_type>
        <quantity>quantidade_de_dispositivos_1</quantity>
        <handle_mode>modo_de_atendimento_1</handle_mode>
    </member>
    <member>
        <media_type>nome_da_midia_2</media_type>
        <quantity>quantidade_de_dispositivos_2</quantity>
        <handle_mode>modo_de_atendimento_2</handle_mode>
    </member>
    ...
    <member>
        <media_type>nome_da_midia_n</media_type>
        <quantity>quantidade_de_dispositivos_n</quantity>
```

```
                <handle_mode>modo_de_atendimento_n</handle_mode>
            </member>
        </devices>
```

Onde:

- **nome_da_midia_X:** nome do tipo de mídia (“voice”, “email” ou “chat”);
- **quantidade_de_dispositivos_X:** quantidade de dispositivos de um determinado tipo de mídia;
- **modo_de_atendimento_X:** modo de atendimento configurado. Pode ter os valores **auto** (atendimento automático) ou **manual** (atendimento que requer aceite do agente).

O evento **on_queue_change** inclui informações sobre as filas de chamadas em serviços nas quais um agente é elegível (inclui nome do serviço e número de chamadas na fila):

Em JSON:

```
{
   "agent": nome_do_agente,
   ...
   "queues": [
               { "service": nome_servico,
                 "length": comprimento_fila,
                 "calls":   [ {    "media_type": tipo_de_midia,
                                   "call_id": identificador_da_chamada
                            }, ...
                           ]
           }, ...
```

```
        ]
}
```

Em XML:

```
   <agent>nome_do_agente</agent>
   ...
   <queues>
         <member>
               <service>nome_servico</service>
         <lenght>comprimento_fila</length>
         <calls>
               <member>
                     <media_type>tipo_de_midia</media_type>
                     <call_id>identificador_da_chamada</call_id>
               </member>
                  ...
         </calls>
</member>
   ...
</queues>
```

Onde:

- **nome_do_agente:** nome do agente;
- **nome_servico:** nome do serviço ao qual a chamada pertence;
- **comprimento_fila:** quantidade de chamadas aguardando atendimento;
- **tipo_de_midia:** nome da mídia a qual a chamada pertence;

- **identificador_da_chamada:** identificador único da chamada.

Em caso de chamada de *Chat* pela mídia Telegram, os eventos **on_call_receive** e **on_call_accept** incluem informações a respeito do bot configurado para atender chamadas de um determinado serviço.

Em JSON:

```
{
    "agent": nome_do_agente,
   ...
    "call_id": identificador_da_chamada
   ...
    "bot":
    {
      "type": tipo_do_bot,
      "data":
      {"id": identificador_do_bot,
       "chat_id": identificador_da_chamada_do_bot
      }
    }
}
```

Em JSON:

```
<agent>nome_do_agente/agent>
...
```

```
<call_id>identificador_da_chamada</call_id>
...
<bot>
   <type> tipo_do_bot </type>
   <data>
       <id> identificador_do_bot </id>
       <chat_id> identificador_da_chamada_do_bot </chat_id>
   </data>
</bot>
```

Onde:

- **tipo_do_bot:** indica qual é o tipo do bot podendo ser *telegram* ou *fb_messenger*;
- **Identificador_do_bot**: string que identifica um bot e que é associado a um serviço do **Interact**;
- **Identificador_da_chamada_do_bot**: chave informada pelo bot e que identifica uma chamada.

O evento **on_call_record** inclui informações sobre a gravação de chamadas e é informado logo ao final de um atendimento (caso a gravação não esteja configurada, o evento não é enviado).

Em JSON:

```
{
    "agent": nome_do_agente,
   ...
    "call_id": identificador_da_chamada,
```

```
    "record_id": identificador_da_gravacao,
    "entity_id": identificador_da_entidade
    "mode": modo_de_gravacao
}
```

Em XML:

```
<agent>nome_do_agente/agent>
...
<call_id>identificador_da_chamada</call_id>
<record_id>identificador_da_gravacao</record_id>
<entity_id>identificador_da_entidade</entity_id>
<mode>modo_de_gravacao</mode>
```

Onde:

- **nome_do_agente:** nome do agente;
- **identificador_da_chamada:** identificador único da chamada;
- **identificador_da_gravação**: identificador único da gravação;
- **identificador_da_entidade:** identificador único de uma entidade (agente, ramal, grupo, ou PA) associada à gravação.
- **modo_de_gravacao:** pode ser “auto” para gravações realizadas devido à programação (gravação automática) ou “on_demand” para gravações realizadas sob demanda (ver capítulo sobre Gravador).

O evento **on_call_conference** inclui o atributo **owner** que identifica o dono da conferência (agente ou ramal que iniciou a conferência e que tem o poder de incluir ou excluir participantes da mesma).

O evento **on_file_transfer** será enviado sempre que houver uma ação de transferência de arquivo em uma chamada de *Chat* em que o agente esteja envolvido e inclui os seguintes atributos:

Em JSON:

```
{
    "file_id": identificador_do_arquivo,
    "file_index": indice_do_arquivo,
    "file_name": nome_do_arquivo,
    "interlocutor": interlocutor_da_chamada,
    "actor": ator_da_acao,
    "actor_alias": nome_alternativo_do_ator,
    "session_id": identificador_da_sessao,
    "status": estado_da_transferencia,
    "timestamp": timestamp_do_evento
  }
```

Em XML:

```
  <file_id> identificador_do_arquivo </file_id>
  <file_index> indice_do_arquivo <\file_index>
  <file_name> nome_do_arquivo <\file_name>
  <interlocutor> interlocutor_da_chamada <\interlocutor>
  <actor> ator_da_acao <\actor>
  <actor_alias> nome_alternativo_do_ator <\actor_alias>
  <session_id> identificador_da_sessao <\session_id>
```

```
<status> estado_da_transferencia <\status>
```

Onde:

- **identificador_do_arquivo**: identificador único do arquivo utilizado para baixar o arquivo da plataforma Dígitro. Ele é informado quando o arquivo já estiver disponível para ser baixado (atributo status com valor available);
- **indice_do_arquivo**: valor sequencial, iniciando em 1, que identifica a transferência dentro de uma chamada específica;
- **nome_do_arquivo**: nome do arquivo que está sendo transferido;
- **interlocutor_da_chamada**: nome do interlocutor da chamada;
- **ator_da_acao**: nome de quem realizou a ação de transferência. Pode ser o nome do próprio agente que recebe o evento ou o nome do interlocutor na chamada;
- **nome_alternativo_do_ator**: nome alternativo do ator;
- **identificador_da_sessao**: identificador da sessão de transferência de arquivo. Este atributo é enviado apenas para o agente que solicitou a transferência, assim que o interlocutor aceitar a solicitação (atributo *status* com valor *accepted*);
- **estado_da_transferencia**: estado da transferência, o qual pode ser:
  - requested (transferência de arquivo solicitada);
  - accepted (transferência de arquivo aceita);
  - rejected (transferência de arquivo rejeitada);
  - canceled (transferência de arquivo cancelada);
  - available (arquivo disponível);
  - accessed (arquivo acessado).

O evento on_agent_status será enviado sempre que houver uma nova assinatura (para cada agente assinado) ou quando for aberto um canal já assinado (também para cada agente assinado). O objetivo do evento é informar o contexto do agente, com sua configuração e chamadas em andamento. Este evento inclui os seguintes dados:

Em JSON:

```
{
  "devices":
  [
    {
      “media_type”: nome_da_midia,
      "quantity": quantidade_de_dispositivos,
      "handle_mode": modo_de_atendimento
    },
    ...
    {...}
  ]
,
 "calls":
  [
    {
      "media_type": nome_da_midia,
      "interlocutor": identificador_do_interlocutor,
      "service": nome_do_servico,
      "call_id": identificador_da_chamada,
      "duration": duracao_da_chamada,
      "call_state": estado_da_chamada,
      "call_type":tipo_da_chamada,
      "call_source": origem_da_chamada,
      "consultation_call_id":identificador_da_chamada_de_consulta,
      "entry_address": identificador_de_endereco_de_entrada_email,
```

```
      "associated_data": dados_associados,
      "classification_time”: tempo_de_classificacao,
      "after_call_work_time": tempo_de_pos_atendimento,
      "expected_handle_time": tempo_esperado_de_atendimento,
      "permissions": permissoes
    },
    ...
    { ... }
  ]
}
```

Em XML:

```
<devices>
  <member>
    <media_type>nome_da_midia</media_type>
    <quantity>quantidade_de_dispositivos</quantity>
    <handle_mode>modo_de_atendimento</handle_mode>
  </member>
  ...
  <member>...</member>
</devices>
<calls>
  <member>
    <media_type>nome_da_midia</media_type>
    <interlocutor>identificador_do_interlocutor</intercolutor>
    <service>nome_do_servico</service>
    <call_id>identificador_da_chamada</call_id>
```

```
    <duration>duracao_da_chamada</duration>
    <call_state>estado_da_chamada</call_state>
    <call_type>tipo_da_chamada</call_type>
    <call_source>origem_da_chamada</call_source>
    <consultation_call_id>identificador_da_chamada_de_consulta
</consultation_call_id>
    <entry_address>identificador_de_endereco_de_entrada_email
</entry_address>
    <associated_data>dados_associados</associated_data>
    <classification_time>
tempo_de_classificacao</classification_time>
    <after_call_work_time> tempo_de_pos_atendimento
</after_call_work_time>
    <expected_handle_time> tempo_esperado_de_atendimento
</expected_handle_time>
    <permissions> permissoes </permissions>
  </member>
  ...
  <member> ... </member>
</calls>
```

Onde:

- **nome_da_midia:** nome do tipo de mídia da chamada (“voice”, “email” ou “chat”);
- **quantidade_de_dispositivos**: quantidade de dispositivos de um determinado tipo de mídia;

- **modo_de_atendimento**: modo de atendimento configurado. Pode ter os valores auto (atendimento automático) ou manual (atendimento que requer aceite do agente);
- **identificador_do_interlocutor**: identificador do interlocutor (pode ser um número de telefone para chamadas de voz, um endereço de e-mail para mensagens de e-mail ou um nome para um cliente de *Chat*);
- **nome_do_servico**: nome do serviço pelo qual a chamada chegou ao agente;
- **identificador_da_chamada**: identificador único da chamada (inclui a data de geração da chamada) e pode ser utilizado como chave para integração com outros produtos da empresa e cruzamento com bancos de dados;
- **duração_da_chamada**: duração da chamada (em segundos);
- **estado_da_chamada**: estado da chamada, o qual pode ser:
  - ready: pronto;
  - dialing:processando chamada (“discando”);
  - ringing: tocando;
  - calling: chamando;
  - busy: ocupado;
  - conference: em conferência;
  - held: em espera;
  - paused: em pausa;
  - after_call:pós-atendimento;
  - classifying: em classificação;
  - unavailable: indisponível;
  - fail: em falha;
  - reserved: reservado;
  - queued: em fila;

  - waiting_authorization: aguardando autorização;
  - authorized: autorizada;
  - blocked: bloqueada;
  - monitoring: em monitoração;
  - unknown: desconhecido.

- **tipo_da_chamada**: tipo da chamada, o qual pode ser:
  - inbound: receptiva (entrante);
  - outbound: ativa (sainte);
  - unknown: desconhecido.

- **origem_da_chamada**: origem da chamada, a qual pode ser:
  - internal: interna;
  - external: externa;
  - internal consultation: interna de consulta;
  - external consultation: externa de consulta;
  - internal conference: interna de conferência;
  - external conference: externa de conferência;
  - internal callback: interna de *Callback*;
  - external callback: externa de *Callback*;
  - unknown: desconhecida.

- **identificador_da_chamada_de_consulta**: identificador da chamada em consulta. Esse campo só será enviado no caso da existência de uma chamada em consulta associada e da mesma mídia;
- **identificador_de_endereco_de_entrada_de_email**: endereço de entrada de e-mails configurado para o serviço. Esse campo apenas é enviado no caso da chamada ser de e-mail;

- **identificador_de_ramal_de_entrada**: ramal de entrada de chamadas de voz configurado para o serviço (ponto de roteamento). Esse campo apenas é enviado no caso da chamada ser de voz;
- **dados_associados**: campo de dados associados à chamada;
- **tempo_de_classificacao**: tempo limite (em segundos) para a classificação da chamada;
- **tempo_de_pos_atendimento**: tempo limite (em segundos) o pós-atendimento da chamada;
- **tempo_esperado_de_atendimento**: duração esperada para o atendimento (em segundos);
- **permissoes**: é um array com um conjunto de operações que podem ser realizadas sobre a chamada. As operações possíveis são:
  - **consult**: permissão para realizar consulta;
  - **transfer**: permissão para realizar transferência;
  - **conference**: permissão para realizar conferência;
  - **email_forward**: permissão para encaminhar mensagens de e-mail (somente para chamadas de e-mail).

O evento **on_video_action** inclui os seguintes dados:

Em JSON:

```
{
     "call_id": identificador_da_chamada,
     "action": acao,
     "motive": motivo_de_cancelamento,
     "message": mensagem_de_cancelamento,
```

```
    "actor": ator,
    "room_id": identificador_da_sala_de_video,
    "peer_id_from": identificador_da_conexao_origem,
    "peer_id_to": identificador_da_conexao_destino,
    "url": caminho_do_video_client
}
```

Em XML:

```
    <call_id> identificador_da_chamada </call_id>
    <action> acao </acao>
    <motive> motivo_de_cancelamento </motive>
    <message> mensagem_de_cancelamento </message>
    <actor> ator </actor>
    <room_id> identificador_da_sala_de_video </room_id>
    <peer_id_from> identificador_da_conexao_origem
</peer_id_from>
    <peer_id_to> identificador_da_conexao_destino </peer_id_to>
    <url> caminho_do_video_client </url>
```

Onde:

- identificador_da_chamada: identificador único da chamada de *Chat* que pode ser utilizado como chave para integração com outros produtos da empresa e cruzamento com bancos de dados;
- acao: ação de vídeo, que pode ser:

  - **start:** sessão de vídeo iniciada em uma chamada de *Chat*;
  - **accept:** sessão de vídeo foi aceita/estabelecida pelo interlocutor em uma chamada de *Chat*;
  - **stop:** sessão de vídeo encerrada em uma chamada de *Chat*;
  - **canceled:** sessão de vídeo foi cancelada (encerramento antes da sessão de vídeo ser estabelecida).
- motivo_de_cancelamento: atributo enviado somente se o atributo *acao* for *canceled* e informa o motivo do cancelamento reportado pelo serviço de vídeo;
- mensagem_de_cancelamento: atributo enviado somente se o atributo *acao* for *canceled* e informa a mensagem de cancelamento reportada pelo serviço de vídeo;
- ator: nome do agente ou interlocutor de *Chat* que originou a ação;
- identificador_da_sala_de_video: identificador único da sessão de vídeo no formato UUID;
- identificador_da_conexao_origem: identificador único da conexão origem (agente) no formato UUID;
- identificador_da_conexao_destino: identificador único da conexão destino (interlocutor de *Chat*) no formato UUID;
- caminho_do_video_client: caminho (URL) para abrir janela do cliente de vídeo em um navegador.

–

O evento **on_inactivity_timer** inclui os seguintes dados:

Em JSON:

```
{
     "call_id": identificador_da_chamada,
     "timer": estado_do_timer,
     "timeout": tempo_do_timer
    }
```

Em XML:

```
     <call_id> identificador_da_chamada </call_id>
     <timer> estado_do_timer </timer>
     <timeout> tempo_do_timer </timeout>
```

Onde:

- identificador_da_chamada: identificador único da chamada de *Chat* que pode ser utilizado como chave para integração com outros produtos da empresa e cruzamento com bancos de dados;
- estado_do_timer: será **on** quando o timer de inatividade for ligado e **off** quando for desligado;
- tempo_do_timer: tempo em segundos de duração do timer de inatividade (após decorrido este tempo, se não houver atividade do agente na chamada de *Chat* correspondente, ela será derrubada).

4

EVENTOS DE CONTROLE

Eventos de controle são mensagens que o **Interact CTI** envia para sinalizar mudança de estado de canais, disponibilidade do servidor **Interact** e expiração de assinatura.

Evento	Descrição
on_open_channel	Ocorre na abertura de um canal.
on_close_channel	Ocorre no fechamento de um canal.
on_subscription_expire	Ocorre na expiração de uma assinatura.
on_subscription_expire_warning	Ocorre 5 minutos antes de expirar uma assinatura.
on_server_state_change	Ocorre na mudança de estado da conexão com o **Interact**.

A seguir são apresentados os formatos de cada um desses eventos:

O evento **on_open_channel** sinaliza que um canal foi aberto. Esse é o primeiro evento que circulará pelo canal e tem a função de certificar o cliente da abertura bem sucedida do canal. Será enviado apenas ao canal recém-aberto. A seguir é apresentada a forma de envio desse evento:

Em JSON:

```
{ "channel_id": identificador_do_canal }
```

Em XML:

```
<channel_id>identificador_do_canal</channel_id>
```

Onde:

- **identificador_do_canal:** número que identifica o canal aberto.

O evento **on_close_channel** sinaliza que um canal foi fechado pelo **Interact CTI**. Isso pode ocorrer como consequência de um comando **close_channel** ou, ainda, caso o canal permaneça por mais de 1 minuto sem nenhuma assinatura associada. Em um cenário típico, em que há uma única assinatura por canal, o canal será fechado 1 minuto após a expiração dessa assinatura. Será enviado apenas ao canal recém-fechado.

A seguir é apresentada a forma de envio desse evento.

Em JSON:

{ "channel_id": identificador_do_canal }

Em XML:

```
<channel_id>identificador_do_canal</channel_id>
```

Onde:

- **identificador_do_canal:** número que identifica o canal aberto.

O evento **on_susbcribe_expire** sinaliza que uma assinatura expirou. Esse evento será enviado apenas pelo canal associado à assinatura e tem o seguinte formato:

Em JSON:

```
{ "id": identificador_da_assinatura,
  "channel_id": identificador_do_canal
}
```

Em XML:

```
<id>identificador_da_assinatura</id>
<channel_id>identificador_do_canal</channel_id>
```

Onde:

- **identificador_da_assinatura:** identifica a assinatura que expirou;
- **identificador_do_canal:** número que identifica o canal aberto.

O evento **on_susbcribe_expire_warning** ocorre 5 minutos antes da expiração da assinatura. Esse evento será enviado apenas pelo canal associado à assinatura e tem o seguinte formato:

Em JSON:

```
{ "id": identificador_da_assinatura,
  "channel_id": identificador_do_canal
```

```
}
```

Em XML:

```
<id>identificador_da_assinatura</id>
<channel_id>identificador_do_canal</channel_id>
```

Onde:

- **identificador_da_assinatura:** identifica a assinatura que expirou;
- **identificador_do_canal:** número que identifica o canal aberto.

O evento **on_server_state_change** sinaliza que houve mudança do estado do servidor Interact. Isso pode ocorrer como consequência de queda do processo **Interact** ou queda da conexão entre o processo **Interact** e o **Interact CTI** e, também, no restabelecimento da conexão entre o processo **Interact** e o **Interact CTI**. Esse evento é enviado a **todos os canais abertos**.

Em JSON:

```
{ "state": estado_do_servidor }
```

Em XML:

```
<state>estado_do_servidor</state>
```

Onde:

- **estado_do_servidor:** indica o estado da conexão entre o **Interact CTI** e o processo **Interact**, podendo assumir os valores *connected* ou *disconnected*.

5

CONSULTA E MONITORAÇÃO DE ITENS CADASTRADOS

A consulta de itens cadastrados está disponível para os recursos agent e supervisor, com os seguintes comandos:

- get_agents_list;
- get_services_list;
- get_pause_type_list;
- get_classification_list

A consulta de um mesmo tipo de item cadastrado não pode ser feita em um intervalo de menos de 5 segundos. Caso este tempo não seja respeitado, o comando retorna um erro:

RESPONSE DATA em JSON:

```
{ "response":
            { "command": nome_do_comando,
```

```
                "status": "error",
                "error_type": "unsafe_request"
            }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
      <command>nome_do_comando</command>
      <status>error</status>
      <error_type>unsafe_request</error_type>
</response>
```

AGENTES CADASTRADOS (COMANDO GET_AGENTS_LIST)

Este comando retorna as listas de agentes cadastrados no **Interact**.

Método HTTP GET

interact_cti/v1.0/recurso/get_agents_list?format=[xml | json]&complete=[true|false]

Onde **recurso** pode ser **agent** ou **supervisor**.

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

O parâmetro **complete** é opcional e permite definir entre dois possíveis formatos de resposta. Caso esse parâmetro não seja enviado, a resposta utilizará o valor *default* de **complete** que é **false**.

Resposta ao Comando get_agents_list:

A resposta para o comando de lista de agentes será conforme os modelos a seguir:

RESPONSE DATA em JSON com complete false:

```
{ "response":
            { "command": "get_agents_list",
              "status": "ok",
              "logged_in_agents": [ nome_agente_logado_1,
                                    nome_agente_logado_2,
                                    ...
                                    nome_agente_logado_n
                                  ],
              "logged_out_agents": [ nome_agente_nao_logado_1,
                                    nome_agente_nao_logado_2,
                                    ...
                                    nome_agente_nao_logado_n
                                   ]
            }
}
```

RESPONSE DATA em XML com complete false:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <command> get_agents_list </command>
  <status> ok </status>
  <logged_in_agents>
      <member> nome_agente_logado_1 </member>
      <member> nome_agente_logado_2 </member>
      ...
      <member> nome_agente_logado_n </member>
  </logged_in_agents>
  <logged_out_agents>
      <member> nome_agente_nao_logado_1 </member>
      <member> nome_agente_nao_logado_2 </member>
      ...
      <member> nome_agente_nao_logado_n </member>
  </logged_out_agents>
</response>
```

RESPONSE DATA em JSON com complete true:

```
{ "response":
  { "command": "get_agents_list",
    "status": "ok",
```

```
    "agents": [ { "name": login_do_agente,
                  "dt_login": data_de_login_do_agente,
                  "state": estado_do_agente,
                  "state_id": id_do_estado_do_agente
                  "branch": ramal_do_agente,
                  "contingency": estado_de_contingencia
                },
                ...
                { ...
                }
              ]
  }
}
```

RESPONSE DATA em XML com complete true:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <command> get_agents_list </command>
  <status> ok </status>
  <agents>
    <member>
      <name> login_do_agente </name>
      <dt_login> data_de_login_do_agente </dt_login>
      <state> estado_do_agente </state>
```

```
      <state_id> id_do_estado_do_agente </state_id>
      <branch> ramal_do_agente </branch>
      <contingency> estado_de_contingencia </contingency>
    </member>
    ...
    <member>
    ...
    </member>
  </agents>
</response>
```

Onde:

- login_do_agente: nome de *login* do agente;
- data_de_login_do_agente: data de *login* do agente (este atributo não é informado se o agente não estiver logado);
- estado_do_agente: nome do estado do agente (ver lista de estados em Eventos Operacionais de Agentes e Chamadas;
- id_do_estado_do_agente: identificador do estado do agente;
- ramal_do_agente: ramal do agente (este atributo não é informado se o agente não estiver logado ou se não tiver mídia de voz configurada);
- estado_de_contingencia: informa se o agente está logado no modo de contingência (true) ou não (false).

SERVIÇOS CADASTRADOS (COMANDO GET_SERVICES_LIST)

Esse comando retorna as listas de serviços do **Interact**.

Método HTTP GET

interact_cti/v1.0/recurso/get_services_list?format=[xml | json]&complete=[true|false]

Onde recurso pode ser **agent** ou **supervisor.**

O parâmetro **format** é vi XML.

O parâmetro complete é opcional e permite definir entre dois possíveis formatos de resposta. Caso esse parâmetro não seja enviado, a resposta utilizará o valor *default* de complete que é false.

Resposta ao Comando get_services_list

A resposta para o comando de lista de serviços será conforme os modelos a seguir:

Response data em JSON com complete false:

```
{ "response":
            { "command": "get_services_list",
              "status": "ok",
              "inbound_services": [ nome_servico_receptivo_1,
```

```
                                        nome_servico_receptivo_2,
                                        ...
                                        nome_servico_receptivo_n
                                      ],
                "outbound_services": [  nome_servico_ativo_1,
                                        nome_servico_ativo_2,
                                        ...
                                        nome_servico_ativo_n
            }
}
```

Response data em XML com complete false:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<command>get_services_list</command>
   <status>ok</status>
   <inbound_services>
      <member> nome_servico_receptivo_1 </member>
      <member> nome_servico_receptivo_2 </member>
      ...
      <member> nome_servico_receptivo_n </member>
  </inbound_service s>
  <outbound_services>
     <member> nome_servico_ativo_1 </member>
     <member> nome_servico_ativo_2 </member>
```

```
    ...
    <member> nome_servico_ativo_n </member>
  </outbound_services>
</response>
```

Response data em JSON com complete true

```
{
  "response":
  {
    "command": "get_services_list",
    "status": "ok",
    "services": [ {
                    "name": nome_do_servico,
                    "type": tipo_do_servico,
                    "media_types": lista_de_midias,
                    "state": estado_do_servico,
                    "operation_begin": timestamp_inicio_operacao,
                    "operation_end": timestamp_termino_operacao
                  } ,
                  ...
                  {
                    ...
                  }
  }
}
```

Response data em XML com complete true:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <command>get_services_list</command>
  <status>ok</status>
  <services>
    <member>
      <name> nome_do_servico </name>
      <type> tipo_do_servico </type>
      <media_types> lista_de_midias </media_types>
      <state> estado_do_servico </state>
      <operation_begin> timestamp_inicio_operacao
</operation_begin>
      <operation_end> timestamp_termino_operacao </operation_end>
     </member>
     ...
     <member>
      ...
         </member>
</response>
```

Onde:

- **nome_do_servico:** nome do serviço;
- **tipo_do_servico:** tipo do serviço. Para serviços receptivos, esse campo terá o valor inbound. Para serviços ativos, este campo poderá ter um dos

seguintes valores: ready, preview, predictive, power ou validation (teste de lote);

- **lista_de_midias:** lista com as mídias habilitadas no serviço. Pode conter os valores “voice”, “email” ou “chat”;
- **estado_do_servico:** estado do serviço. Pode conter os valores on_operation (serviço em operação/expediente) ou out_of_operation (serviço fora de operação/de expediente);
- **timestamp_inicio_operacao:** data configurada de início da operação do serviço em formato UNIX timestamp. Se a data de início não estiver configurada ,o valor deste atributo será 0 (zero);
- **timestamp_termino_operacao:** data configurada de término da operação do serviço em formato UNIX timestamp. Se a data de término não estiver configurada, o valor deste atributo será 0 (zero).

MOTIVOS DE PAUSA CADASTRADOS (COMANDO GET_PAUSE_TYPE_LIST)

Esse comando retorna a lista dos motivos de pausa cadastrados no **Interact**.

Método HTTP GET

interact_cti/v1.0/recurso/get_pause_type_list?format=[xml | json]&trigger=[manual|scheduled]

Onde recurso pode ser **agent** ou **supervisor.**

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

O parâmetro **trigger** também é opcional e permite definir se serão apresentados os motivos de pausa manual (*manual*) ou automática (*scheduled*). Caso esse parâmetro não seja enviado, a resposta retornará os motivos de pausa manual.

Resposta ao Comando get_pause_type_list para motivos de pausa manual

A resposta para o comando de lista de pausas manuais será conforme os modelos a seguir:

RESPONSE DATA em JSON:

```
{ "response":
  { "command": "get_pause_type_list",
    "status": "ok",
    “trigger”: “manual”,
    "pauses":
    [
      {
         "pause": nome_da_pausa,
         "id": id_da_pausa,
         "total": tempo_total,
         "partial": tempo_parcial
       }
   ...
```

```
    ]
  }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
      <command>get_pause_type_list</command>
      <status>ok</status>
      <trigger>manual</trigger>
      <pauses>
            <member>
                  <pause>nome_da_pausa</pause>
                  <id>id_da_pausa</id>
                  <total>tempo_total</total>
                  <partial>tempo_parcial</partial>
            </member>
          ...
      </pauses>
</response>
```

Resposta ao Comando get_pause_type_list para motivos de pausa automática

A resposta para o comando de lista de pausas automáticas será conforme os modelos a seguir:

RESPONSE DATA em JSON:

```
{
        "response": {
              "command": "get_pause_type_list",
              "status": "ok",
              "trigger": "scheduled",
              "pauses": [{
                          "pause": nome_da_pausa,
                          "id": id_da_pausa,
                          "duration": duracao_da_pausa
                    }
                    ...
              ]
        }
   }
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
     <command>get_pause_type_list</command>
     <status>ok</status>
     <trigger>scheduled</trigger>
     <pauses>
     <member>
          <pause>nome_da_pausa</pause>
          <id>id_da_pausa</id>
          <duration>duracao_da_pausa</duration>
```

```
      </member>
     ...
     </pauses>
</response>
```

Onde:

- **nome_da_pausa:** nome do motivo de pausa cadastrado no **Interact**;
- **id_da_pausa:** identificador único do motivo de pausa;
- **tempo_parcial**: tempo máximo por entrada que o agente terá para utilizar esta pausa (em minutos). Ao ultrapassar esse tempo o agente é colocado em “pausa por falha”. O valor zero significa que não há limite de tempo.
- **tempo_total:** tempo máximo em que o agente pode permanecer nesse motivo de pausa por turno de trabalho. Ao ultrapassar esse tempo o agente é colocado em "pausa por falha". O valor zero significa que não há limite de tempo.
- **duracao_da_pausa**: tempo (em minutos) cadastrado para a duração da pausa automática.

CLASSIFICAÇÕES CADASTRADAS (COMANDO GET_CLASSIFICATION_LIST)

Esse comando retorna a lista de classificações dos serviços no **Interact**.

Método HTTP GET

interact_cti/v1.0/recurso/get_classification_list?format=[xml | json]

Onde recurso pode ser **agent** ou **supervisor**.

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

Resposta ao Comando get_classification_list

A resposta para o comando de lista de serviços será conforme os modelos a seguir:

RESPONSE DATA em JSON:

```
{
    "response": {
        "command": "get_classification_list",
        "status": "ok",
        "classification": [ lista_de_classificacoes ]
   }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <command>get_classification_list</command>
   <status>ok</status>
   <classification>lista_de_classificacoes</classification>
</response>
```

Onde:

- **lista_de_classificacoes:** lista de classificações cadastradas no **Interact**, sendo que cada uma contém os seguintes campos:
  - **id**: identificador da classificação;
  - **name**: nome da classificação;
  - **scope**: escopo da classificação, que pode ser general (classificação aplicável a todas as chamadas de serviço), inbound (classificação aplicável a todas as chamadas de serviços receptivos), outbound (classificação aplicável a todas as chamadas de serviços ativos) ou, ainda, pode ser o nome do serviço específico ao qual a classificação se aplica.

MONITORAÇÃO DE ITENS CADASTRADOS

Os serviços disponibilizados pelo **Interact CTI** para monitoração de itens cadastrados possibilitam que as listas de agentes, serviços, motivos de pausas e classificações sejam monitorados para que uma aplicação possa ser notificada quando houver alteração de algum item.

As funcionalidades deste recurso dependem de licença específica.

Os comandos têm o seguinte formato geral:

interact_cti/v1.0/config/monitor/metodo?parametros[&format= (json|xml)]

Onde:

- **metodo**: nome da requisição (comando);
- **parametros**: conjunto de pares nome-valor na forma prm=valor, separados por & (E comercial – ampersand);

- **format**: define o formato desejado para a resposta, que pode ser no formato JSON ou no formato XML (*default*). Em comandos POST e na ausência do parâmetro **format**, será utilizado o mesmo formato informado no header Content-type.

As requisições aos serviços serão sempre por meio de métodos HTTP GET ou POST. O método GET é utilizado em todas as requisições que não enviam apenas parâmetros. O método POST é utilizado em todas as requisições que enviam pacotes de dados (além dos parâmetros). Os parâmetros enviados na URL deverão ser codificados no padrão URL encode. Os dados nos pacotes POST deverão ser codificados em UTF-8. O acesso se dará na porta específica 8582 conforme exemplos a seguir:

Comando de assinatura :

http://XXX.XXX.XXX.XXX:8582/interact_cti/v1.0/config/monitor/subscribe

Comando de cancelamento de assinatura:

http://XXX.XXX.XXX.XXX:8582/interact_cti/v1.0/config/monitor/unsubscribe

Comando de abertura de canal

http://XXX.XXX.XXX.XXX:8582/interact_cti/v1.0/config/monitor/open_channel

Comando de fechamento de canal

http://XXX.XXX.XXX.XXX:8582/interact_cti/v1.0/config/monitor/close_channel

IMPORTANTE

Caso haja algum erro sintático será retornado o erro HTTP 400 Bad Request. Caso o analisador sintático do **Interact CTI** consiga identificar a posição em que há um erro de

sintaxe, será enviado um texto indicando a posição do pacote enviado em que foi detectado o erro (tanto para XML como para JSON).

Caso o **Interact CTI** não esteja conectado ao processo **Interact**, todos os comandos serão recusados, retornando o erro **HTTP 503 Service Unavailable**.

Caso haja um erro no endereçamento do caminho da URL, será retornado um erro **HTTP 404 Not Found**. Isso ocorre, por exemplo, ao se realizar uma chamada com um erro no nome do recurso **config**, como a seguir:

interact_cti/v1.0/cofig/monitor/unsubscribe?id=1&format=xml

Nesse caso, sinalizando que o recurso endereçado (**cofig**, neste caso) não foi encontrado.

ASSINATURA DE MONITORAÇÃO DE CONFIGURAÇÃO (COMANDO SUBSCRIBE)

O comando subscribe do recurso config possibilita que mudanças nas listas de itens cadastrados sejam monitorados por meio de uma ou mais assinaturas. O conceito de assinatura existe para que uma aplicação possa receber apenas os eventos que necessitar.

Método HTTP POST

interact_cti/v1.0/config/monitor/subscribe?format=[xml | json]

POST DATA em JSON:

```
{
    "subscription":
    {
      "id": identificador_da_assinatura,
      "items": lista_de_itens,
      "channel_id": identificador_do_canal,
      "webhook": url_do_cliente,
      "expires": tempo_de_expiracao
    }
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
  <subscription>
    <id> identificador_da_assinatura </id>
    <items> lista_de_itens </items>
    <channel_id> identificador_do_canal </channel_id>
    <webhook> url_do_cliente </webhook>
    <expires> tempo_de_expiracao </expires>
  </subscription>
</request>
```

Onde:

- **identificador_da_assinatura**: número que identifica uma assinatura. Deve ser
- enviado apenas caso se deseje modificar uma assinatura já existente;
- **identificador_de_canal**: identificador do canal de eventos pelo qual os eventos dessa assinatura deverão ser enviados. Os canais podem ter identificadores dentro da faixa numérica desde 1 até 9999;
- **url_do_cliente**: endereço do webservice do cliente que espera receber eventos por HTTP POST.
- **tempo_de_expiracao**: tempo em segundos durante o qual a assinatura permanecerá ativa. O valor máximo é 86400, o que corresponde a um dia completo. Esse parâmetro é obrigatório. Valores aceitos entre 1 e 86400;
- **lista_de_itens**: lista de itens que serão monitorados (detalhes nos exemplos a seguir).

A lista de itens poderá ter os seguintes elementos, identificados pelo **target** (em parênteses):

- lista de serviços (**services_list**);
- lista de agentes (**agents_list**);
- lista de motivos de pausa manuais (**pause_type_list**);
- lista de classificações (**classification_list**).

Em cada item assinado deve ser informado um modo de notificação (**notification_mode**), o qual pode ser ou completo (**“complete”)** ou de alterações (**“changes”**). No primeiro caso (“complete”), sempre que alguma alteração for identificado em uma determinada lista, a lista completa será enviada no evento de

notificação de alteração. No segundo caso (“changes”), apenas os dados do item cadastrado que sofreu alterações é enviado no evento.

Existem casos em que a lista completa é enviada (independente do modo de notificação):

- quando um canal está aberto é enviado um comando de assinatura (subscribe) para aquele canal;
- quando é aberto um canal cujo identificador já está configurado em uma assinatura;
- quando são incluídos ou excluídos itens da lista.

Para cada uma das listas existe um conjunto de filtros de monitoração diferentes, apresentados e exemplificados a seguir

Lista de Serviços

O formato de um item de monitoração para lista de serviços é:

POST DATA em JSON:

```
{
  "target": "services_list",
  "types": lista_de_tipos,
  "states": lista_de_estados,
  "media_types": lista_de_midias,
  "operation_date": data_de_operacao,
  "notification_mode": modo_de_notificacao
}
```

POST DATA em XML:

```
<member>
  <target> services_list </target>
  <types> lista_de_tipos </types>
  <states> lista_de_estados </types>
  <media_types> lista_de_midias </media_types>
  <operation_date> data_de_operacao </operation_date>
  <notification_mode> modo_de_notificacao </notification_mode>
</member>
```

Onde:

- **lista_de_tipos (opcional)**: lista com os tipos de serviço que se deseja monitorar. Pode conter os valores “inbound” (serviço receptivo) ou “outbound” (serviço ativo). Na ausência deste parâmetro, os serviços serão monitorados sem filtro de tipo;
- **lista_de_estados (opcional)**: lista com os estados de serviço que se deseja monitorar. Pode conter os valores “on_operation” (serviço em operação/expediente) ou “out_of_operation” (serviço fora de operação/de expediente). Na ausência deste parâmetro, os serviços serão monitorados sem filtro de estado;
- **lista_de_midias (opcional)**: lista com as mídias habilitadas dos serviços que se deseja monitorar. Pode conter os valores “voice”, “email” ou “chat”. Na ausência deste parâmetro, os serviços serão monitorados sem filtro de mídia;
- **data_de_operacao (opcional)**: data de referência em formato UNIX timestamp para filtrar serviços que estejam dentro do período de operação nesta data. Apenas serviços que possuem data de início ou término configurados são filtrados por este parâmetro. Caso esse parâmetro não

seja informado, os serviços serão monitorados sem filtro de data de operação;

- **modo_de_notificacao**: este parâmetro é obrigatório e pode ter dois valores: “complete” ou “changes”.

Exemplo de assinatura para monitorar lista de serviços:

POST DATA em JSON:

```
{
  "subscription":
  {
    "items": [
               {
                 "target": "services_list",
                 "types": [ "inbound" ],
                 "states": ["on_operation"],
                 "media_types": [ "voice","chat" ],
                 "operation_date": 1438105815 ,
                 "notification_mode": "changes"
               }
             ],
        "channel_id": 25,
        "expires": 36000
    }
}
```

No comando acima está sendo assinada a monitoração de lista de serviços ("target": "services_list"), para serviços receptivos ("types": ["inbound"]) que estejam em expediente ("states": ["on_operation"]), com as mídias Voz e *Chat* habilitadas ("media_types": ["voice","chat"]) e dentro do período de operação na data de 28/07/2015 17:50:15 UTC ("operation_date": 1438105815). Apenas as alterações na lista de serviços serão informadas ("notification_mode": "changes").

Os eventos serão enviados para o canal com identificador 25 e a assinatura expira em 10 horas (36.000 segundos).

Lista de Agentes

O formato de um item de monitoração para lista de agentes é:

POST DATA em JSON:

```
{
  "target": "agents_list",
  "states": lista_de_estados,
  "notification_mode": modo_de_notificacao
}
```

POST DATA em XML:

```
<member>
  <target> agents_list </target>
  <states> lista_de_estados </types>
  <notification_mode> modo_de_notificacao </notification_mode>
</member>
```

Onde:

- **lista_de_estados (opcional)**: lista com os tipos de estado de agente que se deseja monitorar. Pode conter os valores “not_logged_in” (não logado) ou “logged_in” (logado). Na ausência deste parâmetro, os agentes serão monitorados sem filtro de estado.

Exemplo de assinatura para monitorar lista de agentes:

POST DATA em JSON:

```
{
  "subscription":
  {
    "items": [
                {
                  "target": "agents_list",
                  "states": ["logged_in"],
                  "notification_mode": "complete"
                }
              ],
        "channel_id": 25,
        "expires": 36000
    }
}
```

No comando acima está sendo assinada a monitoração de lista de agents ("target": "agents_list") que estejam logados ("states": ["logged_in"]). Qualquer alteração na lista de agentes provocará o envio da lista completa ("notification_mode": "complete").

Lista de Classificações

O formato de um item de monitoração para lista de classificações é:

POST DATA em JSON:

```
{
  "target": "classification_list",
  "types": lista_de_estados,
  "notification_mode": modo_de_notificacao
}
```

POST DATA em XML:

```
<member>
  <target> classification_list </target>
  <types> lista_de_estados </types>
  <notification_mode> modo_de_notificacao </notification_mode>
</member>
```

Onde:

- **lista_de_tipos (opcional)**: lista com os tipos de classificação que se deseja monitorar. Pode conter os valores “inbound” (classificações para serviço receptivo) ou “outbound” (classificações para serviço ativo). Na ausência deste parâmetro, as classificações serão monitoradas sem filtro de tipo;

Exemplo de assinatura para monitorar lista de classificações:

POST DATA em JSON:

```
{
  "subscription":
  {
    "items": [
                {
                  "target": "classification_list",
                  "types": ["outbound"],
                  "notification_mode": "complete"
                }
      ],
        "channel_id": 25,
        "expires": 36000
    }
}
```

No comando acima está sendo assinada a monitoração de lista de classificações ("target": "classification_list") associadas a serviçoa ativos ("states": ["outbound"]). Qualquer alteração na lista de classificações provocará o envio da lista completa ("notification_mode": "complete").

Lista de Pausas

O formato de um item de monitoração para lista de pausas é:

POST DATA em JSON:

```
{
  "target": "pause_type_list",
  "notification_mode": modo_de_notificacao
}
```

POST DATA em XML:

```
<member>
  <target> pause_type_list </target>
  <notification_mode> modo_de_notificacao </notification_mode>
</member>
```

Exemplo de assinatura para monitorar lista de pausas:

POST DATA em JSON:

```
{
  "subscription":
  {
    "items": [
                {
                  "target": "pause_type_list",
                  "notification_mode": "changes"
                }
              ],
        "channel_id": 25,
```

```
        "expires": 36000
    }
}
```

No comando acima está sendo assinada a monitoração de lista de pausas ("target": "pause_type_list"). associadas a serviçoa ativos ("states": ["outbound"]). Apenas as alterações na lista de pausas serão informadas ("notification_mode": "changes").

Resposta ao Comando de Assinatura

A resposta bem sucedida para a assinatura possui o seguinte formato:

RESPONSE DATA em JSON:

```
{
"response":
{
"resource":"config\/monitor",
"command":"subscribe",
"id": identificador_da_assinatura,
"status":"ok"
}
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<response>
       <resource> config\/monitor </resource>
       <command> subscribe </command>
       <id> identificador_da_assinatura </id>
       <status> ok </status>
</response>
```

Onde:

- **identificador_da_assinatura**: identificador numérico único para a assinatura informado pelo **Interact CTI**;

No caso da assinatura não ser bem sucedida, o padrão de resposta será:

RESPONSE DATA em JSON:

```
{
    "response":
    {
        "resource": "config\/monitor",
        "command": "subscribe",
        "status": "error",
        "error_type": tipo_de_erro
        "error_itens": lista_de_itens com erro
    }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
       <resource> config\/monitor </resource>
       <command> subscribe </command>
      <status> error </status>
      <error_type> tipo_de_erro </error_type>
      <error_itens> lista_de_itens_com_erro </error_itens>
</response>
```

Onde:

- **tipo_de_erro**: é um descritivo do tipo de erro (ver capítulo sobre Padrão de Respostas);
- **lista_de_itens_com_erro**: enviado apenas quando o **status** for **error** e o **tipo_de_erro** for **invalid_item**. O conteúdo de **error_items** será uma lista de itens que apresentaram erro na assinatura.

ALTERAÇÃO E RENOVAÇÃO DE ASSINATURA (COMANDO SUBSCRIBE)

Uma assinatura pode ser alterada com o envio de um novo comando de assinatura passando o código identificador da assinatura (**id**). A assinatura anterior será cancelada e a nova ficará ativa em seu lugar.

Para renovação da assinatura, basta que sejam enviados os mesmos parâmetros da assinatura original adicionados do código identificador da assinatura (**id**).

O tempo de expiração será reiniciado para o valor que for informado no campo **expires**. Todos os parâmetros da assinatura podem ser alterados, inclusive o número do canal para envio de eventos.

CANCELAMENTO DE ASSINATURA (COMANDO UNSUBSCRIBE)

Uma assinatura poderá ser cancelada ao se enviar um comando de **unsubscribe** informando-se o código identificador da assinatura (**id**). Havendo uma assinatura com o **id** informado, ela será cancelada. Caso o **id** não exista, será retornado um erro **not found**.

Método HTTP POST

interact_cti/v1.0/config/monitor/unsubscribe?format=[xml | json]

POST DATA em JSON:

```
{
  "id": identificador_da_assinatura
}
```

POST DATA em XML:

```
<request>
   <id>identificador_da_assinatura</id>
</request>
```

CONSULTA DE ASSINATURAS (COMANDO GET_SUBSCRIPTIONS)

As assinaturas ativas do recurso **config** podem ser consultadas ao se enviar um comando **get_subscriptions**. Esse comando tem dois parâmetros opcionais, um para identificar uma assinatura (**id**) e outro para identificar um canal (**channel_id**).

Informando-se o código identificador da assinatura (**id**), serão retornados os dados da respectiva assinatura. Informando-se o código identificador do canal (**channel_id**), serão retornados os dados das assinaturas existentes para aquele canal. Caso o **id** ou o **channel_id** informados não encontrem correspondências nas assinaturas ativas, será retornado um erro *not found*.

Método HTTP GET

interact_cti/v1.0/config/monitor/get_subscriptions?id=nnnn&channel_id=kkkk&format=[xml | json]

O campo **nnnn** deve ser substituído pelo código identificador da assinatura (**id**) que se deseja consultar.

O campo **kkkk** acima deve ser substituído pelo código identificador do canal (**channel_id**) configurado para a(s) assinatura(s) que se deseja consultar.

Resposta ao Comando de Consulta de Assinatura

A resposta para a consulta bem sucedida de assinaturas será uma das formas a seguir:

RESPONSE DATA em JSON:

```
{
  "response":
  {
    "resource": "config\/monitor",
    "command": "get_subscriptions",
    "status": status,
    "subscriptions": lista_de_assinaturas
  }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <command>get_subscriptions</command>
  <status>status</status>
  <subscriptions>lista_de_assinaturas</subscriptions>
</response>
```

Onde:

- **status**: estado da execução do comando (pode ser **ok** em caso de sucesso, ou **error** em caso de erro);
- **lista_de_assinaturas**: um conjunto de dados que atendem aos parâmetros de consulta e têm o seguinte formato:

Formato em JSON:

```
[
  { dados_assinatura_1 } ,
  { dados_assinatura_2 } ,
  …
  { dados_assinatura_n }
]
```

Formato em XML:

```
<member>dados_assinatura_1</member>
<member>dados_assinatura_2</member>
...
<member>dados_assinatura_n</member>
```

Onde **dados_assinatura_x** possui a seguinte estrutura:

Em JSON:

```
{
  "id":identificador_da_assinatura,
  "channel_id": identificador_do_canal,
  "resource":"config",
  "expires": tempo_de_expiracao_restante,
  "items":
  [
```

```
    {
      "target":"agents_list",
      "notification_mode": modo_de_notificacao,
      "states": lista_de_estados_de_agente
    },
    {
      "target":"services_list",
      "notification_mode": modo_de_notificacao,
      "operation_date": data_de_operacao,
      "types": lista_de_tipos_de_servico,
      "states": lista_estados_de_servico
      "media_types": lista_de_midias
    },
    {
      "target":"classification_list",
      "notification_mode": modo_de_notificacao,
      "types": lista_de_tipos_de_classificacao
    },
    {
      "target":"pause_type_list",
      "notification_mode": modo_de_notificacao
    }
  ]
}
```

Em XML:

```
<id> identificador_da_assinatura </id>
<channel_id> identificador_do_canal </channel_id>
<resource> config </resource>
<expires> tempo_de_expiracao_restante </expires>
<items>
  <member>
    <target> agents_list </target>
    <notification_mode> modo_de_notificacao </notification_mode>
    <states> lista_de_estados_de_agente </states>
  </member>
  <member>
    <target> services_list </target>
<notification_mode> modo_de_notificacao </notification_mode>
    <operation_date> data_de_operacao </operation_date>
    <types> lista_de_tipos_de_servico </types>
    <states> lista_estados_de_servico </states>
    <media_types> lista_de_midias </media_types>
  </member>
  <member>
    <target> classification_list </target>
    <notification_mode> modo_de_notificacao </notification_mode>
    <types> lista_de_tipos_de_classificacao </types>
  </member>
  <member>
```

```
    <target> pause_type_list </target>
    <notification_mode> modo_de_notificacao </notification_mode>
  </member>
</items>
```

Onde:

- **identificador_da_assinatura**: número que identifica uma assinatura;
- **identificador_de_canal**: identificador do canal de eventos pelo qual os eventos dessa assinatura são ser enviados;
- **tempo_de_expiracao_restante**: tempo em segundos restantes para a assinatura ;
- **modo_de_notificacao**: indica qual o modo de notificação do item: “complete” ou “changes”;
- **lista_de_estados_de_agente**: lista com os tipos de estado de agente da assinatura. Pode conter os valores “not_logged_in” (não logado) ou “logged_in” (logado). Na ausência deste parâmetro, os agentes serão monitorados sem filtro de estado;
- **data_de_operacao**: data de referência em formato UNIX timestamp para filtrar serviços que estejam dentro do período de operação nesta data;
- **lista_de_tipos_de_servico**: lista com os tipos de serviço da assinatura. Pode conter os valores “inbound” (serviço receptivo) ou “outbound” (serviço ativo). Na ausência deste parâmetro, os serviços serão monitorados sem filtro de tipo;
- **lista_de_estados_de_servico**: lista com os estados de serviço da assinatura. Pode conter os valores “on_operation” (serviço em operação/expediente) ou “out_of_operation” (serviço fora de operação/de expediente). Na ausência deste parâmetro, os serviços serão monitorados sem filtro de estado;

- **lista_de_midias**: lista com as mídias habilitadas nos serviços da assinatura. Pode conter os valores “voice”, “email” ou “chat”. Na ausência deste parâmetro, os serviços serão monitorados sem filtro de mídia;
- **lista_de_tipos_de_classificacao**: lista com os tipos de classificação da assinatura. Pode conter os valores “inbound” (classificações para serviço receptivo) ou “outbound” (classificações para serviço ativo). Na ausência deste parâmetro, as classificações serão monitoradas sem filtro de tipo.

ABERTURA DE CANAL PARA RECEPÇÃO DE EVENTOS (COMANDO OPEN_CHANNEL)

A recepção de eventos será sempre realizada por meio de canais. Um canal é uma conexão HTTP persistente, que deverá ser aberta por meio do comando **open_channel**.

Cada canal possui um identificador único (**channel_id**). Este identificador deve ser informado como um parâmetro passando no comando **open_channel** ou, se não informado, será atribuído automaticamente pelo **Intercat CTI**. Em qualquer caso, o **channel_id** é indicado no evento **on_open_channel**.

Na abertura do canal deverá ser informado um formato para a recepção dos dados (XML ou JSON).

O identificador do canal deverá ser um valor na faixa entre 1 e 9999, e poderá ser associado a várias assinaturas.

Um canal é fechado automaticamente depois de 1 minuto sem alguma assinatura associada. Isso vale também para o caso de se abrir um canal sem nenhuma assinatura associada. Ele permanecerá aberto por 1 minuto e, caso nenhuma assinatura seja feita nesse intervalo, ele será automaticamente fechado.

Método HTTP POST

interact_cti/v1.0/config/monitor/open_channel?channel_id=nnnn&format=[xml | json]

POST DATA em JSON:

```
{
  ”channel_id”: identificador_do_canal
}
```

POST DATA em XML:

```
<request>
  <channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

- **identificador_do_canal**: número idenficador do canal que se deseja abrir.

Caso se deseje abrir um canal com um identificador (**channel_id**) que já existe ou fora da faixa válida (**entre 1 e 9999**), será retornado o erro **HTTP 403 Forbidden**.

FECHAMENTO DE CANAIS DE RECEPÇÃO DE EVENTOS (COMANDO CLOSE_CHANNEL)

Um canal pode ser fechado por meio do envio do comando **close_channel**, informando-se o seu identificador numérico (**channel_id**) e um formato para a recepção dos dados (XML ou JSON).

Ressalta-se que um canal será automaticamente fechado depois de 1 minuto sem alguma assinatura associada, independente do envio do comando **close_channel**.

Método HTTP POST

interact_cti/v1.0/config/monitor/close_channel?format=[xml| json]

POST DATA em JSON:

```
{
  “channel_id”: identificador_do_canal
}
```

POST DATA em XML:

```
<request>
  <channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

- **identificador_do_canal**: número idenficador do canal que se deseja fechar.

MANUTENÇÃO DO CANAL ABERTO

Para que uma conexão HTTP se mantenha de forma persistente é necessário que haja tráfego de pacotes desde o servidor até o cliente (autor da requisição) com certa frequência, caso contrário, a conexão será derrubada por time-out (que pode ocorrer tanto em um browser como em um servidor intermediário, entre o servidor do **Interact CTI** e a aplicação cliente).

O **Interact CTI** enviará um pacote de dados vazio apenas para sinalizar a atividade do canal e manter a conexão estabelecida.

O pacote de “heart beat” possui o seguinte formato: **data: {}**

RECEPÇÃO DOS EVENTOS PELO CANAL

Assinaturas de monitoração de listas no recurso **config** provocarão o envio do evento on_change, que indica que houve alguma alteração na lista(s) monitorada(s). Além deste evento, somente os eventos de controle (veja capítulo Eventos de Controle) são enviados para este recurso.

Os eventos serão enviados dentro do padrão de **Server Send Events da W3C** (**http://www.w3.org/TR/eventsource**). O evento **on_change** tem o seguinte formato:

event: on_change

data: dados_do_evento

Onde:

- **dados_do_evento:** pacote de dados com as informações do evento.

O formato dos dados do evento depende do **target** que gerou o evento (**agents_list**, **services_list**, **classification_list** ou **pause_type_list**). A seguir são apresentados os formatos para cada **target**:

Evento de alteração da lista de serviços (target services_list):

Em JSON:

```
{
    "target":"services_list",
    "notification_mode": modo_de_notificacao,
    "services": [ dados_do_servico_1,
                        dados_do_servico_2,
                           …
                        dados_do_servico_n
                      ]
}
```

Em XML:

```
<data>
    <target> services_list </target>
    <notification_mode> modo_de_notificacao </notification_mode>
  <services>
   <member> dados_do_servico_1 </member>
    <member> dados_do_servico_2 </member>
    ...
```

```
    <member> dados_do_servico_n </member>
  </servicos>
</data>
```

Onde:

- **modo_de_notificacao**: indica qual o modo de notificação do item: “complete” ou “changes”;
- **dados_do_servico_x**: conjunto de informações de um serviço, conforme formato a seguir:

Em JSON:

```
{
  "name": nome_do_servico,
  "type": tipo_do_servico,
  "media_types": lista_de_midias,
  “bots”:[{
         “type”: tipo_sub_midia,
         “data”: dados_da_sub_midia
  }]
  "state": estado_do_servico,
  "operation_begin": timestamp_inicio_operacao,
  "operation_end": timestamp_termino_operacao
}
```

Em XML:

```
<name> nome_do_servico </name>
<type> tipo_do_servico </type>
<media_types> lista_de_midias </media_types>
<bots>
  <member>
   <type> tipo_sub_midia </type>
   <data> dados_da_sub_midia </data>
  <member>
 </bots>
<state> estado_do_servico </state>
<operation_begin> timestamp_inicio_operacao </operation_begin>
<operation_end> timestamp_termino_operacao </operation_end>
```

Onde:

- nome_do_servico: nome do serviço;
- tipo_do_servico: tipo do serviço. Para serviços receptivos, esse campo terá valor ***inbound***. Para serviços ativos, este poderá ter os seguintes valores: ***ready***, ***preview***, ***predictive***, ***power*** ou ***validation*** (teste de lote);
- lista_de_midias: lista com as mídias habilitadas no serviço. Pode conter os valores “voice”, “email” ou “chat”;
- tipo_sub_midia: informar qual o *Chat* de terceiros que o serviço poderá usar. Pode conter o valor: “*telegram*” ou “*fb_messenger*”;
- dados_da_sub_midia: informa os dados necessários para comunicação com o *Chat* de terceiros. No caso de sub mídia Telegram este campo informa o token para comunicação com o bot.

- estado_do_servico: estado do serviço. Pode conter os valores “on_operation” (serviço em operação/expediente) ou “out_of_operation” (serviço fora de operação/de expediente);
- timestamp_inicio_operacao: data configurada de início da operação do serviço em formato UNIX timestamp. Se a data de início não estiver configurada ,o valor deste atributo será 0 (zero);
- timestamp_termino_operacao: data configurada de término da operação do serviço em formato UNIX timestamp. Se a data de término não estiver configurada, o valor deste atributo será 0 (zero).

NOTA

As informações do atributo bots só serão enviadas quando a mídia de chat estiver habilitada e com os dados dos bots configurados.

Evento de alteração da lista de agentes (target agents_list):

Em JSON:

```
{
    "target":"agents_list",
    "notification_mode": modo_de_notificacao,
    "agents": [ dados_do_agente_1,
                    dados_do_agente_2,
                     …
                    dados_do_agente_n
                  ]
}
```

Em XML:

```
<data>
  <target> agents_list </target>
  <notification_mode> modo_de_notificacao </notification_mode>
    <services>
    <member> dados_do_agente_1 </member>
    <member> dados_do_agente_2 </member>
    ...
    <member> dados_do_agente_n </member>
  </servicos>
</data>
```

Onde:

- **modo_de_notificacao**: indica qual o modo de notificação do item: “complete” ou “changes”;
- **dados_do_agente_x**: conjunto de informações de um agente, conforme formato a seguir:

Em JSON:

```
{
  "name": login_do_agente,
  "dt_login": data_de_login_do_agente,
  "state": estado_do_agente,
  "state_id": id_do_estado_do_agente
  "branch": ramal_do_agente,
```

```
  "contingency": estado_de_contingencia
}
```

Em XML:

```
<name> login_do_agente </name>
<dt_login> data_de_login_do_agente </dt_login>
<state> estado_do_agente </state>
<state_id> id_do_estado_do_agente </state_id>
<branch> ramal_do_agente </branch>
<contingency> estado_de_contingencia </contingency>
```

Onde:

- login_do_agente: nome de *login* do agente;
- data_de_login_do_agente: data de *login* do agente (este atributo não é informado se o agente não estiver logado);
- estado_do_agente: nome do estado do agente (ver lista de estados em Eventos Operacionais de Agentes e Chamadas);
- id_do_estado_do_agente: identificador do estado do agente;
- ramal_do_agente: ramal do agente (este atributo não é informado se o agente não estiver logado ou se não tiver mídia de voz configurada);
- estado_de_contingencia: informa se o agente está logado no modo de contingência (true) ou não (false).

Evento de alteração da lista de classificações (target classification_list):

Em JSON:

```
{
    "target":"classification_list",
    "notification_mode": modo_de_notificacao,
    "agents": [ dados_da_classificacao_1,
                      dados_da_classificacao_2,
                          …
                      dados_da_classificacao_n
                    ]
}
```

EmXML:

```
<data>
  <target> classiciation_list </target>
  <notification_mode> modo_de_notificacao </notification_mode>
  <services>
    <member> dados_da_classificacao_1 </member>
    <member> dados_da_classificacao_2 </member>
    ...
    <member> dados_da_classificacao_n </member>
  </servicos>
</data>
```

Onde:

- **modo_de_notificacao**: indica qual o modo de notificação do item: "**complete**" ou "**changes**";
- **dados_da_classificacao_x**: conjunto de informações de uma classificação, conforme formato a seguir:

Em JSON:

```
{
  "id": id_da_classificacao,
  "name": nome_da_classificacao,
  "scope": escopo_da_classificacao
}
```

Em XML:

```
<id> id_da_classificacao </id>
<name> nome_da_classificacao </name>
<scope> escopo_da_classificacao </scope>
```

Onde:

- id_da_classificacao: identificador numérico da classificação;
- nome_da_classificacao: nome da classificação;
- escopo_da_classificacao: escopo da classificação, que pode ser general (classificação aplicável a todas as chamadas de serviço), inbound (classificação aplicável a todas as chamadas de serviços receptivos), outbound (classificação aplicável a todas as chamadas de serviços ativos) ou, ainda, pode ser o nome do serviço específico ao qual a classificação se aplica.

Evento de alteração da lista de motivos de pausa (target pause_type_list):

Em JSON:

```
{
    "target":"pause_type_list",
    "notification_mode": modo_de_notificacao,
    "agents": [ dados_da_pausa_1,
                    dados_da_pausa_2,
                       …
                    dados_da_pausa_n
                ]
}
```

Em XML:

```
<data>
  <target> pause_type_list </target>
  <notification_mode> modo_de_notificacao </notification_mode>
  <services>
    <member> dados_da_pausa_1 </member>
    <member> dados_da_pausa_2 </member>
    ...
    <member> dados_da_pausa_n </member>
  </servicos>
</data>
```

Onde:

- **modo_de_notificacao**: indica qual o modo de notificação do item: “complete” ou “changes”;
- **dados_da_pausa_x**: conjunto de informações de uma pausa, conforme formato a seguir:

Em JSON:

```
{
  "id": id_da_pausa,
  "pause": nome_da_pausa,
  "total": tempo_total,
  "partial": tempo_parcial
}
```

Em XML:

```
<id> id_da_pausa </id>
<pause> nome_da_pausa </pause>
<total> tempo_total </total>
<partial> tempo_parcial </partial>
```

Onde:

- **id_da_pausa**: identificador do motivo de pausa;
- **nome_da_pausa**: nome do motivo de pausa;
- **tempo_total**: tempo máximo em que o agente pode permanecer nesse motivo de pausa por turno de trabalho. Ao ultrapassar esse tempo o agente

é colocado em "pausa por falha". O valor zero significa que não há limite de tempo;

- **tempo_parcial**: tempo máximo por entrada que o agente terá para utilizar esta pausa (em minutos). Ao ultrapassar esse tempo o agente é colocado em “pausa por falha”. O valor zero significa que não há limite de tempo.

6

CONTROLE DE AGENTES E CHAMADAS

Para o controle de agentes estão disponíveis os comandos:

- login;
- logout;
- config;
- set_state.

Para o controle de chamadas estão disponíveis os comandos:

- call_private;
- call_service;
- call_hold;
- call_retrieve;
- call_consult;
- call_accept;

- call_include_conference;
- call_exclude_conference;
- call_transfer;
- call_chat_message;
- call_associate_data;
- call_terminate;
- after_call_terminate;
- call_classify.

NOTA

É necessário licença para utilizar esses recursos.

LOGIN DE AGENTE (COMANDO LOGIN)

O comando **login** permite realizar a operação de *login* de um agente.

Método HTTP POST

interact_cti/v1.0/agent/login?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{  "agent": login_do_agente,
   "password": senha,
   "branch": ramal,
   "web_rtc": voz_via_web_rtc,
   "chat_devices": dispositivos_de_chat,
   "email_devices": dispositivos_de_email
}
```

POST DATA em XML:

```
<request>
  <agent>login_do_agente</agent>
  <password>senha</password>
  <branch>ramal</branch>
  <web_rtc>voz_via_web_rtc</web_rtc>
  <chat_devices>dispositivos_de_chat</amnt_chat>
  <email_devices>dispositivos_de_email</amnt_email>
</request>
```

Onde:

- **login_do_agente:** *login* do agente previamente cadastrado no **Interact Manager**;
- **senha**: senha de *login* do agente;
- **ramal**: número de ramal do agente (opcional);

- **voz_via_web_rtc**: boooleano (true ou false) para indicar se o ramal é Web RTC (opcional).
- **dispositivos_de_chat:** quantidade de chamadas de *Chat* que o agente atenderá simultaneamente (opcional);
- **dispositivos_de_email:** quantidade de chamadas de e-mail que o agente atenderá simultaneamente (opcional).

Para os **campos opcionais**, caso não seja enviada a configuração de ramal/*Chat/e-mail* será realizada a configuração do último *login* do agente. Caso seja o primeiro *login* e não sejam enviadas essas configurações, é necessário realizar a configuração das mídias a partir do comando ***/*agent/config** descrito na documentação. Caso o valor de *dispositivos* enviado em uma mídia seja igual a **zero**, ela será removida do agente.

IMPORTANTE

*A quantidade máxima de dispositivos de **Chat**, **e-mail** e **voz** que o agente pode configurar é definida no cadastro do agente no **Interact Manager**.*

A resposta em caso de **sucesso** para o comando pode ser observada a seguir.

Em JSON:

```
{"response":
   {"command":"login",
    "status":"ok",
    "agent":"login_do_agente",
    "branch": ramal,
    "chat_devices": dispositivos_de_chat,
```

```
    "email_devices": dispositivos_de_email
   }
}
```

Em XML:

```
<response>
  <command>login</command>
  <status>ok</status>
  <agent>login_do_agente</agent>
  <branch>ramal</branch>
  <chat_devices>dispositivos_de_chat</chat_devices>
  <email_devices>dispositivos_de_email</email_devices>
 </response>
```

Onde:

- login_do_agente;
- ramal: número de ramal alocado para o agente;
- dispositivos_de_chat: quantidade de chamadas de *Chat* que o agente pode atender simultaneamente;
- dispositivos_de_email: quantidade de chamadas de e-mail que o agente pode atender simultaneamente.

IMPORTANTE

Caso a quantidade de dispositivos na resposta não seja a que foi solicitada no comando de login, verifique se o agente possui a quantidade desejada nas configurações do agente no ***Interact Manager****.*

Em caso de **erro,** a resposta enviada será:

Em JSON:

```
{"response":
  {"command":"login",
   "status":"error",
   "error_type":"command_failed"
  }
}
```

Em XML:

```
<response>
  <command>login</command>
  <status>error</status>
  <error_type>command_failed</error_type>
</response>
```

LOGOUT DE AGENTE (COMANDO LOGOUT)

O comando **logout** permite realizar a operação de *logout* de um agente.

Método HTTP POST

interact_cti/v1.0/agent/logout?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
 "agent": login_do_agente,
 "password": senha
}
```

POST DATA em XML:

```
<request>
  <agent>login_do_agente</agent>
  <password>senha</password>
 </request>
```

Onde:

- **login_do_agente:** *login* do agente previamente cadastrado no **Interact Manager**.
- **senha:** senha do agente.

CONFIGURAÇÃO DO AGENTE (COMANDO CONFIG)

O comando **config** permite configurar os parâmetros operacionais de um agente.

Método HTTP POST

interact_cti/v1.0/agent/config?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
 "agent": login_do_agente,
 "devices": [ { "media_type": "voice",
                "handle_mode": modo_de_atendimento,
                "branch": ramal,
              },
               ...
              { "media_type": midia,
                "quantity": quantidade_de_dispositivos
                "handle_mode": modo_de_atendimento
              },
            ]
}
```

POST DATA em XML:

```
  <agent>login_do_agente</agent>
  <devices>
```

```
<member>
      <media_type>voice</media_type>
      <handle_mode>modo_de_atendimento</handle_mode>
      <branch>ramal</branch>
    </member>
    ...
    <member>
      <media_type>midia</media_type>
      <quantity>quantidade_de_dispositivos</quantity>
      <handle_mode>modo_de_atendimento</handle_mode>
    </member>
  <media_types>
```

Onde:

- **login_do_agente**: *login* do agente;
- **midia**: nome da mídia (por exemplo "voice", "email" ou "chat");
- **quantidade_de_dispositivos**: quantidade de dispositivos em uma determinada mídia. Esse é o número máximo de chamadas de uma determinada mídia que um agente pode atender simultaneamente (para voz deverá ser 0 ou 1);
- **modo_de_atendimento**: modo de atendimento:
  - *auto* (Automático);
  - *manual*;
  - *branch* (Configuração do ramal, somente para voz);
- **branch**: número do ramal (somente para voz);

A quantidade de dispositivos solicitada (atributo *quantity* para mídias diferentes de voz) será comparada com a já existente no agente. Caso a quantidade de dispositivos seja

menor que a configuração atual, a diferença entre essas quantidades será desconectada dentre os dispositivos livres dessa mídia no agente. Assim, se o agente tem 5 dispositivos de *Chat* e recebe nova configuração para operar com 3 dispositivos, a diferença será 2. Esse será o número de dispositivos desconectados do agente na mídia de *Chat*.

Caso não existam dispositivos livres, é necessário aguardar a finalização das chamadas em andamento para a liberação dos dispositivos do agente.

Se a quantidade de dispositivos enviado em uma mídia for igual a zero a mesma será removida do agente (a agente não atenderá mais chamadas nessa mídia).

NOTA

*Na situação em que o comando é utilizado para a alteração de configuração de um agente logado via **Interact MultiAgent** as mesmas não surtirão efeito até que o agente efetue o login novamente*

ALTERAÇÃO DE ESTADO DO AGENTE (COMANDO SET_STATE)

O comando **set_state** permite alterar o estado de um agente.

Método HTTP POST

interact_cti/v1.0/agent/set_state?format=[xml]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
  "agent": nome_do_agente,
  "state_id": identificador_do_estado
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
 <request>
  <agent>nome_do_agente</agent>
  <state_id>identificador_do_estado</state_id>
 </request>
```

Onde:

- nome_do_agente: nome do agente;
- identificador_do_estado: identificador do estado do agente (modo). O modo pode ser um dos seguintes:
  - 2 – Operando (está logado, não pausado).
  - Os demais valores possíves são os retornados pelo comando **get_pause_type_list**, que relaciona os motivos de pausa configurados no **Interact**.

SOLICITAÇÃO DE CHAMADAS EM ANDAMENTO (COMANDO GET_STATUS)

O comando **get_status** permite receber informações sobre o estado de agentes, incluindo dados das chamadas em andamento..

Método HTTP GET

interact_cti/v1.0/agent/get_status?agent=nome_do_agente&format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (**XML** ou **JSON**). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

O parâmetro agent também é opcional e permite solicitar informações de estado de um único agente. Na ausência deste parâmetro serão retornadas informações de todos agentes.

Resposta em JSON:

```
{
   "response":
  {
     "resource": "agent",
     "command": "get_status",
     "status": "ok",
     "agents": [ { "name": nome_do_agente,
                   "dt_login": data_hora_de_login,
```

```
                    "state": estado_do_agente,
                    "branch": numero_do_ramal,
                    "calls": [ { "media_type": nome_da_midia,
                        "interlocutor": identificador_do_interlocutor,
                        "service": nome_do_servico,
                        "call_id": identificador_da_chamada,
                        "duration": duracao_da_chamada,
                        "call_state": estado_da_chamada,
                        "call_type":tipo_da_chamada,
                        "call_source": origem_da_chamada,
                        "consultation_call_id":
identificador_da_chamada_de_consulta,
                             "entry_address":
identificador_de_endereco_de_entrada_email,
              "associated_data": dados_associados,
              "classification_time”: tempo_de_classificacao,
              "after_call_work_time": tempo_de_pos_atendimento,
              "expected_handle_time": tempo_esperado_de_atendimento,
              "permissions": permissoes
             },
             …
             { … }
            ],
"devices": [{“media_type”: nome_da_midia,
            "quantity": quantidade_de_dispositivos,
```

```
          "handle_mode": modo_de_atendimento
          },
          ...
          {...}
         ]
     },
  …
  { … }
  ]

}
}
```

Resposta em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <resource>agent</resource>
  <command>get_status</command>
  <status>ok</status>
  <agents>
    <member>
     <name> nome_do_agente </agent>
     <dt_login> data_hora_de_login </name>
     <state> estado_do_agente </state>
     <branch> numero_do_ramal </branch>
```

```
<calls>
 <member>
  <media_type>nome_da_midia</media_type>
  <interlocutor>identificador_do_interlocutor</intercolutor>
  <service>nome_do_servico</service>
  <call_id>identificador_da_chamada</call_id>
  <duration>duracao_da_chamada</duration>
  <call_state>estado_da_chamada</call_state>
  <call_type>tipo_da_chamada</call_type>
  <call_source>origem_da_chamada</call_source>
  <consultation_call_id>identificador_da_chamada_de_consulta
</consultation_call_id>
  <entry_address>identificador_de_endereco_de_entrada_email
</entry_address>
  <associated_data>dados_associados</associated_data>
  <classification_time> tempo_de_classificacao</
classification_time>
  <after_call_work_time> tempo_de_pos_atendimento </after_
call_work_time>
  <expected_handle_time> tempo_esperado_de_atendimento
</expected_handle_time>
    <permissions> permissoes </permissions>
 </member>
 ...
 <member> … </member>
</calls>
```

```
      <devices>
      <member>
       <media_type>nome_da_midia</media_type>
       <quantity>quantidade_de_dispositivos</quantity>
       <handle_mode>modo_de_atendimento</handle_mode>
       </member>
       ...
       <member>...</member>
      </devices>
    </member>
    …
    <member> … </member>
  </agents>
</response>
```

Onde:

- **nome_do_agente**: nome de cadastro do agente no sistema;
- **data_hora_de_login**: data e horário de *login* do agente no **Interact**;
- **estado_do_agente**: estado do agente;
- **numero_do_ramal**: número do ramal associado ao agente;
- **nome_da_midia**: nome do tipo de mídia da chamada (“voice”, “email” ou “chat”);

- **identificador_do_interlocutor**:identificador do interlocutor (pode ser um número de telefone para chamadas de voz, um endereço de e-mail para mensagens de e-mail ou um nome para um cliente de *Chat*);
- **nome_do_servico**: nome do serviço pelo qual a chamada chegou ao agente;
- **identificador_da_chamada**: identificador único da chamada (inclui a data de geração da chamada) e pode ser utilizado como chave para integração com outros produtos da empresa e cruzamento com bancos de dados;
- **duração_da_chamada**: duração da chamada (em segundos);
- **estado_da_chamada**: estado da chamada, o qual pode ser:
  - ready: pronto;
  - dialing:processando chamada (“discando”);
  - ringing: tocando;
  - calling: chamando;
  - busy: ocupado;
  - conference: em conferência;
  - held: em espera;
  - paused: em pausa;
  - after_call:pós-atendimento;
  - classifying: em classificação;
  - unavailable: indispónível;
  - fail: em falha;
  - reserved: reservado;
  - queued: em fila;
  - waiting_authorization: aguardando autorização;
  - authorized: autorizada;

  - blocked: bloqueada;
  - monitoring: em monitoração;
  - unknown: desconhecido.
- **tipo_da_chamada**: tipo da chamada, o qual pode ser:
  - inbound: receptiva (entrante);
  - outbound: ativa (sainte);
  - unknown: desconhecido;
- **origem_da_chamada**: origem da chamada, a qual pode ser:
  - internal: interna;
  - external: externa;
  - internal consultation: interna de consulta;
  - external consultation: externa de consulta;
  - internal conference: interna de conferência;
  - external conference: externa de conferência;
  - internal callback: interna de *Callback*;
  - external callback: externa de *Callback*;
  - unknown: desconhecida.
- **identificador_da_chamada_de_consulta:** identificador da chamada em consulta (esse campo só será enviado no caso da existência de uma chamada em consulta associada).

Observação:

Esse campo não é enviado no caso de consulta com mídia cruzada (Ex.: chamada de voz e consulta via *Chat*);

- **identificador_de_endereco_de_entrada_de_email**: endereço de entrada de e-mails configurado para o serviço. Esse campo apenas é enviado no caso da chamada ser de e-mail;
- **identificador_de_ramal_de_entrada**: ramal de entrada de chamadas de voz configurado para o serviço (ponto de roteamento). Esse campo apenas é enviado no caso da chamada ser de voz;
- **dados_associados**: campo de dados associados à chamada.;
- **tempo_de_classificacao**: tempo limite (em segundos) para a classificação da chamada;
- **tempo_de_pos_atendimento**: tempo limite (em segundos) o pós-atendimento da chamada.
- **tempo_esperado_de_atendimento**: duração esperada para o atendimento (em segundos);
- **permissoes**: é um array com um conjunto de operações que podem ser realizadas sobre a chamada. As operações possíveis são:
  - consult: permissão para realizar consulta;
  - transfer: permissão para realizar transferência;
  - conference: permissão para realizar conferência;
  - email_forward: permissão para encaminhar mensagens de e-mail (somente para chamadas de e-mail)
- **nome_da_midia**: nome do tipo de mídia (“voice”, “email” ou “chat”);
- **quantidade_de_dispositivos**: quantidade de dispositivos de um determinado tipo de mídia;
- **modo_de_atendimento**: modo de atendimento configurado. Pode ter os valores **auto** (atendimento automático) ou **manual** (atendimento que requer aceite do agente).

CHAMADA PESSOAL (COMANDO CALL_PRIVATE)

O comando **call_private** permite gerar uma chamada particular.

Método HTTP POST

interact_cti/v1.0/agent/call_private?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XMLou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
 "agent": login_do_agente,
 "to": destino,
 "media_type": midia,
 "account": conta,
 "password": senha,
 "route": rota
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <agent>login_do_agente</agent>
```

```
        <to>destino</to>
        <media_type>midia</media_type>
        <account>conta</account>
        <password>senha</password>
        <route>rota</route>
</request>
```

Onde:

- **login_do_agente**: *login* do agente que está executando o comando;
- **destino**: destino da chamada. Pode ser um ramal, agente, serviço ou número externo;
- **midia**: mídia usada para a geração da chamada. Os valores podem ser:
  - *chat;*
  - *voice;*
  - *email;*
- **conta**: conta usada para a geração de chamadas externas (opcional);
- **senha**: senha da conta usada para a geração de chamadas externas (Opcional);
- **rota**: rota utilizada na geração de chamadas externas (opcional).

Os parâmetros **opcionais** podem ser enviados conforme a necessidade de geração de chamadas. Porém, não são necessários em chamadas internas.

CHAMADA DE SERVIÇO (COMANDO CALL_SERVICE)

O comando **call_service** permite gerar uma chamada de serviço.

Método HTTP POST

interact_cti/v1.0/agent/call_service?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XMLou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
 "agent": login_do_agente,
 "to": destino,
 "media_type": midia,
 "service": nome_do_servico,
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <agent>login_do_agente</agent>
```

```
        <to>destino</to>
        <media_type>midia</media_type>
        <service>nome_do_servico</service>
</request>
```

Onde:

- **login_do_agente**: *login* do agente que está executando o comando;
- **destino**: destino da chamada. Pode ser um ramal, agente, serviço ou número externo;
- **midia**: mídia usada para a geração da chamada. Os valores podem ser:
  - *chat;*
  - *voice;*
  - *email;*
- **nome_do_servico**: nome do serviço utilizado para realizar a chamada.

COLOCAÇÃO DE CHAMADA EM ESPERA (COMANDO CALL_HOLD)

O comando **call_hold** permite que uma chamada seja colocada em espera.

Método HTTP POST

interact_cti/v1.0/agent/call_hold?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XMLou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
  "call_id": identificador_da_chamada,
  "agent": nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
       <agent>nome_do_agente</agent>
</request>
```

Onde:

- **identificador_da_chamada**: Identificador da chamada.
- **nome_do_agente**: deverá ser enviado com o nome do agente caso a chamada tenha mais de um agente envolvido. Caso haja apenas um agente envolvido, este parâmetro é opcional.

NOTA

Para as chamadas de voz, o comando de call_hold vai colocar a chamada em música e manterá o mesmo call_id. Mas para uma chamada de e-mail, a chamada é transferida para um dispositivo de retenção e trocará seu call_id. Para obter a lista de e-mails no dispositivo de retenção deve ser usado o comando ***get_held_emails_list****.*

RETIRADA DE CHAMADA EM ESPERA (COMANDO CALL_RETRIEVE)

O comando **call_retrieve** permite retirar uma chamada da espera.

Método HTTP POST

interact_cti/v1.0/agent/call_retrieve?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
  "call_id":identificador_da_chamada,
  "agent":nome_do_agente,
  "service":nome_do_servico
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
       <agent>nome_do_agente</agent>
       <service>nome_do_servico</service>
</request>
```

Onde:

- **identificador_da_chamada**: identificador da chamada.
- **nome _do_agente:** deverá ser enviado como nome do agente caso a chamada tenha mais de um agente envolvido. Caso haja apenas um agente envolvido, este parâmetro é opcional.
- **nome_do_servico:** o nome do serviço só deve ser enviado quando for para uma chamada de e-mail.

CONSULTA SOBRE CHAMADA (COMANDO CALL_CONSULT)

O comando **call_consult** permite realizar uma chamada de consulta sobre uma chamada em andamento.

Método HTTP POST

interact_cti/v1.0/agent/call_consult?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XMLou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
   "to": destino,
   "call_id": identificador_da_chamada,
    "agent": nome_do_agente,
    "media_type": midia
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
      <to>destino</to>
      <call_id>identificador_da_chamada</call_id>
      <agent>nome_do_agente</agent>
      <media_type>midia</media_type>
</request>
```

Onde:

- **destino**: destino da consulta. Pode ser um Agente, Ramal ou Serviço.
- **identificador_da_chamada**: identificador da chamada original.
- **nome_do_agente**: deverá ser enviado com o nome do agente caso a chamada tenha mais de um agente envolvido. Caso haja apenas um agente envolvido, este parâmetro é opcional.
- **midia**: mídia utilizada para realizar a consulta.

ACEITE DE CHAMADA POR AGENTE (COMANDO CALL_ACCEPT)

O comando **call_accept** deve ser enviado para indicar o aceite de uma chamada direcionada para um agente quando este está configurado no modo de atendimento manual.

Quando o agente estiver operando no modo de atendimento manual, e uma chamada for direcionada para ele, será gerado um evento **on_call_receive** com o atributo **must_accept** com valor ***true***.

Método HTTP POST

interact_cti/v1.0/agent/call_accept?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XMLou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

IMPORTANTE

Caso a chamada não seja aceita durante o tempo para atendimento manual configurado no serviço, o agente entrará em pausa por falha e a chamada será devolvida para a fila do serviço.

POST DATA em JSON:

```
{
  "call_id":identificador_da_chamada
  "agent":nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
</request>
```

Onde:

- **identificador_da_chamada**: identificador da chamada.
- **nome_do_agente**: deverá ser enviado com o nome do agente caso a chamada tenha mais de um agente envolvido. Caso haja apenas um agente envolvido, este parâmetro é opcional.

INCLUSÃO DE PARTICIPANTE EM CONFERÊNCIA (COMANDO CALL_INCLUDE_CONFERENCE)

O comando **call_include_conference** é válido apenas para chamadas de voz e permite estabelecer uma conferência de áudio (chamada com mais de 2 participantes).

Esse comando deve ser enviado sempre sobre uma chamada de consulta gerada pelo agente. O cenário previsto é aquele em que o agente A receba uma chamada de B e quer fazer uma conferência com C. Nesse caso, A realizará uma consulta para C e B ficará em espera. Após o atendimento de C, é enviado o comando de conferência passando o identificador da chamada de consulta como parâmetro. Será estabelecida uma conferência entre A, B e C.

Cada novo participante adicionado a uma conferência gera um novo identificador de chamada (**call_id**), que pode ser capturado pelo evento **on_call_receive**. Os identificadores anteriores (da chamada original e da chamada de consulta) são eliminados.

Método HTTP POST

interact_cti/v1.0/agent/call_include_conference?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
  "call_id": identificador_da_chamada,
```

```
  "agent": nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
       <agent>nome_do_agente</agent>
</request>
```

Onde:

- **identificador_da_chamada**: identificador da chamada.
- **nome_do_agente**: deverá ser enviado com o nome do agente dono da conferência no caso da chamada ter mais de um agente envolvido. Caso haja apenas um agente envolvido, este parâmetro é opcional.

EXCLUSÃO DE PARTICIPANTE DE CONFERÊNCIA (COMANDO CALL_EXCLUDE_CONFERENCE)

O comando **call_exclude_conference** permite excluir um participante de uma conferência.

Para tanto, deve ser informado um atributo identificando o participante a ser excluído, que pode ser o nome de um agente ou o número de um ramal.

Método HTTP POST

interact_cti/v1.0/agent/call_exclude_conference?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
   "call_id":identificador_da_chamada,
   "contact":contato,
   "agent":nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
       <contact>contato</contact>
       <agent>nome_do_agente</agent>
</request>
```

Onde:

- **identificador_da_chamada**: Identificador da chamada;
- **contato:** contato que será excluido da conferência.

- **nome_do_agente:** deverá ser enviado com o nome do agente dono da conferência no caso da chamada ter mais de um agente envolvido. Caso haja apenas um agente envolvido, este parâmetro é opcional.

TRANSFERÊNCIA DE CHAMADA (COMANDO CALL_TRANSFER)

O comando **call_transfer** permite realizar a transferência de uma chamada.

Método HTTP POST

interact_cti/v1.0/agent/call_transfer?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

O conteúdo do POST para os comandos de transferência podem vistos a seguir:

POST DATA em JSON:

```
{
  "call_id": identificador_da_chamada,
  "to": destino
  "agent": nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
```

```
        <call_id>identificador_da_chamada</call_id>
        <to>destino</to>,
        <agent>nome_do_agente</agent>
</request>
```

Onde:

- **identificador_da_chamada**: identificador da chamada;
- **destino:** destino da transferência. Pode ser um Serviço, Agente ou Ramal;
- **nome_do_agente:** deverá ser enviado com o nome do agente a partir do qual se deseja realizar uma transferência. Caso haja apenas um agente envolvido, este parâmentro é opcional.

ENVIO DE MENSAGEM EM *CHAT* (COMANDO CALL_CHAT_MESSAGE)

O comando **call**_*Chat*_**message** permite o envio de uma mensagem de texto em uma chamada de *Chat*.

Método HTTP POST

interact_cti/v1.0/agent/call_chat_message?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
   "call_id": identificador_da_chamada,
   "to": destino,
   "message": texto_da_mensagem,
   "agent": nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
      <call_id>identificador_da_chamada</call_id>
      <to>destino</to>
      <message>texto_da_mensagem</message>
      <agent>nome_do_agente</agent>
</request>
```

Onde:

- **identificador_da_chamada**: identificador da chamada;
- **destino:** destino da mensagem de *Chat* . Pode ser o nome do agente que atendeu a chamada ou o nome do interlocutor da chamada;
- **texto_da_mensagem:** texto que se deseja enviar;

- **nome_do_agente:** deverá ser enviado com o nome do agente que enviará a mensagem. Caso haja apenas um agente envolvido, este parâmetro é opcional.

NOTIFICAÇÃO DE DIGITAÇÃO EM *CHAT* (COMANDO NOTIFY_TYPING_STATUS)

O comando **notify_typing_status** permite o envio de uma notificação do estado da digitação de um agente. O estado pode ser o início ou o término da digitação. No caso do início da digitação, ao receber a notificação, o interlocutor poderá, por exemplo, apresentar a informação "agente NOME_AGENTE está digitando..."

Método HTTP POST

Interact_cti/v1.0/agente/notify_tipyng_status?format=[xml|json]

O parâmetro **format** é opcional e permite definer o format dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
   "call_id": identificador_da_chamada,
   "agent": nome_do_agente,]
   "status": estado_da_digitacao
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
      <call_id>identificador_da_chamada</call_id>
      <agent>nome_do_agente</agent>
      <status>estado_da_digitacao</status>
</request>
```

Onde:

- **identificador_da_chamada**: identificador da chamada;
- **nome_do_agente**: nome do agente que está iniciando ou finalizando a ação de digitação;
- **estado_da_digitacao:** estado da digitação. Pode ser **start** para indicar o início de digitação ou **stop** para indicar o término da digitação.

Observação:

Caso seja enviada uma notificação com status **start**, o interlocutor deverá interpretar que a digitação continua até que seja enviada uma notificação com status **stop**.

ENVIO DE ARQUIVO POR *CHAT* (COMANDOS REQUEST_FILE_TRANSFER / NOTIFY_FILE_TRANSFER)

Para enviar um arquivo em uma chamada de *Chat* é necessário enviar primeiramente uma solicitação. Caso a solicitação seja autorizada pelo interlocutor (informado no evento **on_file_transfer** – ver seção sobre **Eventos Operacionais**), aí então o arquivo poderá ser transferido.

O envio da solicitação é feito pelo comando:

Método HTTP POST

interact_cti/v1.0/agent/request_file_transfer

POST DATA em JSON:

```
{
  "agent": nome_do_agente,
  "call_id": identificador_da_chamada,
  "file_name": nome_do_arquivo,
  "file_size": tamanho_do_arquivo_em_bytes
}
```

Em XML:

```
<agent> nome_do_agente </agent>
```

```
<call_id> identificador_da_chamada </call_id>
<file_name> nome_do_arquivo </file_name>
<file_size> tamanho_do_arquivo_em_bytes </file_size>
```

Após a solicitação de transferência de arquivo ser autorizada, será necessário fazer o *upload* deste arquivo. A autorização é informada pelo evento **on_file_transfer** (tendo o atributo **status** com o valor **accepted**). Neste evento será informado um identificador de sessão (**session_id**) e um índice de arquivo (**file_index**). O identificador da sessão é necessário para se fazer a transferência do arquivo (*upload*) para a plataforma Dígitro e o índice do arquivo é necessário para se enviar a notificação da disponibilidade do arquivo e se realizar um eventual cancelamento da transferência. Todos os procedimentos são explicados a seguir.

A transferência do arquivo deverá ser feita pelo envio de um formulário HTTP, representado a seguir e que contém, além do arquivo, os identificadores da chamada (CALL_ID) e da sessão (SESSION_ID).

Método HTTP POST

file/v2

```
Content-type: multipart/form-data; boundary=STRING_DELIMITADORA
----------------------------STRING_DELIMITADORA\r\n
Content-Disposition: form-data; name="file";
filename="NOME_DO_ARQUIVO"\r\n
Content-Type: TIPO_DO_ARQUIVO\r\n\r\n
CONTEUDO DO ARQUIVO\r\n
----------------------------STRING_DELIMITADORA\r\n
```

```
Content-Disposition: form-data; name="session_id"\r\n\r\n
SESSION_ID\r\n
----------------------------STRING_DELIMITADORA\r\n
Content-Disposition: form-data; name="request_id"\r\n\r\n
CALL_ID\r\n
----------------------------STRING_DELIMITADORA--
```

Observação:

No modelo acima, as quebras de linha são representadas por **\r\n**, as quais são obrigatórias no formato de formulários HTTP.

Exemplo de pacote de dados para transferência de arquivo:

```
----------------------------117111589620148\r\n
Content-Disposition: form-data; name="file";
filename=\"teste.txt\"\r\n
Content-Type: text/plain\r\n\r\n
Teste de envio de arquivo\r\n
----------------------------117111589620148\r\n
Content-Disposition: form-data; name="session_id"\r\n\r\n
3ddb01b0-ee31-11e6-8312-2526ef0f1725\r\n
----------------------------117111589620148\r\n
Content-Disposition: form-data; name="request_id"\r\n\r\n
87ed5110-f44c-11e6-b589-00265a0f19a7\r\n
----------------------------117111589620148--
```

Em resposta a este POST será recebido um pacote de dados no seguinte formato:

```
{
  "files": [
    {
      "name": nome_do_arquivo,

      "uuid": identificador_do_arquivo
    }
  ]
}
```

Onde o identificador do arquivo é um UUID que fica associado ao arquivo e que permite que o arquivo seja recuperado.

Será necessário, agora, informar ao interlocutor que a transferência do arquivo foi bem sucedida, enviando o comando de notificação de transferência de arquivo indicando que o arquivo está disponível (atributo **status** com valor **available**).

Método HTTP POST

interact_cti/v1.0/agent/notify_file_transfer

POST DATA em JSON:

```
  {
    "agent": nome_do_agente,
    "call_id": identificador_da_chamada,
    "file_index": indice_do_arquivo,
```

```
    "file_id": identificador_do_arquivo
,
    "status": "available"
  }
```

Em XML:

```
  <agent> nome_do_agente </agent>
  <call_id> identificador_da_chamada </call_id>
  <file_index> indice_do_arquivo </file_index>
  <file_id> identificador_do_arquivo </file_id>
  <status> available </status>
```

A partir do envio deste comando, o interlocutor da chamada receberá as informações necessárias para baixar o arquivo transferido. Assim que o interlocutor acessar o arquivo, será enviado um evento **on_file_transfer** com o atributo **status** com o valor **accessed**.

Se houver algum erro no processo de transferência de arquivo para a plataforma Dígitro, é possível cancelar a transferência de arquivo com o comando:

Método HTTP POST

interact_cti/v1.0/agent/cancel_file_transfer

POST DATA em JSON:

```
  {
    "agent": nome_do_agente,
```

```
  "call_id": identificador_da_chamada,
  "file_index": indice_do_arquivo
}
```

Em XML:

```
<agent> nome_do_agente </agent>
<call_id> identificador_da_chamada </call_id>
<file_index> indice_do_arquivo </file_index>
```

RECEPÇÃO DE ARQUIVO POR *CHAT* (COMANDOS AUTHORIZE_FILE_TRANSFER / NOTIFY_FILE_TRANSFER)

Quando um arquivo em uma chamada de *Chat* é destinado a um agente, ele receberá o evento **on_file_transfer** que terá o atributo **status** com o valor **requested** e trará também o nome do arquivo (atributo **file_name**) e o índice do arquivo (atributo **file_index**).

Ressalta-se que este mesmo evento também é enviado quando o agente é quem está solicitando o envio do arquivo. Entretanto, o evento traz o atributo **actor** que identifica o autor da ação (neste caso uma requisição de transferência de arquivo) o que permite se fazer a distinção.

Para aceitar uma requisição de transferência de arquivo, deve ser enviado o comando de autorização de transferência:

Método HTTP POST

interact_cti/v1.0/agent/authorize_file_transfer

POST DATA em JSON:

```
{
  "agent": nome_do_agente,
  "call_id": identificador_da_chamada,
  "file_index": indice_do_arquivo,
  "action": "accept"
}
```

Em XML:

```
<agent> nome_do_agente </agent>
<call_id> identificador_da_chamada </call_id>
<file_index> indice_do_arquivo </file_index>
<action> accept </action>
```

O interlocutor fará a transferência do arquivo e, assim que finalizada, enviará uma notificação de disponibilidade do arquivo, que será recebida em um evento **on_file_transfer** com o atributo status com o valor **available**. Este evento trará também os atributos com o índice do arquivo (**file_index**) e o identificador do arquivo (**file_id**). Este último atributo deverá ser utilizado para baixar o arquivo da plataforma Dígitro por meio do comando:

HTTP GET

file/v1/FILE_ID

O arquivo será baixado e agora o agente deverá informar que acessou o arquivo, por meio do comando de notificação de transferência de arquivo:

Método HTTP POST

interact_cti/v1.0/agent/notify_file_transfer

POST DATA em JSON:

```
  {
    "agent": nome_do_agente,
    "call_id": identificador_da_chamada,
    "file_index": indice_do_arquivo,
    "file_id": identificador_do_arquivo
,
    "status": "accessed"
  }
```

Em XML:

```
  <agent> nome_do_agente </agent>
  <call_id> identificador_da_chamada </call_id>
  <file_index> indice_do_arquivo </file_index>
```

```
<file_id> identificador_do_arquivo </file_id>
<status> accessed </status>
```

RECUPERAÇÃO DE HISTÓRICO DE *CHAT* EM ANDAMENTO (COMANDO_ GET_CHAT_CALL_HISTORY)

O comando **get_chat_call_history** permite recuperar o histórico (inclusive arquivos) de chamadas de *Chat* em andamento. Este comando retorna um objeto com o histórico das conversas, com a troca de mensagens e arquivos (disponibilizados para *download* no serviço de transferência).

Pode ser passado o *login* de um agente para que seja retornado o histórico de cada *Chat* em andamento para este agente ou um identificador de chamada para que seja retornado o histórico de uma chamada específica.

Método HTTP GET

interact_cti/v1.0/agent/get_chat_call_history?[agent=login_do_agente] [&call_id=identificador_da_chamada]&format=[xml | json]

RESPONSE DATA em JSON :

```
{
  "calls":
  [
```

```
"call_id": identificador_da_chamada,
"agent": login_do_agente,
"agent_alias": nome_exibicao_do_agente,
"interlocutor": interlocutor_do_chat,
"form": dados_de_formulario,
"history":
[
  {
    "actor": nome_do_ator,
    "action": "message_sent",
    "message: texto_da_mensagem,
    "message_id": identificador_da_mensagem,
    "timestamp": timestamp_da_acao
  },
  {
    "actor": nome_do_ator,
    "action": nome_da_acao_de_arquivo,
    "file_name": nome_do_arquivo,
    "file_index": indice_do_arquivo,
    "file_id": identificador_do_arquivo,
    "download": url_para_download,
    "timestamp": timestamp_da_acao
  },
  {
    "actor": nome_do_ator,
```

```
        "action": nome_da_acao_de_video,
        "motive": motivo_de_cancelamento,
        "message": mensagem_de_cancelamento,
        "room_id": identificador_da_sala_de_video,
        "peer_id_from": identificador_da_conexao_origem,
        "peer_id_to": identificador_da_conexao_destino
        "timestamp": timestamp_da_acao
       },
      ...
      {...}
    ]
  }
}
```

RESPONSE DATA em XML:

```
<calls>
  <member>
  <call_id> identificador_da_chamada </call_id>
  <agent> login_do_agente </agent>
  <agent_alias> nome_exibicao_do_agente </agent_alias>
  <interlocutor> interlocutor_do_chat </interlocutor>
  <form> dados_de_formulario </form>
  <history>
    <member>
```

```
      <actor> nome_do_ator </actor>
      <action> "message_sent" </action>
      <message> texto_da_mensagem </message>
      <message_id> identificador_da_mensagem </message_id>
      <timestamp> timestamp_da_acao </timestamp>
    </member>
    <member>
      <actor> nome_do_ator </actor>
      <action> nome_da_acao </action>
      <file_name> nome_do_arquivo </file_name>
      <file_index> indice_do_arquivo </file_index>
      <file_id> identificador_do_arquivo </file_id>
      <download> url_para_download </download>
      <timestamp> timestamp_da_acao </timestamp>
    </member>
    ...
    <member>
      ...
    </member>
  </history>
</calls>
```

Onde:

- **identificador_da_chamada:** identificador único da chamada de *Chat*;
- **login_do_agente:** nome de *login* do agente;

- **nome_exibicao_do_agente:** nome de exibição configurado para o agente;
- **interlocutor_do_chat:** interlocutor da chamada de *Chat*;
- **dados_de_formulario:** array em que cada dado de formulário enviado na abertura do *Chat* é informado em uma estrutura com os campos **key** (chave) e **value** (valor);
- **nome_do_ator:** nome do responsável pela ação;
- **nome_da_acao_de_arquivo:** nome da ação de arquivo executada. Pode ser:
  - message_sent: ação de envio de mensagem de texto;
  - file_requested: ação de solicitação de transferência de arquivo;
  - file_accepted: ação de aceite de transferência de arquivo;
  - file_rejected: ação de rejeição de transferência de arquivo;
  - file_available: ação de notificação de disponibilidade de arquivo;
  - file_accessed: ação de notificação de acesso ao arquivo;
  - file_canceled: ação de cancelamento de transferência de arquivo.
- **texto_da_mensagem:** texto da mensagem enviada;
- **identificador_da_mensagem:** identificador numérico sequencial único de cada mensagem; em uma determinada chamada de *Chat*;
- **timestamp_da_acao:** timestamp da ação;
- **nome_do_arquivo:** nome do arquivo transferido;
- **indice_do_arquivo:** índice do arquivo transferido (número crescente iniciando em 1);
- **identificador_do_arquivo:** é o identificador para se acessar o arquivo no serviço de transferência de arquivos. Este atributo será enviado quando **nome_da_acao_de_arquivo** for *file_available* ou *file_accessed*;

- **url_para_download:** é o caminho completo para se realizar o *download* do arquivo no serviço de transferência de arquivos. Este atributo será enviado quando **nome_da_acao** for *file_available* ou *file_accessed*;
- **nome_da_acao_de_video:** ação de vídeo que pode ser:
  - -video_start: sessão de vídeo iniciada em uma chamada de *Chat*;
  - -video_origin_registry: registro da conexão de origem em uma sessão de vídeo;
  - -video_destiny_registry: registro da conexão de destino em uma sessão de vídeo;
  - -video_offer: sessão de vídeo oferecida pela origem em uma chamada de *Chat*;
  - -video_accept: sessão de vídeo foi aceita/estabelecida pelo interlocutor em uma chamada de *Chat*;
  - -video_stop: sessão de vídeo encerrada em uma chamada de *Chat*;
  - -video_canceled: sessão de vídeo foi cancelada (encerramento antes da sessão de vídeo ser estabelecida)
- **motivo_de_cancelamento:** atributo enviado somente se o atributo *nome_da_acao_de_video* for *canceled* e informa o motivo do cancelamento reportado pelo serviço de vídeo;
- **mensagem_de_cancelamento:** atributo enviado somente se o atributo *nome_da_acao_de_video* for *canceled* e informa a mensagem de cancelamento reportada pelo serviço de vídeo;
- **identificador_da_sala_de_video:** identificador único da sessão de vídeo no formato UUID;
- **identificador_da_conexao_origem:** identificador único da conexão origem (agente) no formato UUID;
- **identificador_da_conexao_destino:** identificador único da conexão destino (interlocutor de *Chat*) no formato UUID;

SOLICITAÇÃO DE ARQUIVO DE *E-MAIL* (COMANDO GET_EMAIL)

O comando **get_email** permite que o conteúdo da mensagem de um *e-mail* em atendimento seja baixado por meio do **Interact CTI**.

Método HTTP GET

interact_cti/v1.0/agent/get_email?call_id=identificador_da_chamada&format=[XML|JSON]

Resposta em JSON:

```
{
  "response":
  {
    "status": "ok",
    "email":
    {
      "call_id": identificador_da_chamada,
      "direction": direcao_da_chamada
      "header": { "subject": assunto,
                  "date": data,
                  "from": {"name": nome_origem, "address":
endereco_origem},
```

```
                    "to": [ {"name": nome_destino1, "address":
endereco_destino1},...,{...} ],
                    "cc": [ {"name": nome_copia1, "address":
endereco_copia1},...,{...} ],
"cco":[ {"name": nome_copia_oculta1, "address":
endereco_copia_oculta1},...,{...} ],
                    "reply_to": {"name": nome_responder_para ,
"address": endereco_responde_para }
                  },
      "body":   { "content_type": tipo_de_conteudo_corpo ,
                    "charset": codificacao_corpo,
                    "transfer_encoding" :
codificacao_de_transferencia_corpo,
                    "data": dados
                  },
      "attachments": [ { "name": nome_anexo1,
                         "size": tamanho_anexo1,
                         "content_type": tipo_de_conteudo_anexo1,
                         "charset": codificacao_anexo1,
                         "transfer_encoding":
codificacao_de_transferencia_anexo1,
                         "disposition": disposicao_anexo1,
                         "content_id": identificador_de_conteudo1,
                         "download": url_anexo1
                       }
                       ,
```

```
                        ...,
                        {...}
                      ]
    }
  }
}
```

Resposta em XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>ok</status>
  <email>
    <call_id> identificador_da_chamada </call_id>
    <direction> direcao_da_chamada </direction>
    <header>
      <subject> assunto </subject>
      <date> data </date>
      <from>
        <name> nome_origem </name>
        <address> endereco_origem </address>
      </from>
      <to>
        <member>
          <name> nome_destino1 </name>
          <address> endereco_destino1 </address>
```

```
    </member>
    ...
    <member>
      ...
    </member>
  </to>
  <cc>
    <member>
      <name> nome_copia1 </name>
      <address> endereco_copia1 </address>
    </member>
    ...
    <member>
      ...
    </member>
  </cc>
  <cco>
    <member>
      <name> nome_copia_oculta1 </name>
      <address> endereco_copia_oculta1 </address>
    </member>
    ...
    <member>
      ...
    </member>
```

```
      </cco>
      <reply_to>
        <name> nome_responder_para </name>
        <address> endereco_responder_para </address>
      </reply_to>
    </header>
    <body>
      <content_type> tipo_de_conteudo_corpo </content_type>
      <charset> codificacao_corpo </charset>
      <transfer_encoding> codificacao_de_transferencia_corpo
</transfer_encoding>
      <data> dados </data>
    </body>
    <attachments>
      <member>
        <name> nome_anexo1 </name>
        <size> tamanho_anexo1 </size>
        <content_type> tipo_de_conteudo_anexo1 </content_type>
        <charset> codificacao_anexo1 </charset>
        <transfer_encoding> codificacao_de_transferencia_anexo1
</transfer_encoding>
        <disposition> disposicao_anexo1 </disposition>
        <content_id> identificador_de_conteudo1 </content_id>
        <download> url_anexo1 </download>
      </member>
```

```
      ...
      <member>
      </member>
    </attachments>
  </email>
</response>
```

Onde:

- **identificador_da_chamada**: identificador único da chamada;
- **direcao_da_chamada**: poderá ser **in**, indicando chamada entrante ou **out** indicando chamada sainte;
- **assunto**: assunto da mensagem de e-mail;
- **data**: data em que a mensagem foi enviada pela origem;
- **nome_origem**: nome do originador da mensagem;
- **endereco_origem**: endereço de e-mail do originador da mensagem;
- **nome_destinoN**: nome do N-ésimo destinatário da mensagem;
- **endereco_destinoN**: endereço do N-ésimo destinatário da mensagem;
- **nome_copiaN**: nome do N-ésimo destino em cópia da mensagem;
- **endereco_copiaN**: endereço do N-ésimo destino em cópia da mensagem;
- **nome_copia_ocultaN**: nome do N-ésimo destino em cópia oculta da mensagem;
- **endereco_copia_ocultaN**: endereço do N-ésimo destino em cópia oculta da mensagem;
- **nome_responder_para**: nome do destino de resposta da mensagem;

- **endereco_responder_para**: endereço do destino de resposta da mensagem;
- **tipo_de_conteudo_corpo**: tipo de conteúdo (MIME type) do corpo da mensagem;
- **codificacao_corpo**: codificação do corpo da mensagem;
- **codificacao_de_transferencia_corpo**: codificação de transferência do corpo da mensagem;
- **dados**: dados da mensagem (em base64);
- **nome_anexoN**: nome do N-ésimo arquivo anexo da mensagem;
- **tamanho_anexoN**: tamanho em bytes do N-ésimo arquivo anexo da mensagem;
- **tipo_de_conteudo_anexoN**: tipo de conteúdo (MIME type) do N-ésimo arquivo anexo da mensagem;
- **codificacao_anexoN**: codificação do N-ésimo arquivo anexo da mensagem;
- **codificacao_de_transferencia_anexoN**: codificação de transferência do N-ésimo arquivo anexo da mensagem;
- **disposicao_anexoN**: disposição do N-ésimo arquivo anexo da mensagem. Pode ser **attachment** para arquivo anexo, ou **inline** para arquivo referenciado como parte do corpo da mensagem. Neste último caso, o atributo **content_id** possibilita identificar o local no corpo da mensagem em que o arquivo deve ser apresentado;
- **identificador_de_conteudoN**: identificador do conteúdo do anexo (utilizado apenas quando a disposição do mesmo for inline);
- **url_anexoN**: URL para se realizar o *download* do arquivo anexo.

Observação:

Os atributos **cc**, **cco** e **attachments** podem não possuir conteúdo, caso em que seu valor será **null** em JSON ou será uma **tag** vazia em XML.

ENVIO DE *E-MAIL* (COMANDO POST_EMAIL) - SEM ANEXOS

O envio de mensagens de *e-mail* pode ocorrer tanto para uma resposta a uma chamada de *e-mail* (entrante) como para uma chamada gerada (sainte).

O comando **post_email** permite enviar conteúdo da mensagem de um *e-mail* por meio do **Interact CTI**.

Método HTTP POST

interact_cti/v1.0/agent/post_email?call_id=identificador_da_chamada

POST DATA em JSON:

```
{
      “header”: { “subject”: assunto,
                  “to”: [ {"name": nome_destino1, "address":
endereco_destino1},...,{...} ],
                  “cc”: [ {"name": nome_copia1, "address":
endereco_copia1},...,{...} ],
                  “cco”:[ {"name": nome_copia_oculta1, "address":
endereco_copia_oculta1},...,{...} ]
      “body”:   { "content_type": tipo_de_conteudo_corpo ,
```

```
                    "charset": codificacao_corpo,
                    "transfer_encoding" :
codificacao_de_transferencia_corpo,
                    "data": dados
                  }
}
```

POST DATA em XML:

```
{
    <?xml version="1.0" encoding="UTF-8"?>
    <request>
      <header>
        <subject> assunto </subject>
        <to>
          <member>
             <name> nome_destino1 </name>
             <address> endereco_destino1 </address>
          </member>
          ...
          <member>
            ...
          </member>
        </to>
        <cc>
          <member>
```

```
                <name> nome_copia1 </name>
                <address> endereco_copia1 </address>
              </member>
              ...
              <member>
                ...
              </member>
            </cc>
            <cco>
              <member>
                <name> nome_copia_oculta1 </name>
                <address> endereco_copia_oculta1 </address>
              </member>
              ...
              <member>
                ...
              </member>
            </cco>
              </header>
          <body>
            <content_type> tipo_de_conteudo_corpo </content_type>
            <charset> codificacao_corpo </charset>
            <transfer_encoding> codificacao_de_transferencia_corpo
</transfer_encoding>
            <data> dados </data>
```

```
      </body>
    </request>
}
```

Onde:

- **identificador_da_chamada**: identificador único da chamada;
- **assunto**: assunto da mensagem de e-mail;
- **data**: data em que a mensagem foi enviada pela origem;
- **nome_destinoN**: nome do N-ésimo destinatário da mensagem;
- **endereco_destinoN**: endereço do N-ésimo destinatário da mensagem;
- **nome_copiaN**: nome do N-ésimo destino em cópia da mensagem;
- **endereco_copiaN**: endereço do N-ésimo destino em cópia da mensagem;
- **nome_copia_ocultaN**: nome do N-ésimo destino em cópia oculta da mensagem;
- **endereco_copia_ocultaN**: endereço do N-ésimo destino em cópia oculta da mensagem;
- **tipo_de_conteudo_corpo**: tipo de conteúdo (MIME type) do corpo da mensagem;
- **codificacao_corpo**: codificação do corpo da mensagem;
- **codificacao_de_transferencia_corpo**: codificação de transferência do corpo da mensagem;
- **dados**: dados da mensagem (em base64);

Observações:

1) Os atributos **cc**, **cco** e **attachments** podem não possuir conteúdo, caso em que seu valor podem ser omitidos ou enviados com conteúdo **null** em JSON ou **tag** vazia em XML.

2) A origem de uma mensagem de *e-mail* enviada será o ponto de roteamento do serviço pelo qual foi realizada a chamada e será automaticamente atribuída pelo **Interact CTI**.

ENVIO DE *E-MAIL* (COMANDO POST_EMAIL) - COM ANEXOS

Neste caso, cada arquivo de anexo deve ser postado para o serviço de transferência de arquivos, por meio do comando:

Método HTTP POST

IP:PORTA/file/v1

POST DATA: Arquivo

Observação: a requisição de envio deve ser com **Content-type: multipart/form-data**

Em caso de sucesso na transferência de arquivo, será retornado status HTTP 200 OK, e no conteúdo da mensagem será recebido um identificador do arquivo (UUID) em JSON:

Resposta:

```
{
  "files": [ { "name": nome_original,
               "UUID": "e3f4e36a-cbdc-11e3-c000-001e90cca34f-
3086670736-25699"
```

```
            }
          ]
}
```

Observações:

- 1) qualquer erro no serviço de transferência de arquivos será informado por meio de um valor de status diferente de **200**, como, por exemplo, **400** ou **404**.
- 2) o formato de retorno do serviço de transferência de arquivos será sempre JSON. Cada anexo deverá ser transferido e receberá um identificador único. Deve-se então, enviar o comando **post_email** com o atributo **attachments**:

Método HTTP POST

interact_cti/v1.0/agent/post_email?call_id=identificador_da_chamada

POST DATA em JSON:

```
{
  “header”: { “subject”: assunto,
              “to”: [ {"name": nome_destino1, "address":
endereco_destino1},...,{...} ],
              “cc”: [ {"name": nome_copia1, "address":
endereco_copia1},...,{...} ],
              “cco”:[ {"name": nome_copia_oculta1, "address":
endereco_copia_oculta1},...,{...} ],
              },
```

```
  “body”:   { "content_type": tipo_de_conteudo_corpo ,
              "charset": codificacao_corpo,
              "transfer_encoding" :
codificacao_de_transferencia_corpo,
              "data": dados
            },
  “attachments”: [ { "id:" identificador_do_anexo1,
                     "disposition": disposicao1
                   }
                   ...,
                   {...}
                 ]
}
```

POST DATA em XML:

```
{
   <?xml version="1.0" encoding="UTF-8"?>
   <request>
     <header>
        <subject> assunto </subject>
        <to>
          <member>
             <name> nome_destino1 </name>
             <address> endereco_destino1 </address>
          </member>
```

```
  ...
  <member>
    ...
  </member>
</to>
<cc>
  <member>
     <name> nome_copia1 </name>
     <address> endereco_copia1 </address>
  </member>
  ...
  <member>
    ...
  </member>
</cc>
<cco>
  <member>
     <name> nome_copia_oculta1 </name>
     <address> endereco_copia_oculta1 </address>
  </member>
  ...
  <member>
    ...
  </member>
</cco>
```

```
        </header>
       <body>
         <content_type> tipo_de_conteudo_corpo </content_type>
         <charset> codificacao_corpo </charset>
         <transfer_encoding> codificacao_de_transferencia_corpo
</transfer_encoding>
         <data> dados </data>
       </body>
       <attachments>
         <member>
           <id> identificador_do_anexo1 </id>
           <disposition> disposicao1 </disposition>
         </member>
         ...
         <member>
           ...
         </member>
      </attachments>
    </request>
```

Onde:

- **identificador_do_anexoN**: identificador (UUID) da N-ésimo anexo postado;
- **disposicaoN**: disposição do arquivo anexo em relação ao corpo da mensagem. Pode ser **inline** ou **attachment**.

Observação:

Em caso de envio de anexo **inline**, cada arquivo deverá ser referenciado no corpo da mensagem pelo seu identificador (UUID recebido do serviço de transferência de arquivos e enviado no campo **id** de **attachments**).

CONSULTAR A LISTA DE E-MAILS EM ESPERA (COMANDO GET_HELD_EMAILS_LIST)

O comando get_held_emails_list permite obter todos os *e-mails* em espera de um determinado agente.

Método HTTP POST

interact_cti/v1.0/agent/get_held_emails_lis?agent=nome_do_agente&format=[xml|json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

RESPOSTA em JSON:

```
{
    "response": {
        "resource": "agent",
        "command": "get_held_emails_list",
        "status": "ok"
```

```
        "held_emails":
        [
          {
            "service": nome_do_servico,
            "agent": nome_do_agente,
            "interlocutor": identificador_do_interlocutor
            "subject": assunto,
            "call_id": identificador_da_chamada,
            "arrival_date": data_início,
            "hold_time": tempo_hold
            "timeout": limite_de_tempo
          },
           …
          {…
          }
        }
    }
}
```

RESPOSTA em XML:

```
<?xml version="1.0" enconding="UTF-8"?>
<response>
  <command>get_held_emails_list</command>
 <resource>agent</resource>
 <status>ok</status>
```

```
  <held_emails>
   <member>
    <service>nome_do_servico</service>
    <agent>nome_do_agente</agent>
    <interlocutor>identificador_do_interlocutor</interlocutor>
    <subject>assunto</subject>
    <call_id>identificador_da_chamada</call_id>
    <arrival_date>data_inicio</arrival_date>
    <hold_time>tempo_hold</hold_time>
    <timeout>limite_de_tempo</timeout>
   </member>
   …
   <member>
  </held_emils>
</response>
```

Onde:

-
- **Nome_do_agente:** nome do agente;
- **identificador_do_interlocutor**: identificador de *e-mail* do remetente;
- **assunto:** assunto da mensagem de *e-mail;*
- **identificador_da_chamada**: identificador único da chamada;
- **data_início**: data em que o *e-mail* chegou no serviço;
- **tempo_hold**: tempo em “hold”;
- **limite_de_tempo**: data limite da retenção.

ASSOCIAÇÃO DE DADOS A UMA CHAMADA (COMANDO CALL_ASSOCIATE_DATA)

O comando **call_associate_data** permite que dados sejam associados a uma chamada.

Método HTTP POST

interact_cti/v1.0/agent/call_associate_data?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
"call_id" : identificador_da_chamada,
"from" : origem,
"data": dados_associados
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
       <from>origem</from>
```

```
        <data>dados_associados</data>
</request>
```

Onde:

- **identificador_da_chamada**: identificador da chamada;
- **origem**: nome do agente que está associando dados à chamada;
- **dados_associados**: dados que serão associados na chamada.

> **NOTA**
> *O limite de texto permitido para dados associados é de 100 bytes.*

FINALIZAÇÃO DE CHAMADA (COMANDO CALL_TERMINATE)

O comando **call_terminate** possibilita o encerramento de uma chamada.

Método HTTP POST

interact_cti/v1.0/agent/call_terminate?format=[xml |json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
  "call_id": identificador_da_chamada
  "agent": nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
<call_id>identificador_da_chamada</call_id>
<agent>nome_do_agente</agent>
</request>
```

Onde:

- **identificador_da_chamada** é o identificador da chamada.
- **nome_do_agente:** deverá ser enviado com o nome do agente que fará a ação de encerrar a chamadal. Caso haja apenas um agente envolvido, este parâmetro é opcional.

FINALIZAÇÃO DE PÓS-ATENDIMENTO (COMANDO AFTER_CALL_TERMINATE)

O comando **after_call_terminate** permite encerrar o pós-atendimento de uma chamada.

Método HTTP POST

interact_cti/v1.0/agent/after_call_terminate?format=[xml|json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
 "call_id": identificador_da_chamada,
 "agent": nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
        <call_id>identificador_da_chamada</call_id>
        <agent>nome_do_agente</agent>
</request>
```

Onde:

- **identificador_da_chamada:** identificador único da chamada.
- **nome_do_agente:** deverá ser enviado com o nome do agente que fará a ação de encerrar o pós-atendimento da chamada. Caso haja apenas um agente envolvido, este parâmetro é opcional.

CLASSIFICAÇÃO DE CHAMADA (COMANDO CLASSIFY)

O comando **classify** permite classificar uma chamada. O identificador da classificação deve ser obtido com o comando **get_classification_list**.

Método HTTP POST

interact_cti/v1.0/agent/classify?format=[xml|json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
  "call_id": identificador_da_chamada,
  "classify_id": identificador_da_classificacao
  "agent": nome_do_agente
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
<call_id>identificador_da_chamada</call_id>
<classify_id>identificador_da_classificacao</classify_id>
<agent>nome_do_agente</agent>
```

```
</request>
```

Onde:

- **identificador_da_chamada:** identificador da chamada.
- **identificador_da_classificacao:** identificador da classificação.
- **nome_do_agente:** deverá ser enviado com o nome do agente que fará a ação de classificar a chamada. Caso haja apenas um agente envolvido, este parâmetro é opcional.

INÍCIO DE INTERAÇÃO DE VÍDEO (COMANDO START_VIDEO)

O comando **start_video** permite iniciar uma interação de vídeo WebRTC sobre uma chamada de *Chat*.

Método HTTP POST

interact_cti/v1.0/agent/start_video?format=[xml|json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
  "call_id": identificador_da_chamada,
```

```
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
</request>
```

Onde:

- **identificador_da_chamada:** identificador da chamada de *Chat* sobre a qual se deseja iniciar uma interação de vídeo.

Resposta em JSON :

```
{

       "response:" {
              "resource": "agent",
              "command": "start_video",
              "status": status_do_comando,
              "call_id": identificador_da_chamada,
              "room_id": identificador_da_sala_de_video,
              "peer_id_from": identificador_da_conexao_origem,
              "peer_id_to": identificador_da_conexao_destino,
              "url": url_do_video,
```

```
        }
}
```

Resposta em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
        <resource>agent</resource>
        <command>start_video</command>
        <status>status_do_comando</status>
        <call_id>identificador_da_chamada</call_id>
        <room_id>identificador_da_sala_de_video</room_id>
        <peer_id_from>identificador_da_conexao_origem</peer_id_from>
        <peer_id_to>identificador_da_conexao_destino</peer_id_to>
        <url>url_do_video</url>
</response>
```

Onde:

- **status_do_comando:** será **ok** se o comando foi aceito ou **error** no caso de falha. No caso de erro será enviado também o atributo **error_type** informando o tipo de erro;
- **identificador_da_chamada:** identificador da chamada de *Chat* sobre a qual se deseja iniciar uma interação de vídeo;

- **identificador_da_sala_de_video:** identificador da sala de vídeo WebRTC;
- **identificador_da_conexao_origem:** identificador da conexão de origem do vídeo WebRTC;
- **identificador_da_conexao_destino:** identificador da conexão de destino do vídeo WebRTC;
- **url_do_video:** URL que deve ser aberta para visualização do vídeo do lado do agente.

FINALIZAÇÃO DE INTERAÇÃO DE VÍDEO (COMANDO STOP_VIDEO)

O comando **stop_video** finaliza uma interação de vídeo WebRTC sobre uma chamada de *Chat*.

Método HTTP POST

interact_cti/v1.0/agent/stop_video?format=[xml|json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
  "call_id": identificador_da_chamada,
 }
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
</request>
```

Onde:

- **identificador_da_chamada:** identificador da chamada de *Chat* sobre a qual se deseja finalizar uma interação de vídeo.

Resposta em JSON :

```
{

      "response:" {
            "resource": "agent",
            "command": "stop_video",
            "status": status_do_comando,
       }
}
```

Resposta em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<response>
      <resource>agent</resource>
      <command>stop_video</command>
      <status>status_do_comando</status>
</response>
```

Onde:

- **status_do_comando:** será **ok** se o comando foi aceito ou **error** no caso de falha. No caso de erro será enviado também o atributo **error_type** informando o tipo de erro.

7

MONITORAÇÃO DE ESTATÍSTICAS E ALARMES

A monitoração de estatísticas e alarmes está disponível para o recurso **supervisor** do **Interact CTI**. Uma aplicação deve assinar as variáveis estatísticas que deseja receber para que receba atualizações de seus valores em intervalos determinados (entre 5 e 60 segundos). Apenas são enviadas informações das variáveis estatísticas que forem assinadas e que sofreram alteração desde o último envio. Uma assinatura também pode especificar um conjunto de alarmes e as condições em que eles serão ativados.

A utilização da monitoração de estatísticas e alarmes depende de licenças específicas .

NOTA

*O **Interact CTI** guarda as informações das variáveis estatísticas em um cache interno, separado por assinatura, que é utilizada para identificar quais variáveis de estatísticas não sofreram alteração desde o último envio.*

Os serviços disponibilizados pelo **Interact CTI** para monitoração de estatísticas e alarmes por meio de REST têm o seguinte formato geral:

interact_cti/v1.0/supervisor/monitor/metodo?parametros[&format= (json|xml)]

Onde:

- **método:** nome da requisição (comando);
- **parâmetros**: conjunto de pares nome-valor na forma prm=valor, separados por & (E comercial - ampersand);
- **format:** define o formato desejado para a resposta, que pode ser no formato JSON (.json) ou no formato (.xml). Na ausência desse parâmetro, será utilizado o mesmo formato informado no header Content-type. Na ausência do header Content-type, será utilizado XML por *default*.

ASSINATURA DE EVENTOS PARA MONITORAÇÃO (COMANDO SUBSCRIBE)

O comando **subscribe** possibilita que estatísticas de agentes, estatísticas de serviços e alarmes sejam monitorados.

Método HTTP POST

interact_cti/v1.0/supervisor/monitor/subscribe?format=[xml | json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON:

```
{
    "subscription": {
        "id": nnnn,
        "channel_id": identificador_do_canal,
        "webhook": url_do_cliente,
        "expires": tempo_de_expiracao,
        "agents": lista_de_agentes,
        "services": lista_de_servicos,
        "media_types": lista_de_midias,
        "events": lista_de_eventos,
        "agent_stats": lista_de_estatisticas_de_agentes,
        "service_stats": lista_de_estatisticas_de_servicos,
        "alarms": lista_de_alarmes,
        "update_interval": intervalo_de_atualizacao,
    }
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
 <request>
  <subscription>
```

```
    <id> identificador_da_assinatura </id>
    <channel_id> identificador_do_canal </channel_id>
    <webhook> url_do_cliente </webhook>
    <expires> tempo_de_expiracao  </expires>
    <agents> lista_de_agentes </agents>
    <services> lista_de_servicos </services>
    <media_types> lista_de_midias </media_types>
    <events> lista_de_eventos </events>
    <agent_stats> lista_de_estatisticas_de_agentes </stats>
    <service_stats> lista_de_estatisticas_de_servicos </stats>
    <alarms> lista_de_alarmes </stats>
    <update_interval> intervalo_de_atualizacao <update_interval>
  </subscription>
 </request>
```

Onde:

- **identificador_da_assinatura** (opcional): identificador da assinatura. Deve ser enviado apenas caso se deseje modificar uma assinatura já existente;
- **identificador_do_canal:** identificador do canal de eventos pelo qual os eventos dessa assinatura deverão ser enviados. Os canais podem ter identificadores dentro da faixa numérica de 1 até 9999 e são de responsabilidade do integrador;
- **url_do_cliente:** endereço do webservice do cliente que espera receber eventos por HTTP POST;
- **tempo_de_expiracao**: tempo em segundos durante o qual a assinatura permanecerá ativa. O valor máximo é 86400, que corresponde a um dia

completo. Esse parâmetro é obrigatório. São aceitos valores entre 1 e 86400;

- **lista_de_agentes** (opcional): lista com os nomes dos agentes para monitoração (a ausência desse parâmetro ou o uso do valor "all" (ou *) indica monitoração de todos os agentes cadastrados no momento da assinatura - sujeito ao limite de licenças do **Interact CTI**);
- **lista_de_servicos** (opcional): lista com o nome dos serviços para monitoração (a ausência desse parâmetro ou o uso do valor "all" (ou *) indica monitoração de todos os serviços - sujeito ao limite de licenças do **Interact CTI**);
- **lista_de_midias** (opcional): lista de mídias para monitoração (a ausência desse parâmetro ou o uso do valor "all" (ou *) indica monitoração de todas as mídias);
- **lista_de_eventos** (opcional): lista com os nomes dos eventos de operação para monitoração (a ausência desse parâmetro ou o uso do valor "all" (ou *) indica monitoração de todos os eventos). Apenas os eventos de operação (aqueles relacionados a agentes e a chamadas) é que podem ser assinados.
- **lista_de_estatisticas_de_agentes:** lista contendo os nomes das estatísticas de agentes desejadas para monitoração (relacionadas no item Estatísticas Monitoráveis de Agentes);
- **lista_de_estatisticas_de_servicos:** lista contendo os nomes das estatísticas de serviços desejadas para monitoração (relacionadas no item Estatísticas Monitoráveis de Serviço);
- **intervalo_de_atualizacao**: intervalo de tempo em segundos que será respeitado para o envio de atualizações das estatísticas e alarmes para uma assinatura. Esse parâmetro é obrigatório. São aceitos valores entre 5 e 60. **Obs.:** Os dados estatísticos são mantidos em cache no **Interact CTI** para que apenas aqueles que sofreram alterações desde o último envio sejam enviados;

- **lista_de_alarmes**: lista de definições de alarmes monitorados (relacionadas no item Alarmes Monitoráveis) conforme a seguinte estrutura:

Em JSON:

```
"name": nome_do_alarme,
"value": valor_limite,
"media_type": midia
```

Em XML:

```
<name> nome_do_alarme </name>
<value> valor_limite </value>
<media_type> midia </media_type>
```

Onde:

- **nome_do_alarme**: nome do alarme que se deseja monitorar (relacionadas no item Alarmes Monitoráveis);
- **valor_limite**: valor limite a partir do qual o alarme deverá se sinalizado;
- **midia**: nome da mídia para qual vale o ajuste do alarme.

NOTA

Eventos de controle do canal (on_open_channel e on_close_channel), de disponibilidade do servidor (on_server_state_change) e de expiração de assinatura (on_subscription_expire) não são passíveis de serem assinados e sempre serão enviados aos clientes.

Exemplo em JSON:

```
{
  "subscription": {
   "channel_id": 38,
   "agents": [ "monica","fernanda"],
   "services": [ "AtendimentoCliente", "AtendimentoCorporativo" ],
   "events": [ "on_service_stats", "on_agent_stats", "on_alarm" ],
    "media_types": "all",
    "agent_stats": [ "total", "inbound", "abandoned",
                     "transferred", "calls_details"  ],
    "service_stats": [ "handled", "generated",
                     "received","received_abandoned", "handling",
                     "waiting" ],
    "alarms": [ { "name": "max_queue_abandomn_rate",
                  "value": 50,
                  "media_type": "voice" },
                { "name": "max_call_time",
                   "value": 1800,
                   "media_type": "chat" },
     "expires": 3600,
     "update_interval": 10
    }
}
```

Exemplo em XML:

```
 <?xml version="1.0" encoding="UTF-8"?>
```

```
<request>
 <subscription>
  <channel_id>38</channel_id>
  <agents>
   <member>monica</member>
   <member>fernanda</member>
  </agents>
  <services>
   <member>AtendimentoCliente</member>
   <member>AtendimentoCorporativo</member>
  </services>
  <events>
   <member>on_service_stats</member>
   <member>on_agent_stats</member>
   <member>on_alarm</member>
  </events>
  <media_types>all</media_types>
  <agent_stats>
   <member>total</member>
   <member>inbound</member>
   <member>abandoned</member>
   <member>transferred</member>
   <member>calls_details</member>
  </agent_stats>
  <service_stats>
```

```
    <member>handled</member>
    <member>generated</member>
    <member>received</member>
    <member>received_abandoned</member>
    <member>handling</member>
    <member>waiting</member>
   </service_stats>
   <alarms>
     <member>
       <name>max_queue_abandomn_rate</name>
       <value>50</value>
       <media_type>voice</media_type>
     </member>
       <name>max_call_time</name>
       <value>1800</value>
       <media_type>chat</media_type>
     <member>
     </member>
   </alarms>
   <expires>3600</expires>
   <update_interval>10</update_interval>
  </subscription>
</request>
```

Descrição do exemplo:

A aplicação assina a recepção de eventos de estatísticas de **Agentes** (on_agent_stats), **Serviços** (on_service_stats) e **Alarmes** (on_alarm) que ocorram para os agentes **monica** ou **fernanda** em qualquer mídia, para os serviços **AtendimentoCliente** ou **AtendimentoCorporativo**.

As estatísticas monitoradas são **total**, **inbound**, **abandoned**, **transferred** e **calls_details** para os agentes, e **handled**, **generated**, **received**, **received_abandoned**, **handling** e **waiting** para os serviços.

Os alarmes monitorados são **max_queue_abandomn_rate** para a mídia de voz, quando o valor ultrapassar 50, e **max_call_time** para a mídia *Chat,* quando o valor ultrapassar 1800.

A princípio, a assinatura permanecerá válida por **3600 segundos** (1 hora), caso não seja feita renovação da assinatura. Isso significa que durante esse período, o **canal 38**, estando aberto, receberá os eventos de operação monitorados contendo as estatísticas desejadas **no intervalo de 10 segundos** e, também, os eventos de controle.

Resposta ao Comando de Assinatura

A resposta para a assinatura de eventos possui o seguinte formato:

RESPONSE DATA em JSON:

```
{
    "response": {
        "command": "subscribe",
        "id": identificador_da_assinatura,
        "status": status,
```

```
        "error_type": tipo_de_erro,
        "error_items": listas_de_items_com_erro
    }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0"encoding="UTF-8"?>
 <response>
  <command> subscribe </command>
  <id> identificador_da_assinatura </id>
  <status> status </status>
  <error_type> tipo_de_erro </error_type>
  <error_items> listas_de_itens_com_erro </error_items>
 </response>
```

Onde:

- **Identificador_da_assinatura:** identificador numérico único para a assinatura;
- **Status:** estado da execução do comando (pode ser **ok** em caso de sucesso, ou **error** em caso de erro);
- **tipo_de_erro:** enviado apenas quando *status* for error, e pode ser:
  - **empty_data:** enviado quando o pacote de dados estiver vazio;
  - **invalid_data**: enviado quando o pacote de dados estiver em formato inválido;

- **not_found**: enviado em resposta a um comando de atualização com campo de id inexistente no servidor. Pode ocorrer no caso da tentativa de atualização de uma assinatura que já expirou;
- **invalid_item**: enviado quando o pacote de dados contiver algum item inválido (como o nome de um agente, o nome de um serviço, o nome de um evento ou o nome de um tipo de mídia);
- **no_license:** enviado quando não há licenças disponíveis em número suficiente para atender a assinatura.

▪ **lista_de_itens_com_erro:** enviado apenas quando o status for **error** e o **tipo_de_erro** for **invalid_item**. O conteúdo de **error_items** será uma lista de todos os itens que apresentaram erro na assinatura, separados nas **tags agents**, **services**, **events**, **media_types**, **agent_stats**, **service_stats e alarms.**

Exemplo de Resposta para o Caso de Sucesso

Exemplo em JSON:

```
{
    "response": {
        "command": "subscribe",
        "id": 38,
        "status": "ok"
    }
}
```

Exemplo em XML:

```
<?xml version="1.0"encoding="UTF-8"?>
```

```
<response>
 <command> subscribe </command>
 <id> 38 </id>
 <status> ok </status>
</response>
```

NOTA

*Caso haja algum erro sintático será retornado o erro HTTP 400 Bad Request. Caso o analisador sintático do **Interact CTI** consiga identificar a posição em que há um erro de sintaxe, será enviado um texto indicando a posição do pacote enviado em que foi detectado o erro (tanto para XML como para JSON).*

ALTERAÇÃO E RENOVAÇÃO DE ASSINATURA (COMANDO SUBSCRIBE)

Uma assinatura pode ser alterada com o envio de novo comando de assinatura passando o código identificador da assinatura (**id**). A assinatura anterior será cancelada e a nova ficará ativa em seu lugar. Para a renovação da assinatura basta que sejam enviados os mesmos parâmetros da assinatura original adicionados do código identificador da assinatura (**id**). O tempo de expiração será reiniciado para o valor que for informado no campo **expires**. Todos os parâmetros da assinatura podem ser alterados, inclusive o **número do canal** para envio de eventos.

CANCELAMENTO DE ASSINATURA (COMANDO UNSUBSCRIBE)

Uma assinatura poderá ser cancelada ao se enviar um comando de **unsubscribe** informando-se o código identificador da assinatura (**id**). Havendo assinatura com o **id** informado, ela será cancelada. Caso o id não exista, será retornado um erro **not found**.

Método HTTP POST

interact_cti/v1.0/supervisor/monitor/unsubscribe?format=[xml | json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.
POST DATA em JSON:

```
{
   “id”: identificador_da_assinatura
}
```

POST DATA em XML:

```
<request>
  <id>identificador_da_assinatura</id>
</request>
```

Onde:

- **identificador_da_assinatura:** número idenficador da assinatura recebido na resposta de um comando de assinatura (subscribe) bem sucedido.

ABERTURA DE CANAL PARA RECEPÇÃO DE EVENTOS (COMANDO OPEN_CHANNEL)

A recepção de eventos será sempre realizada por meio de c anais. Um canal é uma conexão HTTP persistente, que deverá ser aberta por meio do comando **open_channel**.

Na abertura do canal deverá ser informado um identificador numérico (**channel_id**) e um formato para a recepção dos dados (**XML** ou **JSQN**).

Cada canal possui um identificador único (**channel_id**). Este identificador deve ser informado como um parâmetro passado no comando **open_channel** ou, se não informado, será atribuído automaticamente pelo **Interact_CTI**. Na abertura do canal deverá ser informado um identificador numérico (**channel_id**) e um formato para a recepção dos dados (**XML** ou **JSQN**).

Cada canal possui um identificador único (**channel_id**). Este identificador deve ser informado como um parâmetro passado no comando **open_channel** ou, se não informado, erá atribuído automaticamente pelo **Interact CTI**. Em qualquer caso o **channel_id** é indicado no evento **on_open_channel**.

Na abertura do canal deverá ser informado um formato para a recepção dos dados (**XML** ou **JSQN**).

O identificador do canal deverá ser um valor na faixa entre 1 e 9999 e poderá ser associado a várias assinaturas.

Um canal é fechado automaticamente e depois de um minuto sem alguma assinatura associada. Isso vale, também, para o caso de se abrir um canal sem nenhuma assinatura associada. Ele permanecerá aberto por um minuto e, caso nenhuma assinatura seja feita nesse intervalo, ele será automaticamente fechado.

Método HTTP POST

interact_cti/v1.0/supervisor/monitor/open_channel?channel_id=nnnn&format=[xml | json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
   “channel_id”: identificador_do_canal
}
```

POST DATA em XML:

```
<request>
  <channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

- **identificador_do_canal**: número idenficador do canal que se deseja abrir.

FECHAMENTO DE CANAIS DE RECEPÇÃO DE EVENTOS (COMANDO CLOSE_CHANNEL)

Um canal pode ser fechado por meio do envio do comando **close_channel**, informando-se o seu identificador numérico (**channel_id**) e um formato para a recepção dos dados (XML ou JSON).

Ressalta-se que um canal será automaticamente fechado depois de 1 minuto sem alguma assinatura associada, independente do envio do comando **close_channel**.

Método HTTP POST

interact_cti/v1.0/supervisor/monitor/close_channel?format=[xml| json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
   “channel_id”: identificador_do_canal
}
```

POST DATA em XML:

```
<request>
  <channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

- **identificador_do_canal**: número idenficador do canal que se deseja fechar.

MANUTENÇÃO DO CANAL ABERTO

Para que uma conexão HTTP se mantenha de forma persistente é necessário que haja tráfego de pacotes desde o servidor até o cliente (autor da requisição) com certa frequência, caso contrário, a conexão será derrubada por *time-out* (que pode ocorrer tanto em um *browser* como em um servidor intermediário, entre o servidor do **Interact CTI** e a aplicação cliente).

O **Interact CTI** enviará um pacote de dados vazio apenas para sinalizar a atividade do canal e manter a conexão estabelecida.

O pacote de “heart beat” possui o seguinte formato: data: {}

RECEPÇÃO DOS EVENTOS PELO CANAL

Os eventos serão enviados dentro do padrão de Server Send Events da W3C (**http://www.w3.org/TR/eventsource**).

No **Interact CTI**, cada evento será enviado em um pacote de dados no seguinte formato:

```
event: nome_do_evento
data: dados_do_evento
```

Onde:

- **event**: nome do evento.

- Para controle do estado do canal:
  - on_open_channel.
  - on_close_channel.
- Para controle da assinatura:
  - on_subscription_expire.
  - on_subscription_expire_warning.
- Para controle do estado do servidor **Interact**:
  - on_server_state_change.
- Para o recurso **supervisor** será um dos seguintes eventos (eventos operacionais que podem ser assinados):
  - on_agent_stats.
  - on_service_stats.
  - on_alarm.

EVENTOS DE SUPERVISÃO

Eventos de supervisão são eventos relacionados à monitoração de agentes ou serviços e que podem ser roteados pelo **Interact CTI**. Esse tipo de evento pode ser assinado, ou seja, só será recebido se o integrador desejar. A seguir é apresentada a relação desses eventos:

Evento	Descrição
on_service_stats	Enviado quando ocorre mudança nas estatísticas relacionadas aos serviços.

on_agent_stats	Enviado quando ocorre mudança nas estatísticas relacionadas aos agentes.
on_alarm	Enviado quando uma condição de alarme é detectada.

Os eventos relacionados a estatísticas são enviados respeitando o intervalo de tempo definido na assinatura (**update_interval**). As estatísticas enviadas serão as variáveis que foram assinadas e que sofreram alteração em seu valor desde o último envio. No primeiro envio serão inseridas todas as variáveis selecionadas na assinatura.

Evento on_service_stats

Em JSON

```
{
  "services": [ {
                  "name": nome_do_servico1,
                  "calls_details": detalhes_das_chamadas,
                  "stats": [
                            { "media_type": nome_da_midiaA,
                               estatistica1 : valor,
                                 ...
                               estatistica2 : valor
                            },
                            ...
                            { "media_type": nome_da_midiaB,
```

```
                                    estatistica1 : valor,
                                      ...
                                    estatistica2 : valor,
                                  }
                                ]
                    },
                    ...
                    {
                      "name": nome_do_servico2,
                      ...
                    }
                  ]
}
```

Em XML

```
<services>
   <member>
     <name> nome_do_servico1 </name>
     <calls_details> detalhes_das_chamadas </calls_details>
     <stats>
       <member>
         <media_type> nome_da_midiaA </media_type>
         <estatistica1> valor </estatistica1>
           ...
```

```
        <estatistica2> valor </estatistica1>
      </member>
      ...
      <member>
        <media_type> nome_da_midiaB </media_type>
        <estatistica1> valor </estatistica1>
          ...
        <estatistica2> valor </estatistica1>
      </member>
    </stats>
  </member>
  ...
  <member>
    <name> nome_do_servico2 </name>
     ...
  </member>
</services>
```

Onde:

- **nome_do_servicoX**: nome do serviço que foi assinado para monitoração de estatísticas de serviço;
- **nome_da_midiaX**: nome da mídia que foi assinada;
- **estatisticaX**: nome da variável estatística que foi assinada;
- **detalhes_das_chamadas:** array com os seguintes detalhes de cada chamada em andamento no serviço:

Em JSON

```
{ "agent": nome_do_agente,
  "service": nome_do_servico,
  "media_type": nome_da_midia,
  "interlocutor": identificador_do_interlocutor,
  "call_id": identificador_da_chamada,
  "duration": duracao_da_chamada,
  "call_state": estado_da_chamada,
  "call_type":tipo_da_chamada,
  "call_source": origem_da_chamada,
  "consultation_call_id": identificador_da_chamada_de_consulta,
  "entry_address": endereco_de_entrada_email,
  "associated_data": dados_associados,
  "profile": perfil_da_chamada,
  "priority": prioridade_da_chamada,
  "queued": chamada_em_fila
}
```

Em XML

```
<request>
{ <agent> nome_do_agente </agent>
  <service> nome_do_servico </service>
  <media_type> nome_da_midia </media_type>
```

```
  <interlocutor> identificador_do_interlocutor </interlocutor>
  <call_id> identificador_da_chamada </call_id>
  <duration> duracao_da_chamada </duration>
  <call_state> estado_da_chamada </call_state>
  <call_type> tipo_da_chamada </call_type>
  <call_source> origem_da_chamada </call_source>
  <consultation_call_id> identificador_da_chamada_de_consulta
</consultation_call_id>
  <entry_address> endereco_de_entrada_email </entry_address>
  <associated_data> dados_associados </associated_data>
  <profile> perfil_da_chamada </profile>
  <priority> prioridade_da_chamada </priority>
  <queued> chamada_em_fila </queued>
</request>
```

Onde:

- **nome_do_agente**: nome do agente que atende a chamada;
- **nome_do_servico**: nome do serviço pelo qual a chamada chegou ao agente;
- **nome_da_midia:** nome do tipo de mídia da chamada (“voice”, “email” ou “chat”);
- **identificador_do_interlocutor:** identificador do interlocutor (pode ser um número de telefone para chamadas de voz, um endereço de *e-mail* para mensagens de *e-mail* ou um nome para um cliente de *Chat*);

- **identificador_da_chamada**: identificador único da chamada (inclui a data de geração da chamada) e pode ser utilizado como chave para integração com outros produtos da empresa ecruzamento com bancos de dados;
- **duracao_da_chamada**: duração da chamada (em segundos);
- **estado_da_chamada:** estado da chamada, o qual pode ser *busy* para chamada que já foi encaminhada a um agente ou *queued* para chamada em fila.
- **tipo_da_chamada:** tipo da chamada, o qual pode ser:
  - *inbound*: receptiva (entrante);
  - *outbound*: ativa (sainte);
  - *unknown*: desconhecido.
- **origem_da_chamada:** origem da chamada, a qual pode ser:
  - *internal*: interna;
  - *external*: externa;
  - *internal consultation*: interna de consulta;
  - *external consultation*: externa de consulta;
  - *internal conference*: interna de conferência;
  - *external conference*: externa de conferência;
  - *internal callback*: interna de *Callback*;
  - *external callback*: externa de *Callback*;
  - *unknown*: desconhecida.
- **identificador_da_chamada_de_consulta**: identificador da chamada em consulta (esse campo somente será enviado caso exista uma chamada em consulta associada).Obs.: Esse campo não é enviado no caso de consulta em mídia diferente (Ex.: chamada de voz e consulta via *Chat*);

- **endereco_de_entrada_de_email**: endereço de entrada de *e-mails* configurado para o serviço. Esse campo apenas é enviado no caso da chamada ser de *e-mail;*
- **dados_associados** exibe os dados associados à chamada;
- **perfil_da_chamada:** é o perfil em que se encontra a chamada (somente é apresentado para detalhes de chamadas em estatísticas de serviço);
- **prioridade_da_chamada:** prioridade atual da chamada (só é apresentada para detalhes de chamadas em estatísticas de serviço);
- **chamada_em_fila:** pode ser *true* ou *false*. Se for *true* indica que a chamada está em fila e, se for *false*, indica que está em atendimento ou reservada para atendimento (somente é apresentada para detalhes de chamadas em estatísticas de serviço).

Evento on_agent_stats

Em JSON

```
{
  "agents": [ {
                "name": nome_do_agente1,
                "states_details": dados_dos_estados,
                "calls_details": detalhes_das_chamadas,
                "services": estatisticas_de_servico
              },
              ...
              {
```

```
                "name": nome_do_agente2,
                  ...
               }
            ]
}
```

Em XML

```
<agents>
  <member>
    <name> nome_do_agente1 </name>
    <states_details> dados_dos_estados </states_details>
    <calls_details> detalhes_das_chamadas </calls_details>
    <services> estatisticas_de_servico </services>
  </member>
  ...
  <member>
    <name> nome_do_agente2 </name>
    ...
  </member>
</agents>
```

Onde:

- **nome_do_agente**: nome do agente que atende a chamada;

- **detalhes_das_chamadas**: array com detalhes de cada chamada que está atendendo (formato conforme apresentado para **on_service_stats**);
- **estatisticas_de_servico**: estrutura de informações da tag **services** (formato conforme apresentado para **on_service_stats** sem a *tag calls_details*);
- **dados_dos_estados**: array de objetos com as seguintes informações estatísticas de estados (modos) do agente:

Em JSON

```
{ "id" : identificador_do_estado,
  "name": nome_do_estado,
  "trigger": gatilho_da_pausa,
  "times": entradas_no_estado,
  "total_time": tempo_total_no_estado
}
```

Em XML

```
  <id> identificador_do_estado </id>
  <name> nome_do_estado </name>
  <trigger> gatilho_da_pausa </trigger>
  <times> entradas_no_estado </times>
  <total_time> tempo_total_no_estado </total_time>
```

Onde:

- **identificador_do_estado**: identificador do estado (equivale ao atributo **state_id** enviado nos eventos do recurso **agent**) em que se encontra um agente;
- **nome_do_estado**: nome do estado (equivale ao atributo **state** enviado nos eventos do recurso **agent**) em que se encontra um agente;
- **gatilho_da_pausa**: identifica como a pausa foi disparada. Pode ser manual (selecionada manualmente por agente ou supervisor), *scheduled* (pausa automática configurada no sistema) ou *system* (pausa disparada pelo sistema). Este atributo somente é enviado se o agente estiver em alguma pausa;
- **entradas_no_estado**: quantidade de vezes que o agente entrou em um determinado estado (desde que suas estatísticas foram iniciadas);
- **tempo_total_no_estado**: contabiliza o tempo total (somando as várias entradas) que um agente esteve em um determinado estado (desde que suas estatísticas foram iniciadas).

Exemplos de evento on_agent_stats

EVENT DATA Em JSON


```
{
    "agents": [{
     "name": "monica",
     "states_details": [{
     "id": 3,
     "name": "ready_free",
     "times": 10,
     "total_time": 1823
```

```
 }, {
 "id": 4,
 "name": "ready_service",
 "times": 0,
 "total_time": 4710,
 "pre_pause": {
  "times": 1,
  "total_time": 571
 }
}],
"calls_details": [{
 "service": "AtendimentoCliente",
 "media_type": "voice",
 "interlocutor": 99610418,
 "call_id": "31/10/13 13:31:13 013A 8013/00 ",
 "duration": 132,
 "call_state": "busy",
 "call_type": "inbound",
 "call_source": "external"
}, {
 "service": "AtendimentoCliente",
 "media_type": "chat",
...
}],
"services": [{
```

```
      "name": "AtendimentoCliente",
      "stats": [{
      "media_type": "voice",
      "total": 13,
      "inbound": 12,
      "abandoned": 1,
      "transfered": 2
     }, {
      "media_type": "chat",
    ...
     }
    ...
    ]
  }, {
   "name": "AtendimentoCorporativo",
   "stats": ...
  }
  ...
  ]
},
...
{
 "name": "fernanda",
 ...
}
```

```
  ]
}
```

EVENT DATA Em XML

```
<agents>
  <member>
    <name>monica</name>
    <states_details>
      <member>
         <id>3</id>
         <name>ready_free</name>
         <times>10</times>
         <total_time>1823</total_time>
      </member>
      <member>
         <id>4</id>
         <name>ready_service</name>
         <times>0</times>
         <total_time>4710</total_time>
          <pre_pause>
        <times>1</times>
        <total_time>571</total_time>
        <\pre_pause>
      </member>
    </states_details>
```

```
<call_details>
  <member>
    <service>AtendimentoCliente</service>
    <media_type>voice</media_type>
    <interlocutor>99610418</interlocutor>
    <call_id>31/10/13 13:31:13 013A 8013/00</call_id>
    <duration>132</duration>
    <call_state>busy</call_state>
    <call_type>inbound</call_type>
    <call_source>external</call_source>
  </member>
  <member>
    <service>AtendimentoCorporativo</service>
    <media_type>chat</media_type>
     ...
 </member>
</call_details>
<services>
  <member>
     <name>AtendimentoCliente</name>
     <stats>
        <member>
          <media_type>voice</media_type>
          <total>13</total>
          <inbound>12</inbound>
```

```
                <abandoned>1</abandoned>
                <transfered>2</transfered>
              </member>
              <member>
                <media_type>chat</media_type>
                ...
              </member>
              ...
            </stats>
          </member>
          <member>
            <name>AtendimentoCorporativo</name>
            <stats>
              ...
            </stats>
          </member>
        </services>
      </member>
      <member>
        <name>fernanda</name>
        ...
      </member>
    </agents>
```

NOTA

As informações de pré-pausa somente serão enviadas nos estados ready_service de id 4 e ready_private de id 5.

Evento on_alarm

O evento de **on_alarm** apresenta todos os alarmes ativos, identificados pelo nome e com a lista de agentes ou serviços para os quais está ativo.

Em JSON

```
{
   "alarms": [  {
                        "name": nome_do_alarme1,
                        “active”: lista_de_alarmes_ativos1
                },
                {
                        "name": nome_do_alarme2,
                        “active”: lista_de_alarmes_ativos2
                },
                ...
                {
                        "name": nome_do_alarmeN,
                        “active”: lista_de_alarmes_ativosN
                }
```

```
        ]
}
```

Em XML

```
<alarms>
   <member>
         <name>  nome_do_alarme1 </name>
         <active>  lista_de_alarmes_ativos1 </active>
   </member>
   <member>
         <name>  nome_do_alarme2 </name>
         <active>  lista_de_alarmes_ativos2 </active>
   </member>
   …
   <member>
         <name>  nome_do_alarmeN </name>
         <active>  lista_de_alarmes_ativosN </active>
   </member>
</alarms>
```

Onde:

- **nome_do_alarmeX:** nome do alarme assinado;
- **lista_de_alarmes_ativosX**: lista com o seguinte formato:

Em JSON (para alarmes de serviço):

```
[
   {      “service”: nome_do_servico1,
          “value”: valor_do_parametro1,
          “reference”: valor_de_referencia1
   }
   ...
   {      “service”: nome_do_servicoN,
          “value”: valor_do_parametroN,
          “reference”: valor_de_referenciaN
   }
]
```

Em XML (para alarmes de serviço):

```
<active>
   <member>
          <service> nome_do_servico1 </service>
          <value> valor_do_parametro1 </value>
          <reference> valor_de_referencia1 </reference>
   </member>
   ...
   <member>
          <service> nome_do_servicoN </service>
          <value> valor_do_parametroN </value>
          <reference> valor_de_referenciaN </reference>
```

```
   </member>
</active>
```

Em JSON (para alarmes de agente):

```
[
   {      “agent”: nome_do_agente1,
          “value”: valor_do_parametro1,
          “reference”: valor_de_referencia1
   }
   ...
   {      “agent”: nome_do_agenteN,
          “value”: valor_do_parametroN,
          “reference”: valor_de_referenciaN
   }
]
```

Em XML (para alarmes de serviço):

```
<active>
   <member>
          <agent> nome_do_agente1 </agent>
          <value> valor_do_parametro1 </value>
          <reference> valor_de_referencia1 </reference>
   </member>
   ...
   <member>
```

```
            <agent> nome_do_agenteN </agente>
            <value> valor_do_parametroN </value>
            <reference> valor_de_referenciaN </reference>
    </member>
</active>
```

Onde:

- **nome_do_servicoX:** nome do serviço para o qual o alarme está ativo;
- **nome_do_agenteX:** nome do agente para o qual o alarme está ativo;
- **valor_do_parametroX:** valor atual do parâmetro para o serviço ou agente;
- **valor_de_referenciaX:** valor programado na assinatura para o alarme.

Exemplos de evento on_alarm

Exemplo em JSON

```
{
    "alarms": [
            {
                    "name": "max_queue_length",
                    "active": [
                                    {
                                            "service": "AtendimentoCliente",
                                            "value": 11,
                                            "reference": 10
                                    }
                            ]
```

```
            },
            {
                    "name": "max_fail_pause_quantity",
                    "active": [
                                {
                                        "agent": "monica",
                                        "value": 6,
                                        "reference": 5
                                }
                            ]
            }
    ]
}
```

Exemplo em XML

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <alarms>
      <member>
         <name>max_queue_length</name>
         <active>
            <member>
               <service>AtendimentoCliente</service>
               <value>11</value>
               <reference>10</reference>
```

```
                </member>
             </active>
          </member>
          <member>
             <name>max_fail_pause_quantity</name>
             <active>
                <member>
                   <agent>monica</agent>
                   <value>6</value>
                   <reference>5</reference>
                </member>
             </active>
          </member>
       </alarms>
    </data>
```

Estatísticas Monitoráveis de Serviço

Nome	Descrição
handled	Quantidade de chamadas atendidas.
generated	Quantidade de chamadas geradas.
received	Quantidade de chamadas recebidas.
immediate_handled	Quantidade de chamadas atendidas imediatamente.
handled_awt	Quantidade de chamadas Atendidas TME (atendidas dentro do tempo médio de espera).
dequeued	Quantidade de chamadas que passaram pela fila.
overflowed	Quantidade total de chamadas transbordadas.
overflowed_by_time	Quantidade de chamadas transbordadas por tempo de fila.
overflowed_by_size	Quantidade de chamadas transbordadas por tamanho de fila.
out_operation	Quantidade de chamadas recebidas fora de expediente.
before_queue_abandoned	Quantidade de chamadas abandonadas antes da fila.
in_queue_abandoned	Quantidade de chamadas abandonas na fila.

Nome	Descrição
branch_abandoned	Quantidade de chamadas abandonadas no ramal.
agente_abandoned	Quantidade de chamadas rejeitadas ou derrubadas em menos de dois segundos pelo agente
hold	Quantidade de chamadas em hold.
queued	Quantidade de chamadas em fila.
abandoned	Quantidade total de chamadas abandonadas.
consultation	Quantidade de consultas em andamento.
no_response	Quantidade de atendimentos sem resposta.
generated_abandoned	Quantidade de chamadas geradas e abandonadas.
received_abandoned	Quantidade de chamadas recebidas e abandonadas.
after_call_amnt	Quantidade de chamadas em pós-atendimento.
talk_time	Tempo total em conversação.
queue_time	Tempo total em fila.
ringing_time	Tempo total no estado "tocando".
calling_time	Tempo total no estado "chamando".
before_queue_abandoned_time	Tempo de abandono antes da fila.
in_queue_abandoned_time	Tempo de abandono na fila.
hold_time	Tempo em "hold".

Nome	Descrição
after_call_work_time	Tempo em pós-atendimento.
consult_time	Tempo em consulta.
overflowed_call_time	Tempo total de chamadas que transbordaram.
longest_queue_time	Tempo da chamada que ficou mais tempo em fila.
longest_abandoned_call_time	Maior tempo de chamada abandonada.
longest_handle_time	Maior tempo de chamada atendida.
longest_overflowed_call_time	Chamada de maior duração que transbordou.
longest_abandoned_in_queue_time	Chamada de maior duração abandonada na fila.
abandoned_time	Tempo médio das chamadas abandonadas.
generating	Total de chamadas sendo geradas no momento.
handling	Total de chamadas sendo atendidas no momento.
waiting	Total de chamadas aguardando no momento.
classifying	Total de chamadas em classificação no momento.
before_handling	Total de chamadas em pré-atendimento no momento.
after_call_working	Total de chamadas em pós-atendimento no momento.
terminating	Total de chamadas em finalização no momento.

Nome	Descrição
calls_details	Detalhes das chamadas em fila e em atendimento no(s) serviço(s).
handled_profile_N	Total de chamadas atendidas no perfil N (onde N é um número entre 1 e 5 que identifica o respectivo perfil).
abandoned_profile_N	Total de chamadas abandonadas no perfil N (onde N é um número entre 1 e 5 que identifica o respectivo perfil).
in_queue_profile_N	Total de chamadas em fila no perfil N (onde N é um número entre 1 e 5 que identifica o respectivo perfil).
callback_handled	Quantidade de chamadas de *Callback* atendidas (exclusivo para serviços repetitivos).
callback_rejected	Quantidade de chamadas de *Callback* atendidas (exclusivo para serviços repetitivos).
outbound_before_handle_time	Tempo médio para atender chamadas geradas (exclusivo para serviços ativos).
outbound_handle_time	Tempo médio de chamadas geradas (exclusivo para serviços ativos).

NOTAS

1) *As estatísticas **handled**, **generated**, **received**, **immediate_handled**, **handled_awt**, **abandoned**, **handling**, **generating**, **handled_profile_N** e **abandoned_profile_N** são variáveis estatísticas gerais, ou seja, podem ser solicitadas para um conjunto de serviços consumindo uma única licença de supervisão .*
2) *As demais estatísticas são consideradas variáveis estatísticas de detalhes e implicam no consumo de uma licença de supervisão por serviço que for assinado.*

Estatísticas Monitoráveis de Agentes

Nome	Descrição
consultation	Quantidade de consultas.
abandoned	Quantidade de chamadas abandonadas.
hold	Quantidade de chamadas em hold.
transferred	Quantidade de chamadas transferidas.
after_call_work	Quantidade de chamadas em pós-atendimento.
outbound_abandoned	Quantidade de chamadas geradadas abandonadas.
inbound_abandoned	Quantidade de chamadas recebidas abandonadas.
outbound	Quantidade de chamadas recebidas.
inbound	Quantidade de chamadas geradas.
total	Quantidade de chamadas iniciadas (geradas + recebidas).

Nome	Descrição
handle_time	Tempo em atendimento.
talk_time	Tempo em conversação (Tempo em atendimento - Tempo em hold + Tempo em consultas).
consult_time	Tempo em consultas.
hold_time	Tempo em hold.
after_call_work_time	Tempo em pós-atendimento.
calls_details	Detalhes das chamadas em andamento no(s) agente(s).
states_details	Detalhes dos estados (modos) do(s) agente(s).

NOTAS

*1) As estatísticas **total**, **inbound** e **outbound** são variáveis estatísticas gerais, ou seja, podem ser solicitadas para um conjunto de agentes consumindo uma única licença de supervisão.*

2) As demais estatísticas são consideradas variáveis estatísticas de detalhes e implicam no consumo de uma licença de supervisão por agente que for assinado.

Alarmes Monitoráveis

Nome	Descrição
max_queue_abandomn_rate	Taxa máxima de abandono na fila (será gerado alarme quando a taxa superar o valor definido).

max_branch_abandomn_rate	Taxa máxima de abandono no ramal (será gerado alarme quando a taxa superar o valor definido).
max_queue_time	Tempo máximo em fila (será gerado alarme quando o tempo de uma chamada em fila superar o valor definido).
max_queue_time	Tempo máximo em fila (será gerado alarme quando o tempo de uma chamada em fila superar o valor definido).
max_queue_length	Número máximo de chamadas em fila (será gerado alarme quando o número de chamadas em fila superar o valor definido).
max_call_time	Tempo máximo de uma chamada (será gerado alarme quando o tempo toral de uma chamada superar o valor definido).
max_handle_time	Tempo máximo de atendimento uma chamada (será gerado alarme quando o tempo de atendimento de uma chamada superar o valor definido).
max_fail_pause_quantity	Número máximo de entradas em pausa por falha para agentes (será gerado alarme quando o número superar o valor definido). Neste número são somadas as entradas em todos os tipos de pausa por falha.
min_profile	Perfil mínimo para atendimento (será gerado alarme quando uma chamada atingir o valor do perfil definido).

8

COMANDOS DE SUPERVISÃO

Para o recurso **supervisor**, são disponibilizados os seguintes comandos:

- set_agent_state.
- agent_logout.
- reset_agent_stats.
- reset_service_stats.
- clear_agent_stats.
- clear_service_stats.
- get_user_grants.

Também é possivel enviar os seguintes comandos através do recurso **supervisor** (comandos disponíveis, também, para o recurso **agent**):

- get_agents_list.
- get_services_list.
- get_pause_type_list.
- get_classification_list.
- get_subscriptions.

ALTERAÇÃO DO ESTADO DE AGENTE (COMANDO SET_AGENT_STATE)

Esse comando altera o estado de um agente logado no **Interact** por meio do supervisor.

Método HTTP POST

interact_cti/v1.0/supervisor/set_agent_state?format=[json | xml]

POST DATA em JSON:

```
{
   "supervisor": login_do_supervisor,
   "password": senha_do_supervisor,
   "agent” login_do_agent,
   "state_id": id_do_novo_estado
 }
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <supervisor> login_do_supervisor </supervisor>
       <password> senha_do_supervisor </password>
       <agent> login_do_agent </agent>
       <state_id> id_do_novo_estado </state_id>
```

```
</request>
```

Onde:

- **login_do_supervisor:** nome de *login* de um supervisor cadastrado;
- **senha_do_supervisor:** senha de *login* do supervisor;
- **login_do_agent:** nome de *login* do agente que deverá ter seu estado alterado;
- **state_id**: identificador do novo estado do agente (modo). O modo pode ser um dos seguintes:
  - 2 – Operando (está logado, não pausado).
  - Os demais valores possíveis são os retornados pelo comando **get_pause_type_list**, que relaciona os motivos de pausa configurados no **Interact**.

LOGOUT DE AGENTE (COMANDO AGENT_LOGOUT)

Esse comando desloga um agente por meio do supervisor.

Método HTTP POST

interact_cti/v1.0/supervisor/agent_logout?format=[json | xml]

POST DATA em JSON:

```
{
   "supervisor": login_do_supervisor,
   "password": senha_do_supervisor,
```

```
   "agent” login_do_agent,
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <supervisor> login_do_supervisor </supervisor>
       <password> senha_do_supervisor </password>
       <agent> login_do_agent </agent>
</request>
```

Onde:

- **login_do_supervisor:** nome de *login* de um supervisor cadastrado;
- **senha_do_supervisor:** senha de *login* do supervisor;
- **login_do_agent:** nome de *login* do agente que deverá ser deslogado;

RESET DE ESTATÍSTICAS DE AGENTE (COMANDO RESET_AGENT_STATS)

Esse comando reinicia (zera) as estatísticas do agente no **Interact** e limpa o cache do **Interact CTI**.

Método HTTP POST

interact_cti/v1.0/supervisor/reset_agent_stats?format=[xml | json]

POST DATA em JSON:

```
{
   "agent": nome_do_agente
 }
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <agent>nome_do_agente</agent>
</request>
```

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

NOTA

*O comando **reset_agent_stats** somente resetará as estatísticas do agente caso ele esteja logado no **Interact**. Caso o agente esteja deslogado, a resposta ao comando será **error**.*

Resposta ao Comando reset_agent_stats

A resposta para o comando será conforme os modelos a seguir:

RESPONSE em JSON:

```
{
    "response": {
        "command": "reset_agent_stats",
        "status": "ok"
    }
}
```

RESPONSE em XML:

```
<response>
 <command>reset_agent_stats</command>
 <status>ok</status>
</response>
```

RESET DE ESTATÍSTICAS DE SERVIÇO (COMANDO RESET_SERVICE_STATS)

Esse comando reinicia (zera) as estatísticas do serviço no **Interact** e limpa o cache do **Interact CTI**.

Método HTTP POST

interact_cti/v1.0/supervisor/reset_service_stats?format=[xml | json]

POST DATA em JSON:

```
{
   "service": nome_do_servico
 }
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <service>nome_do_servico</service>
</request>
```

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

Resposta ao Comando reset_service_stats

A resposta para o comando será conforme os modelos a seguir:

RESPONSE em JSON:

```
{
    "response": {
        "command": "reset_service_stats",
```

```
        "status": "ok"
    }
}
```

RESPONSE em XML:

```
<response>
  <command>reset_service_stats</command>
  <status>ok</status>
</response>
```

LIMPEZA DE CACHE DE ESTATÍSTICAS DO AGENTE (COMANDO CLEAR_AGENT_STATS)

Esse comando limpa o cache de estatísticas de agentes para uma determinada assinatura.

Método HTTP POST

interact_cti/v1.0/supervisor/clear_agent_stats?format=[xml | json]

POST DATA em JSON:

```
{
  "subscription_id": id_da_assinatura
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
   <subscription_id>id_da_assinatura</subscription_id>
</request>
```

NOTA

O comando "clear_agent_stats" pode ser usado para forçar a geração de um evento "on_agent_stats".

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

Resposta ao Comando clear_agent_stats

A resposta para o comando será conforme os modelos a seguir:

RESPONSE em JSON:

```
{
    "response": {
        "command": "clear_agent_stats",
        "status": "ok"
    }
}
```

RESPONSE em XML:

```
<response>
 <command>clear_agent_stats</command>
 <status>ok</status>
</response>
```

LIMPEZA DE CACHE DE ESTATÍSTICAS DO SERVIÇO (COMANDO CLEAR_SERVICE_STATS)

Esse comando limpa o cache de estatísticas de serviços para uma determinada assinatura.

Método HTTP POST

interact_cti/v1.0/supervisor/clear_service_stats?format=[xml | json]

POST DATA em JSON:

```
 {
   "subscription_id": id_da_assinatura
 }
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <subscription_id>id_da_assinatura</subscription_id>
```

```
</request>
```

NOTA

O comando "clear_service_stats" pode ser usado para forçar a geração de um evento "on_service_stats".

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

Resposta ao Comando clear_service_stats

A resposta para o comando será conforme os modelos a seguir:

RESPONSE em JSON:

```
{
    "response": {
        "command": "clear_service_stats",
        "status": "ok"
    }
}
```

RESPONSE em XML:

```
<response>
 <command>clear_service_stats</command>
 <status>ok</status>
```

```
</response>
```

CONSULTA DE PERMISSÕES DO USUÁRIO (COMANDO GET_USER_GRANTS)

Esse comando retorna os privilégios de um determinado usuário.

Método HTTP GET

interact_cti/v1.0/supervisor/get_user_grants?name=nome_do_usuario&format=[xml | json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

Resposta ao comando get_user_grants

A resposta para o comando será conforme os modelos a seguir:

RESPONSE em JSON:

```
{
    "response":
    {
        "command": "get_user_grants",
        "status": "ok"
```

```
        "name": nome_do_usuario,
        "grants": { "supervision":
lista_privilegios_de_supervisao,
                    "access": lista_de_privilegios_de_acesso,
                    "agent": lista_de_privilegios_de_agente,
                    "calls": lista_de_privilegios_de_chamadas
                  }
    }
}
```

RESPONSE em XML:

```
<response>
 <command>get_user_grants</command>
 <status>ok</status>
 <name>nome_do_usuario</name>
 <grants>
   <supervision>lista_privilegios_de_supervisao</supervision>
   <access>lista_privilegios_de_acesso</access>
   <agent>lista_de_privilegios_de_agente</agent>
   <calls>lista_de_privilegios_de_chamadas</media_type>
 </grants>
</response>
```

Onde:

- **nome_do_usuario:** nome do usuário (pode ser um agente ou um supervisor);

- **lista_privilegios_de_supervisao**: lista com privilégios de supervisão cadastrados para o usuário (a lista é uma array com os nomes dos privilégios). Os privilégios possíveis são os seguintes:

Privilégio	Descrição
search_calls	Pesquisa de chamadas.
manage_licenses	Licenças.
external_alarm_supervision	Supervisão externa de alarmes.
agents_supervision	Supervisão de agentes.
services_supervision	Supervisão de serviços.
reset_stats	Reset de estatísticas.
agent_send_message	Envio de mensagens para agente .
agent_intercalation	Intercalação de chamada do agente.
agent_monitoring	Monitoração de chamada do agente.
agent_change_state	Alteração do estado do agente.
all_agents_supervision	Supervisão de todos os agentes.
all_services_supervision	Supervisão de todos os serviços.
callback_supervision	Supervisão de *Callback*.
service_change_state	Alteração do estado de serviço.

- **lista_privilegios_de_acesso**: lista com privilégios de recursos de cadastro que o usuário pode acessar (a lista é uma array com os nomes dos privilégios). Os privilégios possíveis são os seguintes:

Privilégio	Descrição
alarms_full_access	Acesso completo ao cadastro de alarmes.
classifications_full_access	Acesso completo ao cadastro de classificações.
special_dates_full_access	Acesso completo ao cadastro de datas especiais.
messages_full_access	Acesso completo ao cadastro de mensagens.
pauses_full_access	Acesso completo ao cadastro de motivos de pausa.
services_full_access	Acesso completo ao cadastro de serviços.
skills_full_access	Acesso completo ao cadastro de *skills.*
teams_full_access	Acesso completo ao cadastro de times.
call_routing_full_access	Acesso completo ao cadastro de listas de fidelização e priorização.
alarms_read_access	Leitura do cadastro de alarmes.
classifications_read_access	Leitura do cadastro de classificações.
special_dates_read_access	Leitura do cadastro de datas especiais.
messages_read_access	Leitura do cadastro de mensagens.
pauses_read_access	Leitura do cadastro de motivos de pausa.
services_read_access	Leitura do cadastro de serviços.

Privilégio	Descrição
skills_read_access	Leitura do cadastro de *skills*.
teams_read_access	Leitura do cadastro de times.
call_routing_read_access	Leitura do cadastro de listas de fidelização e priorização.

- **lista_de_privilegios_de_agente:** lista com privilégios de agente cadastrados para o usuário (a lista é uma array com os nomes dos privilégios). Os privilégios possíveis são os seguintes:

Privilégio	Descrição
login	*Login*.
branch_transfer	Transferência para ramal.
send_message	Envio de mensagens.
call_history	Histórico de chamadas.
chat_call	Geração de chamadas de *Chat.*
view_email_attachments	Visualização de anexos de *e-mail.*
send_email_attachments	Envio de anexos de *e-mail.*
voice_inbound_config	Configuração de modo de atendimento de voz.
chat_inbound_config	Configuração de modo de atendimento de *Chat.*
email_inbound_config	Configuração de modo de atendimento de *e-mail.*

- **lista_de_privilegios_de_chamadas:** lista com privilégios de chamadas por mídia. Essa lista só é preenchida para agentes logados. Os privilégios possíveis são os seguintes:

Privilégio	Descrição
call_generate	Permissão de chamadas saintes.
call_outbound	Permissão de chamadas saintes externas.
call_receive	Permissão de chamadas entrantes.
call_inbound	Permissão de chamadas entrantes externas.
call_consultation	Permissão de chamadas de consulta.
call_transfer	Permissão de transferências de chamadas.
call_conference	Permissão para colocar chamadas em conferência.

9

GRAVADOR

O recurso Recorder do **Interact CTI** possibilita controlar gravações por meio de comandos REST.

O **Interact CTI** dispõe dos seguintes recursos de gravação:

- Consulta a sites.
- Comando de gravação sob demanda para as mídias Voz, *Chat* e *E-mail.*
- ids da gravação para manipulação posterior.
- Monitoração do *status* das gravações.
- Consulta da relação de gravações de um determinado dispositivo (Ramal ou Agente).
- Consulta dos dados de uma gravação.
- Associação de descrição à uma gravação.
- Controle da reprodução de gravação de voz em um ramal.

Os serviços disponibilizados pelo **Interact CTI** para gravação por meio de REST têm o seguinte formato geral:

interact_cti/v1.0/recorder/metodo?parametros[&format= (json|xml)]

Onde:

- **metodo:** nome da requisição (comando);
- **parametros**: conjunto de pares nome-valor na forma prm=valor, separados por & (Ecomercial -ampersand);
- **format:** define o formato desejado para a resposta, a qual pode ser no formato JSON (.json) ou no formato (.xml). Na ausência desse parâmetro será utilizado o mesmo formato informado no header Content-type.

Observações:

- Na ausência do header **Content-type** será utilizado **XML** por *default*.
- Na ausência de parâmetros e de formato, o sinal de interrogação após metodo é desnecessário.
- As requisições aos serviços serão sempre por meio de métodos HTTP GET ou POST.
- O método GET é utilizado em todas as requisições que não enviam apenas parâmetros.
- O método POST é utilizado em todas as requisições que enviam pacotes de dados (além dos parâmetros).
- Os parâmetros enviados na URL deverão ser codificados no padrão URL encode.
- Os dados nos pacotes POST deverão ser codificados em UTF-8.
- O acesso se dará na porta 8582 (valor *default* que pode ser alterado por configuração), conforme exemplo abaixo:
- `http://XXX.XXX.XXX.XXX:8582/interact_cti/v1.0/recorder/metodo`

Onde:

- **XXX.XXX.XXX.XXX:** endereço IP do servidor **Interact CTI**.

CONSULTA DE SITES (COMANDO GET_SITES)

O comando **get_sites** retorna o conjunto de sites de configurados no gravador (id e nome de cada site). O identificador do site é útil para os comandos de gravação sob demanda.

Método HTTP GET

interact_cti/v1.0/recorder/get_sites?operator=nome_do_operador&format=[xml | json]

Onde:

- **nome_do_operador**: nome de um usuário cadastrado com permissão de operador do gravador;
- **format (opcional):** formato desejado para a resposta. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

RESPONSE DATA em JSON:

```
{  "sites": [ { "id": id_1, "name": nome_1 },
              { "id": id_2, "name": nome_2 },
                ...
              { "id": id_n, "name": nome_n }
```

```
                ]
}
```

RESPONSE DATA em XML:

```
<sites>
  <member>
    <id>id_1</id>
    <name>name_1</name>
  </member>
  <member>
    <id>id_2</id>
    <name>name_2</name>
  </member>
  ...
  <member>
    <id>id_n</id>
    <name>name_n</name>
  </member>
</sites>
```

Onde:

- **id:** identificador do site;
- **name:** nome do site.

GRAVAÇÃO SOB DEMANDA

Gravação de única chamada (comandos record_start e record_stop)

Os comandos **record_start** e **record_stop** permitem controlar a gravação de uma chamada específica sem que haja programação de filtros no gravador.

A chamada precisa estar em andamento para a gravação ter efeito. No caso de voz, o áudio da chamada será gravado a partir do momento do envio do comando **record_start** e encerrado a partir do envio do comando **record_stop**. Para as demais mídias só é necessário o envio do comando **record_start**.

interact_cti/v1.0/recorder/ record_start?format=[xml | json]

POST DATA em JSON:

```
{  "call_id": identificador_da_chamada,
   "operator": nome_do_operador,
   "site": identificador_do_site
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <call_id>identificador_da_chamada</call_id>
       <operator>nome_do_operador</operator>
       <site>identificador_do_site</site>
</request>
```

interact_cti/v1.0/recorder/record_stop?format=[xml | json]

POST DATA em JSON:

```
{
     "call_id": identificador_da_chamada,
     "operator": nome_do_operador,
     "site": identificador_do_site
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
     <call_id>identificador_da_chamada</call_id>
     <operator>nome_do_operador</operator>
     <site>identificador_do_site</site>
</request>
```

Em caso de sucesso, a resposta recebida será:

Em JSON:

```
{
     "status": "ok",
     "entity_id": identificador_da_entidade
     "record_id": identificador_da_gravacao
}
```

Em XML:

```
<response>
     <status>ok</status>
     <entity_id> identificador_da_entidade </entidade>
     <record_id> identificador_da_gravacao </record_id>
</response>
```

Onde:

- **identificador_da_chamada:** chave primária da chamada;
- **identificador_da_entidade:** identificador da entidade gravada;
- **identificador_da_gravacao:** identificador da gravação em andamento;

NOTA

*Uma gravação em andamento poderá ser monitorada com o comando **get_status**.*

Configuração de Gravação sob Demanda (comando set_on_demand)

O comando **set_on_demand** permite informar agentes, serviços e mídias que serão gravados sem programação de filtros no gravador.

Método HTTP POST

interact_cti/v1.0/recorder/set_on_demand?operator=nome_do_operador&format=[xml | json]

POST DATA em JSON:

```
{ “agents”: lista_de_agentes,
  “services”: lista_de_servicos ,
  “media_types”: lista_de_mídias,
  “site”: identificador_do_site
}
```

POST DATA em XML:

```
<agents>lista_de_agentes</agents>
<services>lista_de_servicos</services>
<media_types>lista_de_midias</media_types>
<site>identificador_do_site</site>
```

Onde:

- **lista_de_agentes:** lista com os nomes dos agentes que serão gravados;
- **lista_de_servicos:** lista com os nomes dos serviços que serão gravados;
- **lista_de_midias:** lista com os nomes das mídias que serão gravadas;
- **nome_do_operador:** nome de um usuário cadastrado com permissão de operador do gravador;
- **identificador_do_site:** identficador do site do gravador;

- **format (opcional)**: formato desejado para a resposta. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

NOTAS

1) *Esse comando não é uma programação do gravador. Se um novo comando **set_on_demand** for enviado, um comando **set_on_demand** prévio perde o efeito.*
2) *Para interromper as gravações, basta enviar o mesmo comando com as listas de agentes, serviços e mídias vazias.*

ENVIO DE IDENTIFICADOR DA GRAVAÇÃO

No **Interact CTI** é gerado o evento **on_call_record** ao final da gravação, no qual são encaminhados os identificadores e entidade gravada e da gravação. Esse evento só será enviado no caso de gravações que possuam programação com filtros ou no caso de gravações sob demanda.

Para as gravações sob demanda, o **Id da gravação** será informado por meio do mesmo evento ***on_call_record*** adicionado do atributo ***mode=on_demand***. Para as gravações sinalizadas pelo **Interact** (com programação), serão gerados eventos ***on_call_record*** com o atributo ***mode=auto***.

MONITORAÇÃO DE ESTADO DAS GRAVAÇÕES (COMANDO GET_STATUS)

O comando **get_status** permite monitorar o estado de gravações em andamento.

Método HTTP GET

interact_cti/v1.0/recorder/get_status?operator=nome_do_operador&call_id=id_da_chamada&site=identificador_do_site&format=[xml | json]

Onde:

- **nome_do_operador:** nome de um usuário cadastrado com permissão de operador do gravador;
- **id_da_chamada:** chave primária da chamada;
- **identificador_do_site:** identificador do site onde se deseja buscar o estado da chamada.
- **format (opcional):** formato desejado para a resposta. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

RESPONSE DATA em JSON:

```
{
     "command": "get_status",
     "status": "ok" ,
     "record_status": status_da_gravacao,
     "last_transition": ultima_transicao,
     "entity_id": id_da_entidade,
     "record_id": id_da_gravacao,
     "server": servidor,
     "file": arquivo,
}
```

RESPONSE DATA em XML:

```
<command>get_status</command>
<status>ok</status>
<record_status>status_da_gravacao</record_status>
<last_transition>ultima_transicao</last_transition>
<entity_id>id_da_entidade</record_id>
<record_id>id_da_gravacao</record_id>
<server>servidor</server>
<file>arquivo</file>
```

Onde:

- **status_da_gravacao:** indica o estado da gravação;
- **ultima_transicao:** data e horário da última transição de estado;
- **id_da_entidade**: identificador da entidade gravada;
- **id_da_gravacao:** identificador da gravação;
- **servidor:** endereço IP do servidor onde está a gravação;
- **arquivo:** caminho completo do arquivo de gravação.

CONSULTA DA RELAÇÃO DE GRAVAÇÕES DE UM AGENTE (COMANDO GET_LIST)

O comando **get_list** permite consultar as gravações de um agente em uma determinada data, podendo ser filtrado por mídia e serviço.

Método HTTP GET

interact_cti/v1.0/recorder/get_list?agent=nome_do_agente&date=data_das_chamadas&operator=nome_do_operador[&start=registro_inicial&count=quantidade_de_registros&media_type=tipo_de_midia&service=nome_do_servico&direction=direcao_de_ordenacao&call_direction=direcao_da_chamada&format= xml | json]

Onde:

- **nome_do_agente:** nome do agente que teve as chamadas gravadas;
- **tipo_de_midia:** nome da midia para a qual se deseja filtrar a lista de gravações;
- **data_das_chamadas**: data das chamadas gravadas (a consulta retorna informações sobre as gravações de chamadas que iniciaram nessa data);
- **nome_do_servico:** nome do serviço para o qual se deseja filtrar a lista de gravações;
- **nome_do_operador:** nome de um usuário cadastrado com permissão de operador do gravador;
- **format (opcional):** formato desejado para a resposta. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.
- **registro_inicial:** número do registro inicial útil para realizar paginação em conjunto com o parâmetro **count** (se não for informado, assume valor 1);
- **quantidade_de_registros**: quantidade máxima de registros que a consulta deve retornar (se não for informado, assume o valor 10);
- **direcao_de_ordenacao**: a ordenação será sempre pela data da chamada. Será crescente se informado “asc” nesse parâmetro ou decrescente se informado “desc” (se não informado, assume valor “asc”);
- **direcao_da_chamada**: pode assumir o valor “inbound” para filtrar apenas gravações de chamadas entrantes, ou “outbound” para filtrar apenas

gravações de chamadas saintes. Pode ainda assumir o valor “all” para informar todas as gravações (se não informado, assume o valor “all”);

RESPONSE DATA em JSON:

```
{ “command”: "get_list",
"status": "ok" ,
"total": total_de_registros ,
"start": registro_inicial ,
"count": quantidade_de_registros ,
"records": [
                {     "entity_id": id_da_entidade,
                      "record_id": id_da_gravacao,
                      "service": nome_do_servico
                      "media_type": tipo_de_midia,
                      "call_begin": inicio_da_chamada,
                      "record_begin": inicio_da_gravacao,
                      "record_end": termino_da_gravacao,
                      "source": origem_da_chamada,
                      "destination": destino_da_chamada,
                },
                ...
                {     "entity_id": ...
                }
        ]
}
```

RESPONSE DATA em XML:

```
<command>get_list</command>
<status>ok</status>
<records>
        <member>
            <entity_id>id_da_entidade</entity_id>
            <record_id>id_da_gravacao</record_id>
            <service>nome_do_servico</service>
            <media_type>tipo_de_midia</media_type>
            <call_begin>inicio_da_chamada</call_begin>
            <record_begin>inicio_da_gravacao</record_begin>
            <record_end>termino_da_gravacao</record_end>
            <source>origem_da_chamada</source>
            <destination>destino_da_chamada</destination>
        </member>
        ...
        <member>
            <entity_id>....</entity_id>
            ...
        </member>
</records>
```

Onde:

- **id_da_entidade:** identificador da entidade gravada;
- **id_da_gravacao:** identificador da gravação;

- **inicio_da_chamada:** data e horário do início da chamada;
- **inicio_da_gravacao:** data e horário do início da gravação;
- **termino_da_gravacao:** data e horário do término da gravação;
- **origem_da_chamada:** originador da chamada.

CONSULTA DE DADOS DE UMA GRAVAÇÃO (COMANDO GET_RECORD)

O comando **get_record** permite consultar os dados de uma gravação específica, identificada pelo id da entidade e pelo id da gravação.

Método HTTP GET

interact_cti/v1.0/recorder/get_record?entity_id=id_da_entidade&record_id=id_da_gravacao&recover=[true|false]&operator=nome_do_operador&format=[xml | json]

NOTA

*Se o parâmetro **recover** for passado com valor **true**, o arquivo gravado será disponibilizado no serviço de transferência de arquivos. Na resposta deste comando **get_record** será retornado um atributo **session_id** que deverá ser usado no comando **get_record_content** para se obter o identificador e a URL para se baixar o arquivo.*

Onde:

- **id_da_entidade**: identificador da entidade gravada;
- **id_da_gravacao:** identificador da gravação;
- **nome_do_operador**: nome de um usuário cadastrado com permissão de operador do gravador;
- **format (opcional)**: formato desejado para a resposta. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

RESPONSE DATA em JSON:

```
{
     “command”: "get_record",
     “status”: "ok" ,
     "entity_id": id_da_entidade,
     "record_id": id_da_gravacao,
     "service": nome_do_servico,
     "media_type": tipo_de_midia,
     "call_begin": inicio_da_chamada,
     "record_begin": inicio_da_gravacao,
     "record_end": termino_da_gravacao,
     "source": origem_da_chamada,
     "destination": destino_da_chamada,
     "server": servidor,
     "file”: arquivo,
     "associated_data”: dados_associados,
     "session_id": identificador_da_sessao
```

```
}
```

RESPONSE DATA em XML:

```
<command>get_list</command>
<status>ok</status>
<entity_id>id_da_entidade</entity_id>
<record_id>id_da_gravacao</record_id>
<service>nome_do_servico</service>
<media_type>tipo_de_midia</media_type>
<call_begin>inicio_da_chamada</call_begin>
<record_begin>inicio_da_gravacao</record_begin>
<record_end>termino_da_gravacao</record_end>
<source>origem_da_chamada</source>
<destination>destino_da_chamada</destination>
<server>servidor</server>
<file>arquivo</file>
<associated_data>dados_associados</associated_data>
<session_id>identificador_da_sessao</session_id>
```

Onde:

- **id_da_entidade:** identificador da entidade gravada;
- **id_da_gravacao:** identificador da gravação;
- **inicio_da_chamada:** data e horário do início da chamada;
- **inicio_da_gravacao:** data e horário do início da gravação;
- **termino_da_gravacao:** data e horário do término da gravação;

- **origem_da_chamada:** originador da chamada;
- **destino_da_chamada:** destino da chamada;
- **tipo_de_midia:** nome da mídia para a qual se deseja filtrar a lista de gravações;
- **nome_do_servico:** nome do serviço para o qual se deseja filtrar a lista de gravações;
- **servidor:** endereço IP do servidor onde está a gravação;
- **arquivo:** caminho e nome do arquivo de gravação;
- **dados_associados:** dados associados à gravação;
- **identificador_da_sessao**: identificador da sessão de transferência de arquivos que deve ser usado no comando **get_record_content**.

RECUPERAÇÃO DE CONTEÚDO DE UMA GRAVAÇÃO (COMANDO GET_RECORD_CONTENT)

O comando **get_record_content** permite recuperar os dados (inclusive arquivos) de uma gravação. No caso da mídia Voz, é possível baixar o arquivo de áudio do serviço de transferência de arquivos. No caso da mídia *Chat*, é retornado um objeto com o histórico da conversa, com a troca de mensagens e arquivos (disponibilizados para *download* no serviço de transferência). No caso da mídia *E-mail*, é retornado um objeto com os dados da mensagem de *e-mail* interpretados e separados em cabeçalho, corpo e anexos (disponibilizados para *download* no serviço de transferência).

Método HTTP GET

interact_cti/v1.0/recorder/get_record_content?session_id=id_da_sessao&format=[xml | json]

Onde o **id_da_sessao** é o identificador da sessão de transferência de arquivos, obtido no comando **get_record**.

RESPONSE DATA em JSON para mídia Voz:

```
{
    "file_id": identificador_do_arquivo,
    "download": url_para_download
}
```

RESPONSE DATE em XML para mídia Voz:

```
<file_id> identificaor_do_arquivo </file_id>
<download> url_para_download </dowload>
```

Onde:

- **identificador_do_arquivo**: é o identificador para se acessar o arquivo no serviço de transferência de arquivos;
- **url_para_download**: é o caminho completo para se realizar o *download* do arquivo no serviço de transferência de arquivos.

RESPONSE DATA em JSON para mídia *Chat*:

```
{
```

```
  "form": dados_de_formulario,
  "history":
  [
    {
      "actor": nome_do_ator,
      "action": nome_da_acao,
      "message": texto_da_mensagem,
      "message_id": identificador_da_mensagem,
      "timestamp": timestamp_da_acao
    },
    {
      "actor": nome_do_ator,
      "action": nome_da_acao,
      "file_name": nome_do_arquivo,
      "file_index": indice_do_arquivo,
      "file_id": identificador_do_arquivo,
      "video_index": indice_do_arquivo_de_video,
      "motive": motivo_de_cancelamento_de_video,
      "message": mensagem_de_cancelamento_de_video,
      "download": url_para_download,
      "timestamp": timestamp_da_acao
    },
    ...
    {...}
  ]
}
```

RESPONSE DATE em XML para mídia *Chat*:

```
  <form>dados_de_formulario</form>
  <history>
    <member>
```

```
      <actor> nome_do_ator </actor>
      <action> nome_da_acao </action
>
      <message> texto_da_mensagem </message>
      <message_id> identificador_da_mensagem </message_id>
      <timestamp> timestamp_da_acao </timestamp>
    </member>
    <member>
      <actor> nome_do_ator </actor>
      <action> nome_da_acao </action
>
      <file_name> nome_do_arquivo </file_name>
      <file_index> indice_do_arquivo </file_index>
      <file_id> identificador_do_arquivo </file_id>
      <vídeo_index> indice_do_arquivo_de_video </vídeo_index>
      <motive> motivo_de_cancelamento_de_video </motive>
      <message> mensagem_de_cancelamento_de_video </message>
      <download> url_para_download </download>
      <timestamp> timestamp_da_acao </timestamp>
    </member>
    ...
    <member>
      ...
    </member>
  </history>
```

Onde:

- **dados_de_formulario**: array em que cada dado de formulário enviado na abertura do *Chat* é informado em uma estrutura com os campos **key** (chave) e **value** (valor);
- **nome_do_ator**: nome do responsável pela ação;
- **nome_da_acao**: nome da ação executada. Pode ser:
  - *message_sent*: ação de envio de mensagem de texto;
  - *file_requested*: ação de solicitação de transferência de arquivo;
  - *file_accepted*: ação de aceite de transferência de arquivo;
  - *file_rejected*: ação de rejeição de transferência de arquivo;
  - *file_available*: ação de notificação de disponibilidade de arquivo;
  - *file_accessed*: ação de notificação de acesso ao arquivo;
  - *file_canceled*: ação de cancelamento de transferência de arquivo;
  - *video_start*: ação de início de interação de vídeo;
  - *video_stop*: ação de término de interação de vídeo;
  - *video_origin_registry*: ação de registro da origem da interação de vídeo;
  - *video_destiny_registry*: ação de registro do destino de interação de vídeo;
  - *video_offer*: ação de oferta da interação de vídeo;
  - *video_accept*: ação de aceite de interação de vídeo;
  - *video_canceled*: ação de cancelamento de interação de vídeo.
- **texto_da_mensagem**: texto da mensagem enviada;
- **identificador_da_mensagem:** identificador numérico sequencial único de cada mensagem em uma determinada chamada de *Chat*;
- **timestamp_da_acao**: timestamp da ação;
- **nome_do_arquivo**: nome do arquivo transferido;

- **indice_do_arquivo**: índice do arquivo transferido (número crescente iniciando em 1);
- **indice_do_arquivo_de_video**: índice do arquivo de vídeo transferido. Atributo presente apenas em eventos de interação de vídeo;
- **motivo_de_cancelamento_de_video**: nome do motivo do cancelamento de interação de vídeo enviado pelo serviço de vídeo. Atributo presente apenas quando o atributo *nome_da_acao for video_canceled.*
- **mensagem_de_cancelamento_de_video**: mensagem descritiva do motivo do cancelamento de interação de vídeo enviado pelo serviço de vídeo. Atributo presente apenas quando o atributo *nome_da_acao* for *video_canceled.*
- **identificador_do_arquivo**: é o identificador para se acessar o arquivo no serviço de transferência de arquivos;
- **url_para_download**: é o caminho completo para se realizar o *download* do arquivo no serviço de transferência de arquivos.

RESPONSE DATA em JSON para mídia *E-mail*:

```
"email":
{
  "call_id": identificador_da_chamada,
  "direction": direcao_da_chamada
  "header":
 {
    "subject": assunto,
    "date": data,

    "from":
    {
      "name": nome_origem,
      "address": endereco_origem
```

```
  },
  "to":
  [
    {
      "name": nome_destino1,
      "address": endereco_destino1
    },
...,
    {...}
  ],
  "cc":
  [
    {
      "name": nome_copia1,
      "address": endereco_copia1
    },
    ...,
    {...}
  ],
  "cco":
  [
{
      "name": nome_copia_oculta1,
      "address": endereco_copia_oculta1
    },
    ...,
    {...}

  ],
  "reply_to":
  {
    "name": nome_responder_para,
    "address": endereco_responde_para
```

```
    }
  },
  "body":
 {
    "content_type": tipo_de_conteudo_corpo ,
    "charset": codificacao_corpo,
    "transfer_encoding": codificacao_de_transferencia_corpo,
    "data": dados
  },
  "attachments":
  [
    {
      "name": nome_anexo1,
      "size": tamanho_anexo1,
      "content_type": tipo_de_conteudo_anexo1,
      "charset": codificacao_anexo1,
      "transfer_encoding": codificacao_de_transferencia_anexo1,
   "disposition": disposicao_anexo1,
      "contend_id": identificador_de_conteudo1,
      "file_id": identificador_do_arquivo1,
      "download": url_anexo1
    },
    ...,
   {...}
  ]
}
```

RESPONSE DATE em XML para mídia *E-mail*:

```
<email>
  <call_id> identificador_da_chamada </call_id>
  <direction> direcao_da_chamada </direction>
  <header>
```

```
<subject> assunto </subject>
<date> data </date>
<from>
  <name> nome_origem </name>
  <address> endereco_origem </address>
</from>
<to>
  <member>
    <name> nome_destino1 </name>
    <address> endereco_destino1 </address>
  </member>
  ...
  <member>
   ...
  </member>
</to>
<cc>
  <member>
    <name> nome_copia1 </name>
    <address> endereco_copia1 </address>
  </member>
  ...
  <member>
    ...

  </member>
</cc>
<cco>
  <member>
    <name> nome_copia_oculta1 </name>
    <address> endereco_copia_oculta1 </address>
  </member>
  ...
```

```
      <member>
        ...
      </member>
    </cco>
    <reply_to>
      <name> nome_responder_para </name>
      <address> endereco_responder_para </address>
    </reply_to>
  </header>
  <body>
    <content_type> tipo_de_conteudo_corpo </content_type>
    <charset> codificacao_corpo </charset>
    <transfer_encoding> codificacao_de_transferencia_corpo
</transfer_encoding>
    <data> dados </data>
  </body>
  <attachments>
    <member>
      <name> nome_anexo1 </name>
      <size> tamanho_anexo1 </size>
      <content_type> tipo_de_conteudo_anexo1 </content_type>
      <charset> codificacao_anexo1 </charset>
      <transfer_encoding> codificacao_de_transferencia_anexo1
</transfer_encoding>

      <disposition> disposicao_anexo1 </disposition>
      <contend_id> identificador_de_conteudo1 </content_id>
      <file_id> identificador_do_arquivo1 </file_id>
      <download> url_anexo1 </download>
    </member>
    ...
    <member>
    </member>
```

```
  </attachments>
</email>
```

Onde:

- **identificador_da_chamada**: identificador único da chamada;
- **assunto**: assunto da mensagem de *e-mail;*
- **direcao_da_chamada**: poderá ser *in*, indicando chamada entrante ou *out* indicando chamada sainte;
- **data**: data em que a mensagem foi enviada pela origem;
- **nome_origem**: nome do originador da mensagem;
- **endereco_origem**: endereço de *e-mail* do originador da mensagem;
- **nome_destinoN**: nome do N-ésimo destinatário da mensagem;
- **endereco_destinoN**: endereço do N-ésimo destinatário da mensagem;
- **nome_copiaN**: nome do N-ésimo destino em cópia da mensagem;
- **endereco_copiaN**: endereço do N-ésimo destino em cópia da mensagem;
- **nome_copia_ocultaN**: nome do N-ésimo destino em cópia oculta da mensagem;
- **endereco_copia_ocultaN**: endereço do N-ésimo destino em cópia oculta da **mensagem;**
- **nome_responder_para**: nome do destino de resposta da mensagem;
- **endereco_responder_para**: endereço do destino de resposta da mensagem;
- **tipo_de_conteudo_corpo**: tipo de conteúdo (MIME type) do corpo da mensagem;
- **codificacao_corpo**: codificação do corpo da mensagem;

- **codificacao_de_transferencia_corpo**: codificação de transferência do corpo da mensagem;
- **dados**: dados da mensagem (em base64);
- **nome_anexoN**: nome do N-ésimo arquivo anexo da mensagem;
- **tamanho_anexoN**: tamanho em bytes do N-ésimo arquivo anexo da mensagem;
- **tipo_de_conteudo_anexoN**: tipo de conteúdo (MIME type) do N-ésimo arquivo anexo da mensagem;
- **codificacao_anexoN**: codificação do N-ésimo arquivo anexo da mensagem;
- **codificacao_de_transferencia_anexoN**: codificação de transferência do N-ésimo arquivo anexo da mensagem;
- **disposicao_anexoN**: disposição do N-ésimo arquivo anexo da mensagem. Pode ser attachment para arquivo anexo, ou *inline* para arquivo referenciado como parte do corpo da mensagem. Neste último caso, o atributo **content_id** possibilita identificar o local no corpo da mensagem em que o arquivo deve ser apresentado;
- **identificador_de_conteudoN**: identificador do conteúdo do anexo (utilizado apenas quando a disposição do mesmo for inline);
- **identificador_do_arquivoN**: identificador do arquivo no serviço de transferência de arquivos;
- **url_anexoN**: URL para se realizar o *download* do arquivo anexo.

NOTA

*Os atributos **cc**, **cco** e **attachments** podem não possuir conteúdo, caso em que seu valor será **null** em JSON ou será uma tag vazia em XML.*

ASSOCIAÇÃO DE DESCRIÇÃO A UMA GRAVAÇÃO (COMANDO ASSOCIATE_DATA)

O comando **associate_data** permite associar uma descrição (dados associados) a uma gravação específica, identificada pelo id da entidade e pelo id da gravação.

Método HTTP POST

interact_cti/v1.0/recorder/associate_data?entity_id=id_da_entidade&record_id=id_da_gravacao&operator=nome_do_operador&format=[xml | json]

POST DATA em JSON:

```
{
   "data": dados_associados
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
       <data>dados_associados</data>
</request>
```

Onde:

- **id_da_entidade**: identificador da entidade gravada;
- **id_da_gravacao**: identificador da gravação;

- **nome_do_operador**: nome de um usuário cadastrado com permissão de operador do gravador;
- **format (opcional):** formato desejado para a resposta. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição;
- **dados_associados:** são os dados a serem associados à gravação.

NOTA

O tamanho máximo do texto de dados associados para uma gravação é de 254 bytes.

CONTROLE DE REPRODUÇÃO DE GRAVAÇÃO DE VOZ EM UM RAMAL (COMANDO SET_PLAYER)

O comando **set_player** é específico para gravações na mídia Voz e permite controlar a reprodução de uma gravação em um ramal determinado.

Método HTTP POST

interact_cti/v1.0/recorder/set_player?entity_id=id_da_entidade&record_id=id_da_gravacao&operator=nome_do_operador&format=[xml | json]

POST DATA em JSON:

```
{
   "operation": tipo_de_operacao,
```

```
        "gain": ganho,
        "branch": ramal
    }
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
        <operation>tipo_da_operacao</operation>
        <gain>ganho</gain>
        <branch>ramal</branch>
</request>
```

Onde:

- **id_da_entidade**: identificador da entidade gravada;
- **id_da_gravacao:** identificador da gravação;
- **nome_do_operador:** nome de um usuário cadastrado com permissão de operador do gravador;
- **format (opcional):** formato desejado para a resposta. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição;
- **tipo_da_operacao**: define a operação a ser realizada sobre a reprodução: start, stop, pause, resume, forward, rewind ou set_volume;
- **ganho:** ganho de volume em decibéis (apenas para operação do tipo set_volume);
- **ramal:** ramal onde é executada a reprodução da gravação.

NOTA

*A primeira operação que deve ser enviada nesse comando é **start**. A reprodução se iniciará no ramal indicado. Poderá ser comandada a pausa (operação **pause**) e a retirada de pausa (**resume**). Também poderá ser comandado o avanço de 10s na gravação (operação **forward**) ou o retrocesso de 10s na gravação (operação **rewind**). Para encerrar a reprodução deve ser enviada a operação **stop**.*

10

DISCADOR

O recurso **Dialer** disponibiliza algumas funcionalidades dos serviços do tipo **ativo** do **Interact** para integração.

Os recursos disponibilizados pelo recurso **Dialer** do **Interact CTI** são:

- Iniciar serviço - Este comando inicia um serviço de discagem.
- Pausar serviço - Este comando pausa um serviço em andamento.
- Retirar serviço de pausa - Este comando reinicia um serviço pausado.
- Importar lista de contatos - Este comando possibilita a importação de uma lista de contatos para um determinado serviço.
- Incluir contato - Este comando possibilita a inclusão de um contato em uma lista de um serviço.
- Atualizar contato - Este comando possibilita a atualização de um contato em uma lista de um serviço.
- Solicitar contato(s) - Este comando possibilita a recuperação de um ou mais contatos de uma lista de um serviço.
- Excluir contato - Este comando possibilita a exclusão de um contato de uma lista.
- Monitoração de eventos de alteração de estado da campanha.

- Monitoração de eventos de *status* de discagem dos contatos.

Os serviços disponibilizados pelo **Interact CTI** para discador por meio de REST têm o seguinte formato geral:

interact_cti/v1.0/dialer[monitor/]metodo?parametros[&format= (json|xml)]

Onde:

- **metodo:** nome da requisição (comando).
- **parametros:** conjunto de pares nome-valor na forma prm=valor, separados por & (Ecomercial-ampersand).
- **format:** define o formato desejado para a resposta, a qual pode ser no formato JSON (.json) ou no formato (.xml). Na ausência desse parâmetro será utilizado o mesmo formato informado no header Content-type.

Observações:

- Na ausência do header **Content-type** será utilizado **XML** por *default*.
- Na ausência de parâmetros e de formato, o sinal de interrogação após método é desnecessário.
- As requisições aos serviços serão sempre por meio de métodos HTTP GET ou POST.
- O método GET é utilizado em todas as requisições que não enviam apenas parâmetros.
- O método POST é utilizado em todas as requisições que enviam pacotes de dados (além dos parâmetros).
- Os parâmetros enviados na URL deverão ser codificados no padrão URL encode.
- Os dados nos pacotes POST deverão ser codificados em UTF-8.

- O acesso se dará na porta 8582 (valor *default* que pode ser alterado por configuração), conforme exemplo abaixo:
- *http://XXX.XXX.XXX.XXX:8582/interact_cti/v1.0/dialer/subrecurso/metodo*

Onde:

- **XXX.XXX.XXX.XXX é o** endereço IP do servidor **Interact CTI**

MONITORAÇÃO DE SERVIÇOS DO TIPO ATIVO

A monitoração de *status* de discagem dos contatos e alterações no estado dos serviços tipo ativo estão disponíveis no recurso **Dialer** do **Interact CTI.**

A utilização da monitoração de *status* de discagem e alterações de estado dos serviços ativos dependem de licença específica.

Os serviços disponibilizados pelo **Interact CTI** para monitoração de estatísticas e alarmes por meio de REST têm o seguinte formato geral:

interact_cti/v1.0/dialer/monitor/metodo?parametros[&format= (json|xml)]

Onde:

- **metodo**: nome da requisição (comando);
- **parametros** : conjunto de pares nome - valor na forma prm=valor, separados por & (E comercial - ampersand);
- **format**: define o formato desejado para a resposta, que pode ser no formato JSON (.json) ou no formato (.xml). Na ausência desse parâmetro, será utilizado o mesmo formato informado no header Content - type. Na ausência do header Content - type será utilizado XML por *default*.

ASSINATURA DE EVENTOS PARA MONITORAÇÃO (COMANDO SUBSCRIBE)

O comando subscribe possibilita que o *status* de discagem dos contatos e alterações nos estados dos serviços sejam monitorados.

Método HTTP POST

interact_cti/v1.0/dialer/monitor/subscribe?format=[xml | json]

O parâmetro **format** é opcional. Caso não seja enviado, a resposta será no mesmo formato do pacote de dados enviado na requisição.

POST DATA em JSON

```
{
 "subscription": {
        "id":  identificador_da_assinatura,
        "agents": [ lista_de_agentes ],
        "services": [ lista_de_servicos ],
        "events": [ lista_de_eventos  ],
        "channel_id": identificador_do_canal,
        "webhook": url_do_cliente,
        "expires": tempo_de_expiracao
    }
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8">
<request>
   <subscription>
   <id> identificador_da_assinatura </id>
   <agents> lista_de_agentes </agents>
   <services> lista_de_servicos </services>
   <events> lista_de_eventos </events>
   <channel_id> identificador_do_canal </channel_id>
   <webhook> url_do_cliente </webhook>
   <expires> tempo_de_expiracao </expires>
  </subscription>
</request>
```

Onde:

- **identificador_da_assinatura (opcional)** - É o identificador da assinatura. Deverá ser enviado apenas caso se deseje modificar uma assinatura já existente.
- **lista_de_agentes (opcional)** - É uma lista com os nomes dos agentes para monitoração (a ausência deste parâmetro ou o uso do valor "all" (ou *) indica monitoração de todos os agentes - dentro do limite da licença CTI).
- **lista_de_eventos (opcional)** - É uma lista com os nomes dos eventos de operação para monitoração (a ausência deste parâmetro ou o uso do valor "all" (ou *) indica monitoração de todos os eventos). Apenas os eventos de operação (aqueles relacionados a agentes e a chamadas) é que podem ser assinados. Eventos de controle são sempre enviados. Ver observação abaixo).
- **lista_de_servicos (opcional) -** É uma lista com o nome dos serviços do tipo ativo para monitoração (a ausência deste parâmetro ou o uso do valor "all" (ou *) indica monitoração de todos os serviços)

- **tempo_de_expiracao -** É o tempo em segundos durante o qual a assinatura permanecerá ativa. O valor máximo é 86400, o que corresponde a um dia completo. Este parâmetro é obrigatório. Valores aceitos entre 1 e 86400.
- **Identificador_do_canal -** É o identificador do canal de eventos pelo qual os eventos desta assinatura deverão ser enviados. Os canais podem ter identificadores dentro da faixa numérica deste 1 até 9999.
- **url_do_cliente:** endereço do webservice do cliente que espera receber eventos por HTTP POST.

Os eventos de controle do canal (**on_open_channel** e **on_close_channel**), de disponibilidade do servidor (**on_server_state_change**) e de expiração de assinatura (**on_subscription_expire**) não são passíveis de serem assinados e sempre serão enviados aos clientes.

NOTA

Uma assinatura pode ser alterada com o envio de novo comando de assinatura passando o código identificador da assinatura (id). A assinatura anterior será cancelada e a nova ficará ativa em seu lugar. Para renovação da assinatura basta que sejam enviados os mesmos parâmetros da assinatura original adicionado do código identificador da assinatura (id). O tempo de expiração será reiniciado para o valor que for informado no campo expires. Todos os parâmetros da assinatura podem ser alterados, inclusive o número do canal para envio de eventos.

Resposta Ao Comando De Assinatura

A resposta para a assinatura de eventos possui o seguinte formato:

RESPONSE DATA em JSON:

```
{
    "response": {
        "command": "subscribe",
        "id": identificador_da_assinatura,
        "status": status,
        "error_type": tipo_de_erro,
        "error_items": listas_de_items_com_erro
    }
}
```

RESPONSE DATA em XML:

```
<?xml version="1.0"encoding="UTF-8"?>
<response>
  <command> subscribe </command>
  <id> identificador_da_assinatura </id>
  <status> status </status>
  <error_type> tipo_de_erro </error_type>
  <error_items> listas_de_items_com_erro </error_items>
</response>
```

Onde:

- **identificador_da_assinatura**: identificador numérico único para a assinatura;
- **status**: estado da execução do comando (pode ser **ok** em caso de sucesso, ou **error** em caso de erro);
- **tipo_de_erro**: enviado apenas quando *status* for error, e pode ser:
  - **empty_data** : enviado quando o pacote de dados estiver vazio;

- **invalid_data** : enviado quando o pacote de dados estiver em formato inválido;
- **not_found:** enviado em resposta a um comando de atualização com campo de id inexistente no servidor. Pode ocorrer no caso da tentativa de atualização de uma assinatura que já expirou;
- **ivalid_item:** enviado quando o pacote de dados contiver algum item inválido (como o nome de um agente, o nome de um serviço, o nome de um evento ou o nome de um tipo de mídia);
- **no license:** enviado quando não há licenças disponíveis em número suficiente para atender à assinatura.

▪ **lista_de_itens_com_erro:** enviado apenas quando o status for **error** e o **tipo_de_erro** for **invalid_item**. O conteúdo de **error_items** será uma lista de todos os itens que apresentaram erro na assinatura, separados nas tags **agentes, services, events, media_types, agente_stats, servise_stats** e **alarms.**

ALTERAÇÃO E RENOVAÇÃO DE ASSINATURA (COMANDO SUBSCRIBE)

Uma assinatura pode ser alterada com o envio de novo comando de assinatura passando o código identificador da assinatura **(id)**. A assinatura anterior será cancelada e a nova ficará ativa em seu lugar. Para renovação da assinatura basta que sejam enviados os mesmos parâmetros da assinatura original adicionados do código identificador da assinatura (id). O tempo de expiração será reini ciado para o valor que for informado no campo expires. Todos os parâmetros da assinatura podem ser alterados, inclusive o número do canal para envio de eventos.

CANCELAMENTO DE ASSINATURA (COMANDO UNSUBSCRIBE)

Uma assinatura poderá ser cancelada ao se enviar um comando de unsubscribe informando - se o código identificador da assinatura (id). Havendo assinatura com o id informado, ela será cancelada. Caso o id não exista será retornado o erro **not found**.

Método HTTP POST

interact_cti/v1.0/dialer/monitor/unsubscribe?format=[xml | json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON

```
{
“id”: identificador_da_assinatura
}
```

POST DATA em XML:

```
<request>
<id>identificador_da_assinatura</id>
</request>
```

Onde:

- **identificador_da_assinatura**: número idenficador da assinatura recebido na resposta de um comando de assinatura (subscribe) bem sucedido.

ABERTURA DE CANAL PARA EVENTOS (COMANDO OPEN_CHANEL)

A recepção de eventos será sempre realizada por meio de canais. Um canal é uma conexão HTTP persistente, que deverá ser aberta por meio do comando **open_channel.**

Na abertura do canal deverá ser informado um identificadornumérico (**channel_id**) e um formato para a recepção dos dados (XML ou JSON).

Cada canal possui um identificador único (**channel_id**). Este identificador deve ser informado como um parâmetro passado no comando **open_channel** ou, se não informado, será atribuído automaticamente pelo **Interact CTI**. Em qualquer caso, o **channel_id** é indicado no evento **on_open_channel**.

Na abertura do canal deverá ser informado um formato para a recepção dos dados (XML ou JSON).

O identificador do canal deverá ter um valor na faixa entre 1 e 9999 e poderá ser associado à várias assinaturas.

Um canal é fechado automaticamente depois de 1 minuto sem alguma assinatura associada. Isso vale, também, para o caso de se abrir um canal sem nenhuma assinatura associada. Ele permanecerá aberto por 1 minuto e, caso nenhuma assinatura seja feita nesse intervalo, ele será automaticamente fechado.

Método: HTTP GET.

interact_cti/v1.0/dialer/monitor/open_channel?channel_id=nnnn&format=[xml | json]

O campo nnnn acima deve ser substituído pelo código identificador do canal(channel_id) que se deseja abrir. O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON). Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
“channel_id”: identificador_do_canal
}
```

POST DATA em XML:

```
<request>
<channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

identificador_do_canal: número idenficador do canal que se deseja abrir.

Observação:

Caso deseje abrir um canal com um identificador (channel_id) que já existe ou fora da faixa válida (entre 1 e 9999), será retornado o erro HTTP 403 Forbidden.

FECHAMENTO DE CANAIS DE RECEPÇÃO DE EVENTOS (COMANDO CLOSE_CHANEL)

Um canal pode ser fechado por meio do envio do comando close_channel , informando-se o seu identificador numérico (channel_id) e um formato para a recepção dos dados (XML ou JSON). Ressalta-se que um canal será automaticamente fechado depois de 1 minuto sem alguma assinatura associada, independente do envio do comando close_channel.

Método HTTP POST

interact_cti/v1.0/dialer/monitor/close_channelformat=[xml|json]

O parâmetro **format** é opcional e permite definir o formato dos dados da resposta (XML ou JSON).

Caso esse parâmetro não seja enviado, a resposta será no formato XML.

POST DATA em JSON:

```
{
“channel_id”: identificador_do_canal
}
```

POST DATA em XML:

```
<request>
<channel_id>identificador_do_canal</channel_id>
</request>
```

Onde:

- **identificador_do_canal**: número identificador do canal que se deseja fechar.

MANUTENÇÃO DO CANAL ABERTO

Para que uma conexão HTTP se mantenha de forma persistente é necessário que haja tráfego de pacotes desde o servidor até o cliente (autor da requisição) com uma certa frequência, caso contrário a conexão acabará sendo derrubada por time-out (que pode ocorrer tanto em um browser como em um servidor intermediário, entre o servidor do **Interact CTI** e a aplicação cliente). O host RestServer do **Interact CTI** pode ser configurado (em arquivo XML) para enviar um pacote de "heart beat" (batida de coração) sempre que um canal ficar mais do que um determinado tempo sem comunicação. o **Interact CTI** enviará um pacote de dados vazio apenas para sinalizar a atividade do canal e manter a conexão estabelecida.

O pacote de "heart beat" possui o seguinte formato: **`data: {}`**

RECEPÇÃO DOS EVENTOS PELO CANAL

Os eventos serão enviados dentro do padrão de Server Send Events da W3C (http://www.w3.org/TR/eventsource).

No **Interact CTI**, cada evento será enviado em um pacote de dados no seguinte formato:

event: nome_do_evento

data: dados_do_evento

Onde:

event: nome do evento.

- Para controle do estado do canal:
  - *on_open_channel*
  - *on_close_channel*
- Para controle da assinatura:
  - *on_subscription_expire*
  - *on_subscription_expire_warning*
- Para controle do estado do servidor **Interact**:
  - *on_server_state_change*
- Para o recurso dialer será um dos seguintes eventos (eventos que podem ser assinados):
  - *on_dialer_state_change.*
  - *on_dialer_contact_status.*

EVENTOS DE MONITORAÇÃO DO DISCADOR

Os eventos gerados pelo recurso dialer estão relacionados às campanhas (serviços do tipo ativo) e chamadas do **Interact**. Este tipo de evento pode ser assinado, ou seja, só será recebido se o integrador assim o desejar). Segue a relação dos eventos:

Evento	Descrição
on_dialer_state_change	Ocorre na alteração de estado de uma campanha.
on_dialer_contact_status	Ocorre na discagem de um contato da campanha.

On_dialer_state_change

O evento on_dialer_state_change contém os seguintes atributos:
"service":nome_servico,
"type":tipo_servico,
"state": estado_do_servico,
"cause": motivo_de_parada,
"paused": pausado,
"succeed": qtd_sucesso,
"failed": qtd_falha

Onde:

- **nome_servico:** Nome do serviço.
- **tipo_servico**: Algoritimo de discagem da campanha, os algoritomo são:
  - ready
  - preview
  - predictive
  - power
  - validation
- **estado_do_servico**: Estado em que o serviço se encontra, os estados possíveis são:
  - running
  - stopped
- **motivo_de_parada**: Motivo de parada do serviço. Somente é enviado caso o estado do serviço seja stopped. Os motivos são:

- user_pause: Pausado pelo usuário.
- out_of_service: Fora de expediente.
- no_credit: Sem crédito para gerar chamada.
- no_service_license: Sem licença de campanha.
- no_handle_license: Sem licença de atendimento.
- no_contacts: Sem contatos no mailing.
- invalid_config: Alguma configuração inválida.
- degraded: A campanha esta degradada gerando menos chamada.
- pabx_failure: Sem conexão com o PABX.
- hardware_failure: Número excessivo de falhas de hardware, pausa a campanha.
- software_failure: Número excessivo de falhas nos números do lote, pausa a campanha.
- scheduled: Antes do período de agendamento.
- scheduling_completed: Depois do período de agendamento
- disabled_media: Mídia Voz foi desabilitada
- unknown: desconhecido.

- **pausado**: Campanha está pausada, os valores são:
  - true
  - false
- **qtd_sucesso**: Quantidade de contatos discados com sucesso.
- **qtd_falha**: Quantidade de contatos em que houve falha na discagem.

On_dialer_contact_status

O evento on_dialer_state_change contém os seguintes atributos:

```
"service": nome_servico,
"type": tipo_servico,
"contact": {
    "call_id": identificador_chamada,
    "name": nome,
    "complement": complemento,
    "contact_id": identificador,
    "number": número,
    "attempts": tentativas,
    "status": status_de_geração
}
```

Onde:

- **nome_servico**: Nome do serviço.
- **tipo_servico**: Algoritimo de discagem da campanha, os algoritmos são:
  - ready
  - preview
  - predictive
  - power
  - validation
- **identificador_chamada**: Identificador da chamada gerada.
- **nome**: Nome do contato discado.
- **complemento**: Complemento do contato discado.
- **identificador:** Identificador do contato discado.
- **número**: Número discado do contato.

- **tentativas**: Número de tentativas de discagem para esse número.
- **status_de_geração**: *Status* de geração do contato, os estados possíveis são:
  - success
  - busy
  - invalid_number
  - refused_by_agent
  - refused
  - overflowed
  - no_available_agent
  - generation_error

ALTERAÇÃO DO ESTADO DE UMA CAMPANHA (COMANDO_SET_STATE)

Este método permite alterar o estado de uma campanha (serviço ativo) do **Interact**.

Método: HTTP POST

interact_cti/v1.0/dialer/set_state?format=[xml|json]

POST data em XML:

```
<request>
     <service>nome_do_servico</service>
     <state>estado</state>
```

```
</request>
```

POST data em JSON:

```
{
    "service": nome_do_servico,
    "state": estado
}
```

Onde:

- **nome_do_servico**: nome do serviço.
- **estado**: estado em que a campanha será colocada. sendo possível enviar os seguintes estados:
  - start: Inicia uma campanha.
  - pause: Pausa uma campanha.
  - resume: Sair de pausa.

NOTAS

1) *Só é possivel alterar o estado de uma campanha para um outro estado.*
2) *O estado para iniciar (**start**) a campanha só é aceito caso o estado da campanha seja **stopped** e a campanha e esteja em horário de funcionamento.*
3) *Não é possível parar uma campanha, a campanha entra e sai de funcionamento a partir dos horários de expediente definidos.*

Resposta para o comando:

As respostas para o comando seguem o padrão de respostas.

CONSULTA AO ESTADO DE UMA CAMPANHA (COMANDO GET_STATE)

Este método permite saber em qual estado se encontra uma campanha (serviço ativo) do **Interact**.

Método: HTTP GET

interact_cti/v1.0/dialer/get_state?service=nome_do_servico&format=[xml|json]

Onde:

- **nome_do_servico:** nome do serviço que se deseja obter os dados de estado.

Resposta

XML

```
<response>
 <command>get_state</command>
 <status>ok</status>
 <campaign_state>estado_da_campanha</campaign_state>
 <paused>estado_pausa</paused>
 <cause>motivo_falha</cause>
```

```
</response>
```

JSON

```
{
   "response":
   {
       "command": "get_state",
       "status": "ok",
       "campaign_state": estado_da_campanha,
       "paused": estado_pausa,
       "cause": motivo_falha
   }
}
```

Onde:

- **estado_da_campanha**: Os possíveis estados em que a campanha pode se encontrar são:
  - running: A campanha está em execução
  - stopped: A campanha está parada por alguma falha.
- **estado_pausa**: Retorna ***true*** caso a campanha esteja pausada e **false** caso não esteja.
- **motivo_falha**: Informa a falha que parou a campanha(verificar tabela de falhas da campanha).. Esse atributo só será exibido quando o estado da campanha for **stopped**.

FALHAS GERADAS PELA CAMPANHA

Falha	Descrição
user_pause	Pausado pelo usuário
out_of_service	Fora de expediente
no_credit	Sem crédito para gerar chamada
no_service_license	Sem licença de campanha
no_handle_license	Sem licença de atendimento
no_contacts	Sem contatos na lista de contatos do serviço
invalid_config	Alguma configuração inválida
degraded	A campanha está degradada gerando menos chamada.
pabx_failure	Sem conexão do PABX
hardware_pabx	Número excessivo de falhas de hardware, pausa a campanha
Software_failure	Número excessivo de falhas nos números do lote, pausa a campanha
scheduled	Antes do período de agendamento
scheduling_completed	Depois do período de agendamento
disabled_media	Mídia Voz foi desabilitada
unknown	Desconhecido

INCLUSÃO DE CONTATO (COMANDO CREATE_CONTACT)

Este método permite inserir um contato no mailing de uma campanha (serviço ativo) do **Interact**.

Método: HTTP POST

interact_cti/v1.0/dialer /create_contact?format=[xml|json]

POST Data em XML:

```
<?xml version=”1.0” enconding=”UTF-8”?>
<request>
<service>nome_do_servico</service>
<duplicate>duplicar_numeros</duplicate>
<contact>nome_contato</contact>
<complement>complemento</complement>
<phone_list>lista_de_numeros</phone_list>
<schedule>
   <day>dia_agendamento</day>
   <month>mes_agendamento</month>
   <year>ano_agendamento</year>
   <start>
       <hour>hora_inicial_agendamento</hour>
       <min>minuto_inicial_agendamento</hour>
```

```
    </start>
    <end>
        <hour>hora_final_agendamento</hour>
        <min>minuto_final_agendamento</hour>
    </end>
</schedule>
</request>
```

POST Data em JSON:

```
{
   "service": nome_do_servico,
   "duplicate": duplicar_numeros,
   "contact": nome_contato,
   "complement": complemento,
   "phone_list": lista_de_numeros
   "schedule":
   { "day": dia_agendamento,
     "month": mes_agendamento,
     "year": ano_agendamento,
     "start":
     { "hour": hora_inicial_agendamento,
       "min": minuto_inicial_agendamento
     }
     "end":
     { "hour": hora_final_agendamento,
```

```
        "min": minuto_final_agendamento
      }
    }
}
```

Onde:

- **nome_do_servico**: Nome do serviço que o contato será incluído.
- **duplicar_números**: Opção para validar números. Caso o valor seja ***true*** será permitida a inclusão de números duplicados. Caso o valor seja ***false*** será impedida a inclusão do contato de algum dos números informados na **lista_de_numeros** já existir no serviço em contatos não discados, contatos reagendados ou na própria **lista_de_numeros**, retornando, neste caso, o tipo de erro **duplicated_number.**
- **nome_contato**: Nome do contato. <OPCIONAL>
- **complemento**: Informação complementar do contato. <OPCIONAL>
- **lista_de_números**: Lista de números do contato, sendo que dez é o limite de números por contato.

O agendamento do contato é opcional. Para efetuá-lo, é necessário informar no objeto **schedule** o dia, o mês e o horário inicial da discagem. O horário final é opcional. Os parâmetros para agendamento são:

- **dia_agendamento**: dia do agendamento (1-31).
- **mes_agendamento**: mês do agendamento (1-12).
- **ano_agendamento:** ano do agendamento.
- **hora_inicial_agendamento**: hora de início da faixa de agendamento (0-23).
- **minuto_inicial_agendamento**: minuto de início da faixa de agendamento (0-59).

- **hora_final_agendamento**: hora de término da faixa de agendamento (0-23).
- **minuto_final_agendamento**: minuto de término da faixa de agendamento (0-59).

Resposta:

Em XML:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<response>
<resource>dialer</resource>
<command>create_contact</command>
<id>id_contato</id>
<status>ok</status>
</response>
```

Em JSON

```
{
   "resource": "dialer",
   "command": "create_contact",
   "id": id_contato,
   "status": "ok"
}
```

Onde:

- id_contato: Identificador do contato.

Em caso de erro, a resposta segue o padrão de respostas.

ATUALIZAÇÃO DE CONTATO (COMANDO UPDATE_CONTACT)

Este método permite alterar um contato no mailing de uma campanha (serviço ativo) do **Interact**.

Método: HTTP POST

interact_cti/v1.0/dialer/update_contact?format=[xml|json]

POST data em XML:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<request>
 <service>nome_servico</service>
 <id>id_contato</id>
 <contact>nome_contato</contact>
 <complement>complemento</complement>
 <phone_list>lista_de_numeros</phone_list>
 <schedule>
       <day>dia_agendamento</day>
       <month>mes_agendamento</month>
       <year>ano_agendamento</year>
       <start>
```

```
                <hour>hora_inicial_agendamento</hour>
                <min>minuto_inicial_agendamento</hour>
        </start>
        <end>
                <hour>hora_final_agendamento</hour>
                <min>minuto_final_agendamento</hour>
        </end>
  </schedule>

</request>
```

POST Data JSON

```
{
   "service": nome_servico
   "id": id_contato
   "contact": nome_contato,
   "complement": complemento,
   "phone_list": lista_de_numeros
   "schedule":
   {      "day": dia_agendamento,
          "month": mes_agendamento,
          "year": ano_agendamento,
          "start":
          {       "hour": hora_inicial_agendamento,
                  "min": minuto_inicial_agendamento
```

```
            }
            "end":
            {       "hour": hora_final_agendamento,
                    "min": minuto_final_agendamento
            }
      }

}
```

Onde:

- **nome_servico**: Nome do serviço.
- **id:** id_contato.
- **nome_contato**: Nome do contato.
- **complemento**: Complemento.
- **lista_de_numeros**: Lista de números do contato, sendo que dez é o limite de números por contato.

O agendamento do contato é opcional. Para efetuá-lo, é necessário informar no objeto ***schedule*** o dia, o mês e o horário inicial da discagem. O horário final é opcional. Os parâmetros para agendamentos são:

- **dia_agendamento:** dia do agendamento (1-31).
- **mes_agendamento:** mês do agendamento (1-12).
- **ano_agendamento:** ano do agendamento.
- **hora_inicial_agendamento:** hora de início da faixa de agendamento (0-23).
- **minuto_inicial_agendamento:** minuto de início da faixa de agendamento (0-59).

- **hora_final_agendamento:** hora de término da faixa de agendamento (0-23).
- **minuto_final_agendamento:** minuto de término da faixa de agendamento (0-59).

NOTA

Só é possível alterar o estado de contatos que não foram discados.

Resposta:

A resposta do comando segue o padrão de resposta.

EXCLUSÃO DE CONTATO (COMANDO DELETE_CONTACT)

Este método permite deletar um contato no mailing de uma campanha(serviço ativo) do **Interact**.

Método: HTTP POST

interact_cti/v1.0/dialer/delete_contact?format=[xml|json]

POST data em XML:

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

```
<request>
 <service>nome_servico</service>
 <id>id_contato</id>
</request>
```

POST Data JSON

```
 {
   "service": nome_servico
   "id": id_contato
 }
```

Onde:

- **nome_servico**: Nome do serviço.
- **id_contato**: Identificador do contato.

Resposta:

A resposta do comando segue o padrão de respostas.

RECUPERAÇÃO DE CONTATOS (COMANDO GET_CONTACT)

Este método possibilita obter os dados de um contato ou de um conjunto paginado de contatos.

Método: HTTP GET

a) Solicitação de um contato específico:
interact_cti/v1.0/dialer/get_contact?id=id_do_contato&format=[xml|json]

Resposta em caso de sucesso:

```
{
    "resource": "dialer",
    "command": "get_contact",
    "status": "ok",
    {
        "id": id_do_contato,
        "contact": nome_contato,
        "complement": complemento,
        "state": estado_contato,
        "phone1": numero_de_telefone,
        "phone2": numero_de_telefone,
        "phone3": numero_de_telefone,
        "phone4": numero_de_telefone,
        "phone5": numero_de_telefone,
        "phone6": numero_de_telefone,
        "phone7": numero_de_telefone,
        "phone8": numero_de_telefone,
        "phone9": numero_de_telefone,
        "phone10": numero_de_telefone,
```

```
        "update_date": data_atualização,
        "creation_date": data_inserção,
        "schedule":
        {       "day": dia_agendamento,
                "month": mes_agendamento,
                "year": ano_agendamento,
                "start":
                {       "hour": hora_inicial_agendamento,
                        "min": minuto_inicial_agendamento
                }
                "end":
                {       "hour": hora_final_agendamento,
                        "min": minuto_final_agendamento
                }
        }
}
```

Onde:

- **id_do_contato**: Identificador do contato.
- **nome_contato**: Nome do contato.
- **complemento**: Complemento.
- **estado_contato**: Estado do contato, verificar estado do contato.
- **numero_de_telefone**: Telefones do contato.
- **data_atualização:** Data da última alteração do contato.
- **data_inserção:** Data em que o contato foi inserido.

O agendamento do contato é opcional. Se o contato estiver agendado, será informado no objeto schedule o dia, o mês e o horário inicial do agendamento da discagem. O horário final é opcional. Os parâmetros para agendamento são:

- **dia_agendamento:** dia do agendamento (1-31).
- **mes_agendamento:** mês do agendamento (1-12).
- **ano_agendamento:** ano do agendamento.
- **hora_inicial_agendamento:** hora de início da faixa de agendamento (0-23).
- **minuto_inicial_agendamento:** minuto de início da faixa de agendamento (0-59).
- **hora_final_agendamento:** hora de término da faixa de agendamento (0-23).
- **minuto_final_agendamento:** minuto de término da faixa de agendamento (0-59).

b) Solicitação de conjunto paginado de contatos:

interact_cti/v1.0/dialer/get_contact?service=nome_do_servico[filter=texto_de_filtro] [&order_by=nome_da_variavel] [&direction=sentido_da_ordenacao&start=registro_inicial&count=numero_de_registros]&format=[xml|json]

Onde:

- **nome_do_servico**: nome do serviço;
- **texto_de_filtro**: texto para filtrar os contatos (aplicado sobre os nomes dos contatos).
- **nome_da_variavel:** id (default) | contact | complement | state | default_agent| creation_date;

- **sentido_da_ordenacao**: [asc | desc]. Valor *default* asc;
- **registro_inicial:** número inicial do registro desejado (a partir de uma determinada ordenação, a partir de 1). Valor *default* 1;
- **numero_de_registros**: número de registros desejados para que a consulta retorne. Valor *default* 10;

Resposta em caso de sucesso:

JSON:

```
{ "response":
        { "resource": "dialer",
          "command": "get_contact",
          "status": "ok",
          "total" : total_de_registros,
          "start" : registro_inicial,
          "count" : numero_de_registros,
          "contacts" : [
                        {
                        "id": id_do_contato1,
                        "contact": nome_contato1,
                        "complement": complemento1,
                        "state": estado_contato1,
                        "phone1": numero_de_telefone,
                        "phone2": numero_de_telefone,
                        "phone3": numero_de_telefone,
                        "phone4": numero_de_telefone,
```

```
                "phone5": numero_de_telefone,
                "update_date": data_atualização
                "creation_date": data_inserção,
              },
              ...
              {
              "id": id_do_contato1,
              "contact": nome_contato1,
              "complement": complemento1,
              "state": estado_contato1,
              "phone1": numero_de_telefone,
              "phone2": numero_de_telefone,
              "phone3": numero_de_telefone,
              "phone4": numero_de_telefone,
              "phone5": numero_de_telefone,
              "update_date": data_atualização
               "creation_date": data_inserção,
            }
          ]
        }
    }
```

Onde:

- **total_de_registros**: Total de contatos no mailing.
- **registro_inicial**: Número do registro inicial, útil para realizar paginação em conjunto com o parâmetro **count**.

- **numero_de_registros**: Quantidade de registros que a consulta retornou.
- **id_do_contato**: Identificador do contato.
- **nome_contato**: Nome do contato.
- **complemento**: Complemento.
- **estado_contato**: Estado do contato, verificar estado do contato.
- **numero_de_telefone**: Telefones do contato.

IMPORTAÇÃO DE CONTATOS (COMANDO IMPORT_CONTACTS)

Este método possibilita importar um conjunto de contatos para uma lista específica.

Método: HTTP POST

interact_cti/v1.0/dialer/import_contacts?service=nome_servico[&mode=modo][&duplicate=duplicar_contatos]&format=[xml|json]

Onde:

- **nome_servico**: Nome do serviço.
- **modo**: Existem dois modos para importação de contatos:
  - **append:** Os registros são adicionados ao mailing (modo *default*);
  - **overwrite:** O mailing é previamente esvaziado para receber os registros importados.
- **duplicar_contatos**: Valida os contatos enviados verificando se existem números de telefone duplicados.

- **true** - não realiza a verificação(opção *default*);
- **false** - realiza a verificação de contatos duplicados.

NOTA

1) O número de contatos enviados para importação está limitado a 100 por comando.

2) Cada contato pode ter até cinco números de telefones.

POST data em XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request>
  <contacts>
   <member>
    <item>id_item1</item>
    <contact>contato_1</contact>
    <complement>complemento1</complement>
    <phone_list>
    <member>numero_telefone1_1</member>
    <member>numero_telefone1_2</member>
    <member>numero_telefone1_3</member>
    <member>numero_telefone1_4</member>
    <member>numero_telefone1_5</member>
   </phone_list>
  </member>
  ...
  <member>
```

```
    <item>id_itemN</item>
    <contact>contatoN</contact>
    <complement>complementoN</complement>
    <phone_list>
    <member>numero_telefoneN_1</member>
    ...
    <member>numero_telefoneN_5</member>
   </phone_list>
   <schedule>
          <day>dia_agendamento</day>
          <month>mes_agendamento</month>
          <year>ano_agendamento</year>
          <start>
                 <hour>hora_inicial_agendamento</hour>
                 <min>minuto_inicial_agendamento</hour>
          </start>
          <end>
                 <hour>hora_final_agendamento</hour>
                 <min>minuto_final_agendamento</hour>
          </end>
    </schedule>
   </member>
  </contacts>
</request>
```

POST data em JSON:

```
{
  "contacts": [
             {
              "item": id_item1,
              "contact": contato1,
              "complement": complemento1,
              "phone_list": [
                          numero_telefone1_1,
                          numero_telefone1_2,
                          numero_telefone1_3,
                          numero_telefone1_4,
                          numero_telefone1_5
                          numero_telefone1_6,
                          numero_telefone1_7,
                          numero_telefone1_8,
                          numero_telefone1_9,
                          numero_telefone1_10
                 },
                 "schedule":
                 {       "day": dia_agendamento,
                         "month": mes_agendamento,
                         "year": ano_agendamento,
                         "start":
                         {       "hour": hora_inicial_agendamento,
```

```
                                        "min": minuto_inicial_agendamento
                                    }
                                    "end":
                                    {       "hour": hora_final_agendamento,
                                            "min": minuto_final_agendamento
                                    }
 }
                ...
                {
                "item": id_itemN,
                "contact": contatoN,
                "complement": complementoN,
                "phone_list": [
                          numero_telefoneN_1,
                          numero_telefoneN_2,
                          ...
                          numero_telefoneN_5
                        ]
              }
            ]
}
```

Onde:

- **id_item:** código identificador de um item para o cliente.
- **contato**: nome do contato (opcional).

- **complemento**: informação complementar do contato (opcional)
- **numero_telefone**: número do contato, sendo que cinco é a quantidade máxima de números por contato.

Resposta em caso de sucesso

```
    {
       "response":
       {
          "resource": "dialer",
          "command": "import_contacts",
          "status": "ok",
          "total": total,
          "count": contador,
          "initial_id": id_primeiro_contato
  }
}
```

Onde:

- **total**: Quantidade de contatos no mailling desse serviço.
- **contador**: Quantidade de contatos inseridos com sucesso.
- **id_primeiro_contato**: Identificador do primeiro contato inserido.

Resposta em caso de sucesso realizando a validação de números duplicados

```
   {
       "resource": "dialer",
```

```
    "command": "import_contacts",
    "status": "ok",
    "total": total,
    "count": contador,
    "initial_id": id_primeiro_contato,
    "errors": quantidade_de_erros,
    "error_list":
    [
        {
            "item": id_contato1,
            "contact": nome_contato
        },
         {
            "item": id_contato2,
            "contact": nome_contato2
        }
    ]
}
```

Onde:

- **total**: quantidade de contatos no mailling desse serviço.
- **contador**: quantidade de contatos inseridos com sucesso.
- **id_primeiro_contato**: identificador do primeiro contato inserido.
- **quantidade_de_erros**: quantidade de contatos com erro.
- **id_contatoN**: identificador do contato.

- **Nome_contatoN**: nome do contato.

11

CALLBACK

O seguinte conjunto de funcionalidades de ***Callback*** está disponibilizado para integração com o **Interact**:

- Escrita e leitura de parâmetros de configuração do *Callback*
- Ativação e desativação de coleta e geração de chamadas de *Callback*
- Inserção de contatos para *Callback*
- Limpeza de contatos pendentes de *Callback*

O uso destas funcionalidades depende de licença.

Os serviços disponibilizados pelo **Interact CTI** para *Callback* utilizam a filosofia REST e têm o seguinte formato geral:

interact_cti/v1.0/callback/metodo?parametros[&format= (json|xml)]

Onde:

- **metodo:** nome da requisição (comando).
- **parametros:** conjunto de pares nome-valor na forma prm=valor, separados por & (Ecomercial-ampersand).

- **format:** define o formato desejado para a resposta, a qual pode ser no formato JSON (.json) ou no formato (.xml). Na ausência desse parâmetro será utilizado o mesmo formato informado no header Content-type.

Observações:

- Na ausência do header **Content-type** será utilizado **XML** por *default*.
- Na ausência de parâmetros e de formato, o sinal de interrogação após método é desnecessário.
- As requisições aos serviços serão sempre por meio de métodos HTTP GET ou POST.
- O método GET é utilizado em todas as requisições que não enviam apenas parâmetros.
- O método POST é utilizado em todas as requisições que enviam pacotes de dados (além dos parâmetros).
- Os parâmetros enviados na URL deverão ser codificados no padrão URL encode.
- Os dados nos pacotes POST deverão ser codificados em UTF-8.
- O acesso se dará na porta 8582 (valor *default* que pode ser alterado por configuração), conforme exemplo abaixo:
- *http://XXX.XXX.XXX.XXX:8582/interact_cti/v1.0/callbanck/metodo*

Onde:

- **XXX.XXX.XXX.XXX é o** endereço IP do servidor **Interact CTI**

OBTENÇÃO DOS PARÂMETROS DE CONFIGURAÇÃO DE *CALLBACK* (GET_PARAMETERS)

Este comando retorna um conjunto de parâmetros de configuração de *Callback* de um determinado serviço no **Interact**.

Método HTTP GET

interact_cti/v1.0/callback/get_parameters?[service=nome_do_servico | entry_point=ponto_de_roteamento][&format=[xml | json]]

Onde:

- **nome_do_servico** é o nome de um serviço cadastrado no **Interact**;
- **ponto_de_roteamento** são as cifras do ramal associado ao serviço como ponto de roteamento de chamadas no **Interact**;
- **format** (opcional) é o formato desejado para a resposta. Caso não seja enviado, a resposta será em XML.

RESPONSE DATA em JSON:

```
{
        "response":
        {
            "enabled": [ true |false ],
            "mode": [ "manual | "auto" ],
```

```
            "source": origem_dos_contatos,
            "agents_availability": disponibilidade_de_agentes,
            "fails":
            {
                "busy":             { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "refused":          { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "invalid_number":   { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "refused_by_agent": { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "others":           { "wait": tempo_de_espera,
"retries": numero_de_retentativas }
            }
        }
    }
```

RESPONSE DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
    <response>
        <enabled> [ true |false ] </enabled>
        <mode> [ manual | auto ] </mode>
        <source> origem_dos_contatos </source>
        <agents_availability> disponibilidade_de_agentes
<agents_availability>
```

```
   <fails>
      <busy>
         <wait> tempo_de_espera </wait>
         <retries> numero_de_retentativas </retries>
      </busy>
      <refused>
         <wait> tempo_de_espera </wait>
         <retries> numero_de_retentativas </retries>
      </refused>
      <invalid_number>
         <wait> tempo_de_espera </wait>
         <retries> numero_de_retentativas </retries>
      </invalid_number>
      <refused_by_agent>
         <wait> tempo_de_espera </wait>
         <retries> numero_de_retentativas </retries>
      </refused_by_agent>
      <others>
         <wait> tempo_de_espera </wait>
         <retries> numero_de_retentativas </retries>
      </others>
   </fails>
</response>
```

Onde:

- **origem_dos_contatos** pode ser **"inbound"** para contatos internos, **"outbound"** para contatos externos ou **"both"** para ambos;
- **disponibilidade_de_agentes** é um número entre **10 e 100** e que indica a porcentagem de agentes livres para que sejam geradas chamadas de *Callback*;
- **tempo_de_espera** é o tempo em **minutos** que o sistema deverá esperar para retentar uma chamada de *Callback* que falhou.
- **numero_de_retentativas** é o número de vezes que o sistema deverá retentar uma chamada de *Callback* que falhou.

Observações:

- O parâmetro **enable** indica se o *Callback* está habilidado (**true**) ou desabilitado (**false**);
- O parâmetro **mode** indica se o *Callback* está no modo manual (**manual**) ou automático (**auto**);
- Os tipos de falha de geração são:
  - **busy**: destino da chamada está ocupado;
  - **refused**: destino da chamada não atendeu;
  - **invalid_number**: destino da chamada é um número inválido;
  - **refused_by_agent**: a chamada foi abandonada pelo agente;
  - **other**: demais casos de falha.

CONFIGURAÇÃO DOS PARÂMETROS DO *CALLBACK* (SET_PARAMETERS)

Este comando permite definir o conjunto de parâmetros de configuração de *Callback* para um determinado serviço no **Interact**.

Método HTTP POST

interact_cti/v1.0/callback/set_parameters?[service=nome_do_servico | entry_point=ponto_de_roteamento][&format=[xml | json]]

Onde:

- **nome_do_servico** é o nome de um serviço cadastrado no **Interact**;
- **format** (opcional) é o formato desejado para a resposta. Caso não seja enviado, a resposta será em XML.

POST DATA em JSON

```
{
        "enabled": [ true |false ],
        "mode": [ "manual | "auto" ],
        "source": origem_dos_contatos,
        "agents_availability": disponibilidade_de_agentes,
        "fails":
        {
```

```
                "busy":             { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "refused":          { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "invalid_number":   { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "refused_by_agent": { "wait": tempo_de_espera,
"retries": numero_de_retentativas },
                "others":           { "wait": tempo_de_espera,
"retries": numero_de_retentativas }
             }
        }
```

POST DATA em XML:

```
   <?xml version="1.0" encoding="UTF-8"?>
          <enabled> [ true |false ] </enabled>
          <mode> [ manual | auto ] </mode>
          <source> origem_dos_contatos </source>
          <agents_availability> disponibilidade_de_agentes
<agents_availability>
          <fails>
             <busy>
                <wait> tempo_de_espera </wait>
                <retries> numero_de_retentativas </retries>
             </busy>
             <refused>
```

```
      <wait> tempo_de_espera </wait>
<retries> numero_de_retentativas </retries>
   </refused>
   <invalid_number>
      <wait> tempo_de_espera </wait>
      <retries> numero_de_retentativas </retries>
   </invalid_number>
   <refused_by_agent>
      <wait> tempo_de_espera </wait>
      <retries> numero_de_retentativas </retries>
   </refused_by_agent>
   <others>
      <wait> tempo_de_espera </wait>
      <retries> numero_de_retentativas </retries>
   </others>
</fails>
```

Onde:

- **origem_dos_contatos** pode ser **"inbound"** para contatos internos, **"outbound"** para contatos externos ou **"both"** para ambos;
- **disponibilidade_de_agentes** é um número entre **10 e 100** e que indica a porcentagem de agentes livres para que sejam geradas chamadas de *Callback*;
- **tempo_de_espera** é o tempo em **minutos** que o sistema deverá esperar para retentar uma chamada de *Callback* que falhou.

- **numero_de_retentativas** é o número de vezes que o sistema deverá retentar uma chamada de *Callback* que falhou.

Observações:

- O parâmetro **enable** indica se o *Callback* está habilidado (**true**) ou desabilitado (**false**);
- O parâmetro **mode** indica se o *Callback* está no modo manual (**manual**) ou automático (**auto**);
- Os tipos de falha de geração são:
  - **busy**: destino da chamada está ocupado;
  - **refused**: destino da chamada não atendeu;
  - **invalid_number**: destino da chamada é um número inválido;
  - **refused_by_agent**: a chamada foi abandonada pelo agente;
  - **other**: demais casos de falha.

A resposta a este comando segue o padrão de respostas do **Interact CTI**.

ATIVAÇÃO DA COLETA DE CONTATOS E GERAÇÃO DE CHAMADAS (ACTIVATE)

Este comando permite ativar a coleta de contatos e a geração de chamadas de *Callback* para um determinado serviço no **Interact**.

Método HTTP GET

interact_cti/v1.0/callback/activate?[service=nome_do_servico | entry_point=ponto_de_roteamento][&action=acao][&format=[xml | json]]

Onde:

- **nome_do_servico** é o nome de um serviço cadastrado no **Interact;**
- **ponto_de_roteamento** são as cifras do ramal associado ao serviço como ponto de roteamento de chamadas no **Interact**;
- **acao (opcional)** pode ser **collect** para ativar a coleta de contatos, **call** para ativar a geração de chamadas de *Callback* ou **both** para ativar ambos. **Observação**: este parâmetro não tem validade se o *Callback* estiver configurado no modo automático (neste modo coleta e geração são ativados juntos automaticamente). Em modo manual, se não for enviado, será considerado o valor *default* **both**;
- **format** (opcional) é o formato desejado para a resposta. Caso não seja enviado, a resposta será em XML.

A resposta a este comando segue o padrão de respostas do **Interact CTI**.

DESATIVAÇÃO DA COLETA DE CONTATOS E GERAÇÃO DE CHAMADAS (DEACTIVATE)

Este comando permite desativar a coleta de contatos e a geração de chamadas de *Callback* para um determinado serviço no **Interact**.

Método HTTP GET

interact_cti/v1.0/callback/deactivate?[service=nome_do_servico | entry_point=ponto_de_roteamento][&action=acao][&format=[xml | json]]

Onde:

- **nome_do_servico** é o nome de um serviço cadastrado no **Interact**;
- **ponto_de_roteamento** são as cifras do ramal associado ao serviço como ponto de roteamento de chamadas no **Interact**;
- **acao** pode ser **collect** para desativar a coleta de contatos, **call** para desativar a geração de chamadas de *Callback* ou **both** para desativar ambos. **Observação**: este parâmetro não tem validade se o *Callback* estiver configurado no modo automático (neste modo coleta e geração são desativados juntos automaticamente);
- **format** (opcional) é o formato desejado para a resposta. Caso não seja enviado, a resposta será em XML.

A resposta a este comando segue o padrão de respostas do **Interact CTI**.

INSERÇÃO DE CONTATO PARA *CALLBACK* (ADD_CONTACT)

Este comando permite definir o conjunto de parâmetros de configuração de *Callback* para um determinado serviço no **Interact**.

Método HTTP POST

interact_cti/v1.0/callback/add_contact?[service=nome_do_servico | entry_point=ponto_de_roteamento][&format=[xml | json]]

Onde:

- **nome_do_servico** é o nome de um serviço cadastrado no **Interact**;
- **ponto_de_roteamento** são as cifras do ramal associado ao serviço como ponto de roteamento de chamadas no **Interact**;
- **format** (opcional) é o formato desejado para a resposta. Caso não seja enviado, a resposta será em XML.

A resposta a este comando segue o padrão de respostas do **Interact CTI**.

POST DATA em JSON

```
{
    "name": nome_do_contato,
    "complement": complemento,
    "number": numero_telefonico,
    "schedule":
    { "day": dia_agendamento,
      "month": mes_agendamento,
      "year": ano_agendamento,
      "start":
      { "hour": hora_inicial_agendamento,
```

```
      "min": minuto_inicial_agendamento
    },
    "end":
    { "hour": hora_final_agendamento,
      "min": minuto_final_agendamento
    }
  }
}
```

POST DATA em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<name> nome_do_contato </name>
<complement> complemento </complement>
<number> numero_telefonico </number>
<schedule>
  <day> dia_agendamento </day>
  <month> mes_agendamento </month>
  <year> ano_agendamento </year>
  <start>
    <hour> hora_inicial_agendamento </hour>
    <min> minuto_inicial_agendamento </hour>
  </start>
  <end>
    <hour> hora_final_agendamento </hour>
```

```
        <min> minuto_final_agendamento </hour>
      </end>
    </schedule>
```

Onde:

- **nome_do_contato** é o nome do contato que se deseja adicionar ao *Callback* de um serviço;
- **complemento (opcional)** é uma informação complementar sobre o contato que se deseja adicionar ao *Callback* de um serviço;
- **numero_telefonico** é o número para o qual o sistema deverá gerar a chamada de call-back.

O agendamento do contato é opcional. Para efetuá-lo, é necessário informar no objeto *schedule* o dia, o mês e o horário inicial da discagem. O horário final é opcional. Os parâmetros para agendamento são:

- **dia_agendamento**: dia do agendamento (1-31);
- **mes_agendamento**: mês do agendamento (1-12);
- **ano_agendamento**: ano do agendamento;
- **hora_inicial_agendamento**: hora de início da faixa de agendamento (0-23);
- **minuto_inicial_agendamento**: minuto de início da faixa de agendamento (0-59);
- **hora_final_agendamento**: hora de término da faixa de agendamento (0-23);
- **minuto_final_agendamento**: minuto de término da faixa de agendamento (0-59).

A resposta a este comando segue o padrão de respostas do **Interact CTI**.

LIMPEZA DE CONTATOS PENDENTES DE *CALLBACK* (CLEAR)

Este comando permite eliminar (limpar) o conjunto de contatos pendentes de *Callback* para um determinado serviço no **Interact**.

Método HTTP GET.

interact_cti/v1.0/callback/clear?[service=nome_do_servico | entry_point=ponto_de_roteamento][&format=[xml | json]]

Onde:

- **nome_do_servico:** é o nome de um serviço cadastrado no **Interact**;
- **ponto_de_roteamento:** são as cifras do ramal associado ao serviço como ponto de roteamento de chamadas no **Interact**;
- **format** (opcional): é o formato desejado para a resposta. Caso não seja enviado, a resposta será em XML.

A resposta a este comando segue o padrão de respostas do **Interact CTI**.

12

VARIÁVEIS MIB (MANAGEMENT INFORMATION BASE)

O recurso **mibs** do **Interact CTI** possibilita coletar valores de variáveis úteis para a manutenção de uma base de informações gerenciais.

As variáveis disponibilizadas para consulta pelo **Interact CTI** estão tabuladas a seguir:

ID da variável	Descrição	Tipo	Faixa
4687.22.1	**Atendentes**		
4687.22.1.1	**Licenças de atendentes**		
4687.22.1.1.1	Quantidade de licenças de interface de **MultiAgent** existentes.	Inteiro	0 a n
4687.22.1.1.2	Quantidade de licenças de interface de **MultiAgent** em uso.	Inteiro	0 a n
4687.22.1.1.3	Quantidade de licenças de atendentes habilitados para mídia Voz existentes.	Inteiro	0 a n
4687.22.1.1.4	Quantidade de licenças de atendentes habilitados para mídia Voz em uso.	Inteiro	0 a n
4687.22.1.1.5	Quantidade de licenças de atendentes habilitados para a mídia *Chat* existentes.	Inteiro	0 a n
4687.22.1.1.6	Quantidade de licenças de atendentes habilitados para a mídia *Chat* em uso.	Inteiro	0 a n
4687.22.1.1.7	Quantidade de licenças de atendentes habilitados para a mídia *E-mail* existentes.	Inteiro	0 a n
4687.22.1.1.8	Quantidade de licenças de atendentes habilitados para a mídia *E-mail* em uso.	Inteiro	0 a n
4687.22.1.2	**Estatísticas de atendentes**		
4687.22.1.2.1	Quantidade de atendentes logados.	Inteiro	0 a n

4687.22.2	**Estatísticas e Supervisão**		
4687.22.2.2	**Estatísticas de chamadas**		
4687.22.2.2.1	Quantidade de chamadas da mídia Voz em andamento.	Inteiro	0 a n
4687.22.2.2.2	Quantidade de chamadas da mídia *Chat* em andamento.	Inteiro	0 a n
4687.22.2.2.3	Quantidade de chamadas da mídia *E-mail* em andamento.	Inteiro	0 a n
4687.22.2.2.4	Quantidade de chamadas da mídia Voz em recebidas.	Inteiro	0 a n
4687.22.2.2.5	Quantidade de chamadas da mídia *Chat* em recebidas.	Inteiro	0 a n
4687.22.2.2.6	Quantidade de chamadas da mídia *E-mail* em recebidas.	Inteiro	0 a n
4687.22.2.2.7	Quantidade de chamadas da mídia Voz geradas.	Inteiro	0 a n
4687.22.2.2.8	Quantidade de chamadas da mídia *Chat* geradas.	Inteiro	0 a n
4687.22.2.2.9	Quantidade de chamadas da mídia *E-mail* geradas.	Inteiro	0 a n
4687.22.2.2.10	Quantidade de chamadas da mídia *E-mail* respondidas.	Inteiro	0 a n
4687.22.2.2.11	Quantidade de chamadas da mídia Voz em fila de entrada.	Inteiro	0 a n

4687.22.2.2.12	Quantidade de chamadas da mídia C*hat* em fila de entrada.	Inteiro	0 a n
4687.22.2.2.13	Quantidade de chamadas da mídia *E-mail* em fila de entrada.	Inteiro	0 a n
4687.22.2.2.14	Quantidade de chamadas da mídia *E-mail* em fila de saída.	Inteiro	0 a n
4687.22.3	**Serviços Receptivos**		
4687.22.3.1	**Licenças de serviços receptivos**		
4687.22.3.2	**Estatísticas de serviços receptivos**		
4687.22.3.2.1	Quantidade de serviços receptivos em funcionamento.	Inteiro	0 a n
4687.22.3.2.2	Quantidade de serviços receptivos com mídia *E-mail* habilitada.	Inteiro	0 a n
4687.22.3.2.3	Quantidade de serviços receptivos com envio de conversa de *Chat* por *e-mail.*	Inteiro	0 a n
4687.22.3.2.4	Quantidade de serviços receptivos com falha na recepção de *e-mails.*	Inteiro	0 a n
4687.22.3.2.5	Quantidade de serviços receptivos com falha no envio de *e-mails.*	Inteiro	0 a n
4687.22.4	**Serviços Ativos**		
4687.22.4.1	**Licenças serviços ativos**		
4687.22.4.1.1	Quantidade de licenças de chamadas em campanhas do tipo POWER DIALING existentes.	Inteiro	0 a n

4687.22.4.1.2	Quantidade de licenças de chamadas em campanhas do tipo POWER DIALING em uso.	Inteiro	0 a n
4687.22.4.1.3	Quantidade de licenças de chamadas em campanhas do tipo PREDITIVO existentes.	Inteiro	0 a n
4687.22.4.1.4	Quantidade de licenças de chamadas em campanhas do tipo PREDITIVO em uso.	Inteiro	0 a n
4687.22.4.1.5	Quantidade de licenças de chamadas em campanhas do tipo PREVIEW existentes.	Inteiro	0 a n
4687.22.4.1.6	Quantidade de licenças de chamadas em campanhas do tipo PREVIEW em uso.	Inteiro	0 a n
4687.22.4.1.7	Quantidade de licenças de chamadas em campanhas do tipo AGENT READY existentes.	Inteiro	0 a n
4687.22.4.1.8	Quantidade de licenças de chamadas em campanhas do tipo AGENT READY em uso.	Inteiro	0 a n
4687.22.4.1.9	Quantidade de licenças de chamadas em campanhas do tipo TESTE DE LOTE existentes.	Inteiro	0 a n
4687.22.4.1.10	Quantidade de licenças de chamadas em campanhas do tipo TESTE DE LOTE em uso.	Inteiro	0 a n
4687.22.4.2	**Estatísticas de serviços ativos**		

4687.22.4.2.1	Quantidade de serviços ativos em funcionamento.	Inteiro	0 a n
4687.22.5	**ChatClient**		
4687.22.5.1	**Licenças de ChatClient**		
4687.22.5.1.1	Quantidade de interfaces do **ChatClient** existentes.	Inteiro	0 a n
4687.22.5.1.2	Quantidade de interfaces do **ChatClient** em uso.	Inteiro	0 a n
4687.22.5.1.3	Quantidade de licenças de chamadas simultâneas de *Chat* com vídeo existentes.	Inteiro	0 a n
4687.22.5.1.4	Quantidade de licenças de chamadas simultâneas de *Chat* com vídeo em uso.	Inteiro	0 a n
4687.22.6	**CTI**		
4687.22.6.1	**Licenças do CTI**		
4687.22.6.1.1	Quantidade de licenças de sincronismo de tela para o **Interact CTI** existentes.	Inteiro	0 a n
4687.22.6.1.2	Quantidade de licenças de sincronismo de tela para o **Interact CTI** em uso.	Inteiro	0 a n
4687.22.6.1.3	Licença para sincronismo de tela de um número ilimitado de agentes por meio do **Interact CTI** existente.	Inteiro	1 (habilitado) 2 (desabilitado)
4687.22.6.1.4	Licença para sincronismo de tela de um número ilimitado de agentes por meio do	Inteiro	1 (habilitado) 2 (desabilitado)

	Interact CTI em uso.		
4687.22.6.1.5	Quantidade de licenças de controle de agentepara o **Interact CTI** existentes.	Inteiro	0 a n
4687.22.6.1.6	Quantidade de licenças de controle de agentepara o **Interact CTI** em uso.	Inteiro	0 a n
4687.22.6.1.7	Quantidade de licenças de supervisão para o **Interact CTI** existentes.	Inteiro	0 a n
4687.22.6.1.8	Quantidade de licenças de supervisão para o **Interact CTI** em uso.	Inteiro	0 a n
4687.22.6.1.9	Licença para possibilitar o uso do recurso gravador por meio do **Interact CTI** existentes.	Inteiro	1 (habilitado) 2 (desabilitado)
4687.22.6.1.10	Licença para possibilitar o uso do recurso gravador por meio do **Interact CTI** em uso.	Inteiro	1 (habilitado) 2 (desabilitado)
4687.22.6.1.11	Quantidade de licenças para controle da quantidade de serviços tipo **Ativo** do **Interact** que podem ser monitorados através do **CTI** existentes.	Inteiro	0 a n
4687.22.6.1.12	Quantidade de licenças para controle da quantidade de serviços tipo **Ativo** do **Interact** que podem ser monitorados através do **CTI** em uso.	Inteiro	0 a n
4687.22.6.1.13	Licença para controle dos serviços tipo **Ativo** do **Interact** existente.	Inteiro	1 (habilitado) 2 (desabilitado)
4687.22.6.1.14	Licença para controle dos serviços tipo	Inteiro	1 (habilitado)

	Ativo do **Interact** em uso.		2 (desabilitado)
4687.22.6.1.15	Quantidade de licenças de monitoração de configuração do Interact por meio do **Interact CTI** em uso.	Inteiro	0 a n
4687.22.6.1.16	Quantidade de licenças de monitoração de configuração do Interact por meio do **Interact CTI** existentes.	Inteiro	0 a n
4687.22.7	Integrações		
4687.22.7.1	Licenças rede social		
4687.22.7.1.1	Licença para habilitar integração com redes sociais existente.	Inteiro	1 (habilitado) 2 (habilitado)
4687.22.7.1.2	Licença para habilitar integração com redes sociais em uso.	Inteiro	1 (habilitado) 2 (habilitado)

Os serviços disponibilizados pelo **Interact CTI** para ***Management Information Base (MIBs)*** por meio de **REST** têm o seguinte formato geral:

interact_cti/v1.0/mibs/metodo?parametros[&format= (json|xml)]

Onde:

- **metodo**: nome da requisição (comando);
- **parametros**: conjunto de pares nome-valor na forma prm=valor, separados por & (Ecomercial -ampersand);
- **format**: define o formato desejado para a resposta, que pode ser no formato JSON (.json) ou no formato (.xml).

CONSULTA DE VARIÁVEIS MIB (COMANDO GET)

O comando *get* retorna um conjunto de variáveis **MIB** solicitadas. Para solicitar as variáveis **MIBs** ao **Interact CTI**, deve ser informado um parâmetro **variables** na URL, com uma lista de variáveis separadas por vírgulas, conforme sintaxe a seguir:

interact_cti/v1.0/mibs/get?variables=lista_de_variaveis[&format= (json|xml)]

Onde **lista_de_variaveis** é uma lista com ids das variáveis desejadas separadas por vírgulas.

Exemplo1:

interact_cti/v1.0/mibs/get?variables=4687.22.1.1.6,4687.22.2.2.2,4687.22.6.1.6

Neste exemplo estão sendo solicitadas as variáveis com ids **4687.22.1.1.6**, **4687.22.2.2.2** e **4687.22.6.1.6**.

As variáveis também podem ser solicitadas em blocos por meio da utilização de caracteres curinga (***** ou **?**). O curinga ***** deve ser utilizado ao final de um padrão, como em **4687.22.1.1.***, já o curinga **?** pode ser utlizado em qualquer posição e indica que qualquer caractere pode estar ali presente.

Exemplo2:

interact_cti/v1.0/mibs/get?variables=4687.22.1.1.*,4687.22.5.1.2,4687.22.6.1.2

O exemplo citado acima solicita todas as variáveis do ramo **4687.22.1.1** (variáveis com ids de **4687.22.1.1.**1 a **4687.22.1.1.1**) e também as as variáveis **4687.22.5.1.2** e **4687.22.6.1.2**.

Exemplo3:

interact_cti/v1.0/mibs/get?variables=4687.22.?.1.1

O exemplo citado acima solicita todas as variáveis que possuem um padrão que inicia por **4687.22.** seguido por um caractere qualquer e, por fim, seguido por **.1.1**. Serão retornadas as variáveis com ids de **4687.22.1.1.1**, **4687.22.2.1.1**, **4687.22.3.1.1** e **4687.22.6.1.1**.

Formato de resposta

O conjunto de variáveis solicitado é retornado em um vetor ***variables*** em que cada membro é uma variável identificada pelo ID no campo ***key*** e com seu valor no campo ***value***.

O formato da resposta ao comado ***get*** é apresentado a seguir para a solicitação do exemplo 3.

RESPONSE DATA em JSON:


```
{
  "response":
   {
      "resource": "mibs",
      "command": "get",
```

```
    "status": "ok",
    "variables":
    [
      {
        "key": "4687.22.1.1.1",
        "value": 100
      },
      {
        "key": "4687.22.4.1.1",
        "value": 1000
      },
      {
        "key": "4687.22.5.1.1",
        "value": 1000
      },
      {
        "key": "4687.22.6.1.1",
        "value": 1000
      }
    ]
  }
}
```


RESPONSE DATE em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <resource>mibs</resource>
  <command>get</command>
  <status>ok</status>
  <variables>
    <member>
      <key>4687.22.1.1.1</key>
      <value>100</value>
    </member>
    <member>
      <key>4687.22.4.1.1</key>
      <value>1000</value>
    </member>
    <member>
      <key>4687.22.5.1.1</key>
      <value>1000</value>
    </member>
    <member>
      <key>4687.22.6.1.1</key>
      <value>1000</value>
    </member>
  </variables>
</response>
```

13

PADRÃO DE RESPOSTAS

Resposta para o comando em caso de sucesso:

Em JSON:

```
{"response":
  {"command": nome_do_commando,
   "status":"ok"
  }
}
```

Em XML:

```
<response>
  <command>nome_do_commando</command>
  <status>ok</status>
 </response>
```

Resposta em Caso de Erro

Em JSON:

```
{"response":
 {"command": nome_do_commando,
  "status":"error",
  "error_type":tipo_de erro
  “error_msg”: mensagem_de_erro
 }
}
```

Em XML:

```
<response>
  <command>nome_do_commando</command>
  <status>error</status>
  <error_type>tipo_de_erro</error_type>
  <error_msg>mensagem_de_erro</error_msg>
</response>
```

Onde:

- **nome_do_comando**: nome do comando executado;
- **tipo_de_erro**:
  - *empty_data: era esperado um pacote de dados e ele veio vazio;*
  - *invalid_data: dados recebidos não são válidos;*
  - *missing_data: atributo requerido pelo comando não foi enviado;*
  - *not_found: item requisitado não foi encontrado;*

- *command_failed: erro na execução do comando;*
- *command_denied:* comando negado por falta de privilégio para sua execução;
- *no_license*: sem licença para uso do recurso.
- *duplicated_number*: comando negado pois o número que se pretende inserir ou atualizar já existe (pode ocorrer em inserção de registros de **discador** e **call-back**)

- **mensagem_de_erro:** só será enviada na resposta do call_generate do recurso **ChatClient** e quando o atributo "*error_msg"* for "*out_of_operation".* A mensagem que será enviada é a mesma de fora de horário de serviço configurada na mídia de *Chat.*