Vonix Integração - Flash

A integração dos sistemas ocorre através de um componente Flash adicionado a uma página Web. Através de chamadas a métodos Javascript, é possível enviar comandos ao componente. O componente também observa eventos referentes ao agente, disparando métodos no Javascript no momento em que os mesmos ocorrem no servidor.

O componente permite a uma página WEB disparar as seguintes ações de agente:

- Iniciar uma chamada
- Pausar
- Despausar
- Adicionar uma tag a uma chamada
- Solicitar o status do agente (Em chamada, Pausado, etc)

O componente também observa os seguintes eventos de agente:

- Agente recebe uma chamada (chamada toca no ramal do agente)
- Agente atende esta chamada
- Chamada não atendida é abandonada pelo chamador ou oferecida para outro agente
- Agente inicia uma chamada (chamada toca para o telefone de destino)
- Chamada é atendida pelo destino
- Chamada não é atendida pelo destino (ocupado, número inválido, etc)
- Desligamento de chamada atendida (tanto receptiva quanto ativa)
- Agente entra em pausa
- Agente retorna da pausa

Utilização

Para que o componente seja inicializado e os eventos do agente possam ser observados, é necessário que a página HTML chame o método Javascript:

```
VonixFlashJS.connect(action_id, host, agent_code);
```

Onde:

- **action_id** = uma string definida pelo usuário que serve como código identificador de qualquer ação enviada pela página. Muitos eventos retornam com esse valor, indicando que aquele evento foi o resultado de uma ação gerada com tal string.
- **host** = o endereço do servidor de integração.
- **agent_code** = matrícula do agente.

Exemplo:

```
VonixFlashJS.connect("conexao_inicial", "192.168.0.1", "1004");
```

Quando a conexão for efetuada com sucesso, o método:

```
VonixFlashJS.onConnect(date, action_id);
```

será disparado, onde:

- **date** = data e hora de conexão (formato Date)
- **action_id** = string enviada no campo action_id do método **connect()**.

Em caso de falha de conexão, o **onConnect** não será chamado.

Após uma conexão bem sucedida, o componente passa a observar os eventos do agente, e disparar as funções observadoras (que iniciam com "on", por exemplo, "onDial") presentes no módulo **VonixFlashJS**. Estes métodos deverão ser preenchidos com o código da aplicação Web que irá utilizar o componente.

Ações

Para iniciar uma chamada, a página deve chamar o método:

```
VonixFlashJS.doDial(action_id, to, name, queue, billing_group_id)
```

onde:

- **action_id** = código de identificação definido pelo desenvolvedor que será retornado no evento onDial (formato String)
- **to** = número do telefone de destino, apenas dígitos (formato String)
- **name** = nome do cliente (formato String)
- **queue** = fila que originará a chamada (formato String)
- **billing_group_id** = campanha ou centro de custo (formato Number)

Para pausar o agente, a página deve chamar o método:

```
VonixFlashJS.doPause(action_id, reason)
```

onde:

- **action_id** = código de identificação definido pelo desenvolvedor que será retornado no evento onPause (formato String)
- **reason** = código do motivo de pausa (formato Number)

Para despausar o agente, a página deve chamar o método:

```
VonixFlashJS.doUnpause(action_id)
```

onde:

- **action_id** = código de identificação definido pelo desenvolvedor que será retornado no evento onUnpause (formato String)

Para adicionar uma tag a uma chamada, a página deve chamar o método:

```
VonixFlashJS.doTag(action_id, call_id, tag)
```

onde:

- **action_id** = código de identificação definido pelo desenvolvedor (formato String)

- **call_id** = código identificador da chamada que se deseja adicionar a tag (formato String)
- **tag** = nome da tag que será adicionada à chamada (formato String)

Para verificar o status atual do agente, a página deve chamar o método:

```
VonixFlashJS.doStatus(action_id, queue)
```

onde:
- **action_id** = código de identificação definido pelo desenvolvedor (formato String)
- **queue** = fila do agente (formato String)

O retorno do status do agente ocorre no evento **onStatus()** (veja sintaxe na seção Eventos).

Para verificar a versão do componente Flash, a página deve chamar o método:

```
VonixFlashJS.doVersion(action_id)
```

onde:
- **action_id** = código de identificação definido pelo desenvolvedor (formato String)

O retorno da versão do componente ocorre no evento **onVersion()** (veja sintaxe na seção Eventos).

Eventos

Quando o agente origina uma chamada, o método:

```
VonixFlashJS.onDial(call_id, date, queue, from, to, callfilename, action_id)
```

é disparado automaticamente, onde:
- **call_id** = código identificador da chamada (formato String)
- **date** = data da chamada (formato Date)
- **queue** = fila da chamada (formato String)
- **from** = número do telefone que originou a chamada (formato String)
- **to** = número do telefone de destino da chamada (formato String)
- **callfilename** = nome do arquivo WAV de gravação da chamada (formato String)
- **action_id** = código de identificação da ação, caso ela tenha sido originada do componente (formato String)

Se essa chamada não for completada, o método:

```
VonixFlashJS.onDialFailure(call_id, date, cause_id, cause_description)
```

é disparado automaticamente, onde:
- **call_id** = código identificador da chamada (formato String)
- **date** = data do descarte da chamada (formato Date)
- **cause_id** = código da causa do não completamento (formato Number)
- **cause_description** = descrição da causa do não completamento (formato String)

Se essa chamada for atendida, o método:

```
VonixFlashJS.onDialAnswer(call_id, date)
```

é disparado automaticamente, onde:
- **call_id** = código identificador da chamada (formato String)

- **date** = data do atendimento da chamada

Ao desligar essa chamada atendida, o método:

```
VonixFlashJS.onHangUp(call_id, date, cause_id, cause_description)
```

é disparado automaticamente, onde:
- **call_id** = código identificador da chamada (formato String)
- **date** = data do desligamento (formato Date)
- **cause_id** = código da causa do desligamento (formato Number)
- **cause_description** = descrição da causa do desligamento (formato String)

Quando o agente recebe uma chamada (o ramal do agente começa a tocar), o método:

```
VonixFlashJS.onReceive(call_id, date, queue, from, to, callfilename,
action_id)
```

é disparado automaticamente, onde:
- **call_id** = código identificador da chamada (formato String)
- **date** = data do recebimento da chamada (formado Date)
- **queue** = fila em que a chamada está sendo recebida (formato String)
- **from** = número de telefone que originou a chamada; número do cliente (formato String)
- **to** = número de telefone de destino da chamada; geralmente o ramal do agente (formato String)
- **callfilename** = nome do arquivo de som com a gravação da chamada (formato String)
- **action_id** = código de identificação da ação, caso ela tenha sido originada do componente (formato String)

Quando o agente atende esta chamada que está tocando para ele, o método:

```
VonixFlashJS.onReceiveAnswer(call_id, date, waiting_seconds)
```

é disparado automaticamente, onde:
- **call_id** = código identificador da chamada (formato String)
- **date** = data do atendimento da chamada (formato Date)
- **waiting_seconds** = o número de segundos que a chamada ficou aguardando ser atendida (formato Number)

Ao desligar essa chamada atendida, o método:

```
VonixFlashJS.onHangUp(call_id, date, cause_id, cause_description)
```

é disparado automaticamente. (O mesmo método é disparado tanto para chamadas ativas quanto receptivas.)

Quando a chamada está tocando para o agente e não é atendida (por abandono ou oferecimento a outro agente), o método:

```
VonixFlashJS.onReceiveFailure(call_id, date, ringing_seconds)
```

é disparado automaticamente, onde:
- **call_id** = código identificador da chamada (formato String)
- **date** = data de falha de recebimento da chamada (formato Date)

- **ringing_seconds** = o número de segundos que a chamada ficou aguardando ser atendida (formato Number)

Quando o agente entra em pausa, o método:

```
VonixFlashJS.onPause(date, reason, action_id)
```

é disparado automaticamente, onde:
- **date** = data e hora em que o agente entrou em pausa (formato Date)
- **reason** = código do motivo de pausa (formato Number)
- **action_id** = código de identificação da ação, caso ela tenha sido originada do componente (formato String)

Quando o agente retorna de uma pausa, o método:

```
VonixFlashJS.onUnpause(date, reason, pause_seconds, action_id)
```

é disparado automaticamente, onde:
- **date** = data e hora em que o agente retornou da pausa (formato Date)
- **reason** = código do motivo de pausa (formato Number)
- **pause_seconds** = o número de segundos em que o agente ficou em pausa (formato Number)
- **action_id** = código de identificação da ação, caso ela tenha sido originada do componente (formato String)

Quando o componente retorna algum erro, o método:

```
VonixFlashJS.onError(action_id, message)
```

é disparado automaticamente, onde:
- **action_id** = código de identificação da ação, caso ela tenha sido originada do componente (formato String)
- **message** = mensagem de erro reportada pelo componente (formato String)

Quando o componente retorna o status do agente (solicitado através da ação **doStatus()**), o método:

```
VonixFlashJS.onStatus(status, location, action_id)
```

é disparado, onde:
- **status** = status do agente *(veja tabela no final deste manual)*
- **location** = número do ramal do agente
- **action_id** = código de identificação da ação, caso ela tenha sido originada do componente (formato String)

Para teste ou depuração, o método:

```
VonixFlashJS.onLog(message)
```

é disparado automaticamente a cada vez que o componente envia um comando ao componente Javascript, onde:
- **message** = mensagem enviada pelo componente.

Estas mensagens são apenas informativas e devem ser utilizadas apenas para depuração ou teste de funcionalidade do componente.

Quando o componente retorna a versão do mesmo (solicitado através da ação **doVersion()**), o método:

```
VonixFlashJS.onVersion(action_id, version)
```

é disparado, onde:

- **action_id** = código de identificação da ação, caso ela tenha sido originada do componente (formato String)
- **version** = versão do componente (formato String)

Lembrando que as ações e eventos apenas estarão disponíveis após o evento **onConnect()** ser disparado pelo componente.

Controle de Erros

O componente retornará possíveis erros disparando o método:

```
VonixFlashJS.onError(action_id, message);
```

onde:

- **action_id** = nos casos em que o erro gerado é devido à uma ação enviada, será a string enviada no campo action_id da ação.
  Em alguns casos, esse ID retornado pode ser próprio do componente. Os IDs próprios são:
  "_NOT_CONNECTED" (quando o componente envia uma ação sem conectar no servidor antes); e
  "_STOMP_ERROR" (quando ocorre erro no envio de mensagens ao servidor).
- **message** = mensagem de erro retornada pelo componente/servidor.

Instalação

O componente pode ser instalado em qualquer página HTML. A versão do plugin Flash a ser utilizada deve ser a 9 ou superior.

Passo 1: Inserir na tag <HEAD> do código HTML o seguinte parâmetro (substituir se já existente):

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Passo 2. Inserir na tag <HEAD> do código HTML o seguinte script:

```
<script type="text/javascript">
  <!--

  var VonixFlashJS = {

    /* Base methods - Do not modify */

    flashObject: function() {
```

```
          return document.getElementById("VonixFlash");
        },

        connect: function(action_id, host, agent_code) {
          return this.flashObject().connect(action_id, host, agent_code);
        },

        /* Actions - Do not modify */

        doDial: function(action_id, to, name, queue, billing_group_id) {
          this.flashObject().dial(action_id, to, name, queue,
billing_group_id);
        },
        doPause: function(action_id, reason) {
          this.flashObject().pause(action_id, reason);
        },
        doUnpause: function(action_id) {
          this.flashObject().unpause(action_id);
        },
        doTag: function(action_id, call_id, tag) {
          this.flashObject().tag(action_id, call_id, tag);
        },
        doStatus: function(action_id, queue) {
          this.flashObject().status(action_id, queue);
        },
        doVersion: function(action_id) {
          this.flashObject().version(action_id);
        },

        /* Events - Place business code inside the methods below */

        onConnect: function(date, action_id) {

        },
        onDial: function(call_id, date, queue, from, to, callfilename,
action_id) {

        },
        onDialAnswer: function(call_id, date) {

        },
        onDialFailure: function(call_id, date, cause_id, cause_description) {

        },
        onHangUp: function(call_id, date, cause_id, cause_description) {

        },
        onReceive: function(call_id, date, queue, from, to, callfilename,
action_id) {

        },
```

```
        onReceiveAnswer: function(call_id, date, waiting_seconds) {

        },
        onReceiveFailure: function(call_id, date, ringing_seconds) {

        },
        onPause: function(date, reason, action_id) {

        },
        onUnpause: function(date, reason, pause_seconds, action_id) {

        },
        onStatus: function(status, location, action_id) {

        },
        onError: function(action_id, message) {

        },
        onVersion: function(action_id, version) {

        },
        onLog: function(message) {

        }
      }

      //-->
    </script>
```

Passo 3: inserir o objeto "**VonixFlash**" dentro da tag <BODY> do código HTML:

```
    <object id="VonixFlash" type="application/x-shockwave-flash"
data="vonix_flash.swf" width="0" height="0">
      <param name="movie" value="vonix_flash.swf" />
      <param name="allowScriptAccess" value="always" />
    </object>
```

O arquivo **vonix_flash.swf** deve estar no mesmo diretório do arquivo HTML. Caso não esteja, o caminho completo deve ser especificado nos dois campos sublinhados acima.

Tabelas de id/dados dos retornos

Motivos de Pausa

1 - Avaliação
2 - Treinamento

3 - Lanche
4 - Toalete
5 - Ginástica Laboral
6 - Erro no Sistema

Causas de Término de Chamada

Obs.: estes códigos são enviados pela operadora.

1 - Número vago (não existe)
17 - Ocupado
18 - Usuário não atende
19 - Usuário não atende (timeout)
21 - Chamada rejeitada pela operadora
22 - Número de destino mudou
28 - Formato de número inválido
34 - Congestionamento, sem canais vagos
38 - Número de destino com problema de rede
41 - Falha temporária de rede na operadora
127 - Falha de interconexão entre operadoras

Status de Agente

ONLINE - Agente conectado e disponível
PAUSED - Agente em pausa
RINGING - Agente discando um número ou recebendo uma chamada
ONTHEPHONE - Agente ao telefone
OFFLINE - Agente desconectado