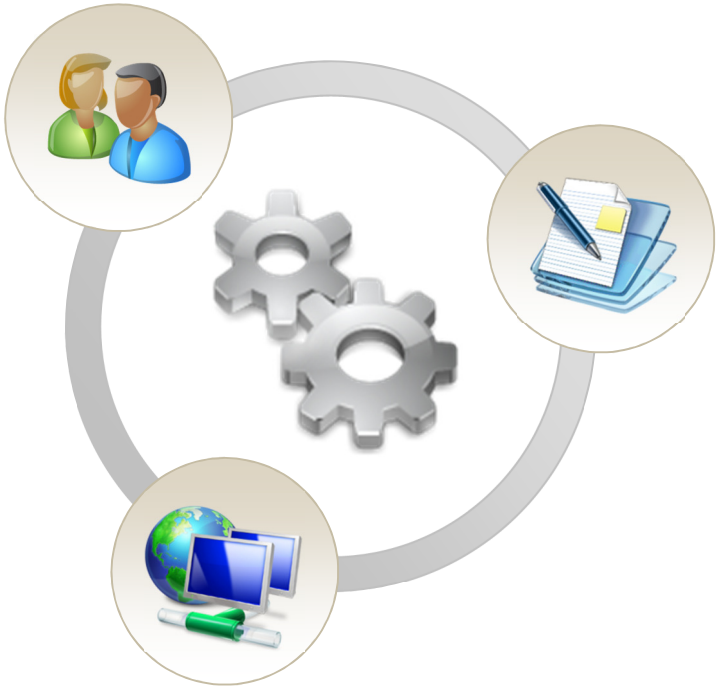
Manual de Integração

JavaScript e SignalR



Sumário

Objetivos

O objetivo desta documentação é orientar o desenvolvedor sobre como integrar com o API da Talk Telecom, descrevendo as funcionalidades, os métodos a serem utilizados, listando informações a serem enviadas e recebidas, e provendo exemplos.

O mecanismo de integração com o EPBX é simples, de modo que apenas conhecimentos intermediários em linguagem de programação para Web, requisições HTTP/HTTPS e manipulação de arquivos JSON, são necessários para implantar a solução com sucesso.

Nesse manual você encontrará a referência sobre todas as operações disponíveis na API. Estas operações devem ser executadas utilizando um Token que será melhor detalhado ao longo da documentação.

Ambiente Integração

Ambiente Manager: http://integracao.epbx.com.br/#/login
Conta SIP: homologação.epbx.com.br (Porta 5060)
Usuário e senha: Serão fornecidos por E-mail.

Glossário

Para facilitar o entendimento, listamos abaixo um pequeno glossário com os principais termos relacionados a integração do CRM e Discador.

- **Login (Token):** processo para assegurar que todas as requisições feitas estão seguras baseada na autenticação do usuário na plataforma. A integração para executar qualquer comando precisa que o Token seja válido.
- **SignalR**: Biblioteca Asp.Net que auxilia a implementação de funcionalidades Real-time. Possibilitando a comunicação assíncrona entre o servidor e a aplicação cliente. Atraves de uma conexão websocket o servidor consegue invocar métodos no cliente, implementando desta forma a comunicação em duas vias.
- **WebApi**: Padrão de implementação de web services (REST), utilizando o protocolo Http e troca de informações em Json e Xml.

Visão Geral

O EPBX é a implementação de uma central telefônica PBX (Private Branch eXchange) em software. O EPBX foi originalmente desenvolvido para Windows, mas atualmente pode ser instalado e executado no Linux.

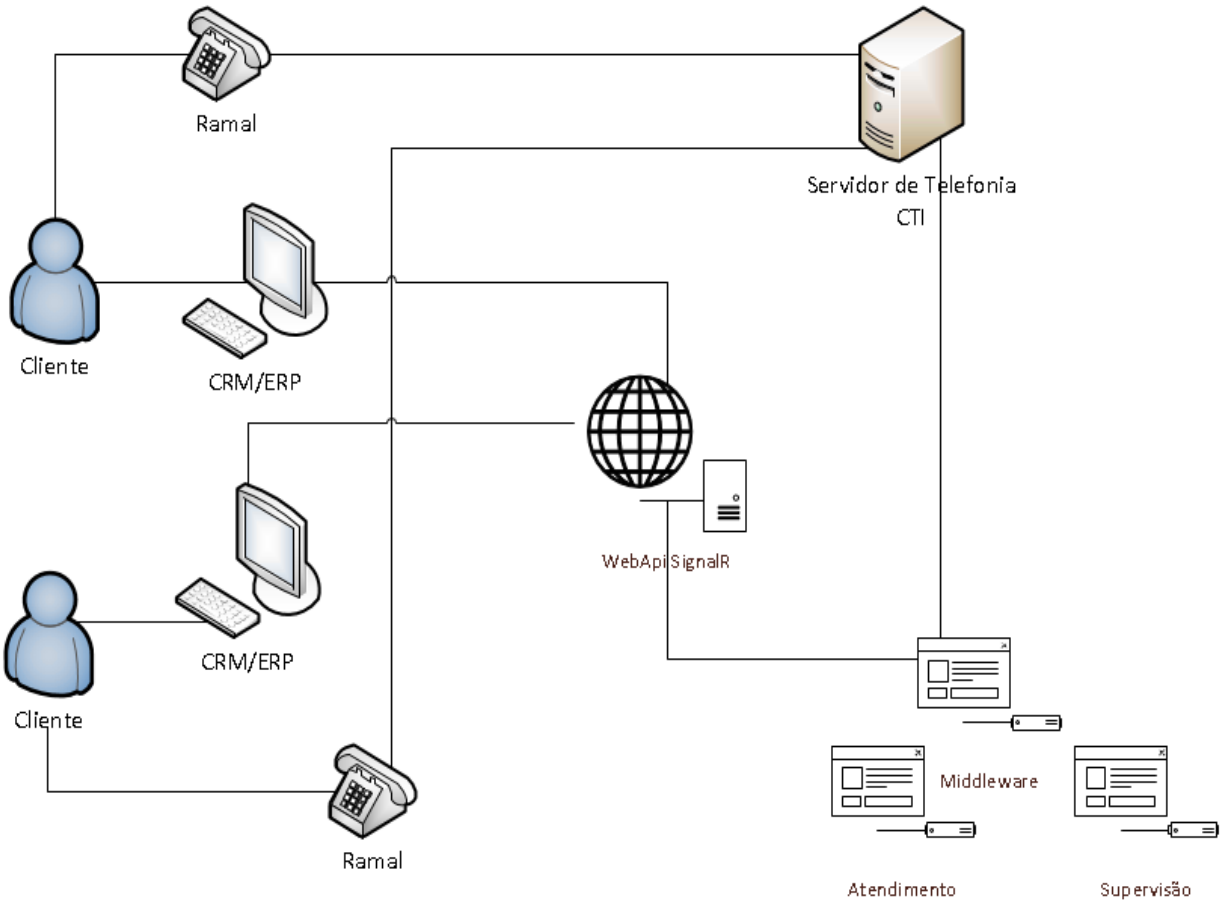
Características da Plataforma

CTI Manager: é um software que executa todas as funções de uma central telefônica convencional através das principais tecnologias de comunicação existentes no mercado, como por exemplo: linhas telefônicas analógicas, links de telefonia digital via placas de comunicação TDM (Time-Division Multplexing, ou Multiplexação por divisão de tempo), VoIP (Voice over IP, ou Voz sobre IP) através do protocolo SIP.

Middleware

O Middleware é um software que é executado em modo serviço. Essa ferramenta tem como funcionalidade principal comunicar diretamente com o CRM e traduzir as ações do CRM para o CTI Manager. Utilizando a integração mais detalhada ao longo da documentação é possível que o CRM implemente todas as funcionalidades de uma central.

A comunicação com o Middleware é feita por API disponibilizada por nós usando SignalR. A integração será feita toda através de JavaScript, onde o integrador terá acesso à uma grande variedade de métodos.



Preparando o ambiente

- **Instalar Ramal**

  Para começar o desenvolvimento é mandatório possuir o ramal.

  Será necessário utilizar um SoftPhone para registrar seu Ramal.

  Link do Softphone : http://integracao.epbx.com.br/Arquivos/eyeBeam.zip

  Configuração do Softphone:

  Display Name : 1000 (Número do Ramal fornecido por nós)

  User name : 1000 (Número do Ramal fornecido por nós)

  Password: *****(Senha fornecida por nós)

  Domain: homologacao.epbx.com.br

  Sip Listen Port Manual : 5060

- **Configurar seu Ambiente de Integração**

  Importar no projeto:

```
<script src="http:// integracao.epbx.com.br/service/scripts/jquery-3.1.0.min.js"></script>
<script src=" http:// integracao.epbx.com.br/service/scripts/jquery.signalR.min.js"></script>
<script src=" http:// integracao.epbx.com.br/service/signalr/hubs"></script>
<!-- Scripts do dominio da aplicação -->
<script src="app/app.js"></script>
```

  Com as bibliotecas referenciadas no seu projeto e o Softphone registrado, será possível iniciar as chamadas da integração.

Autenticação

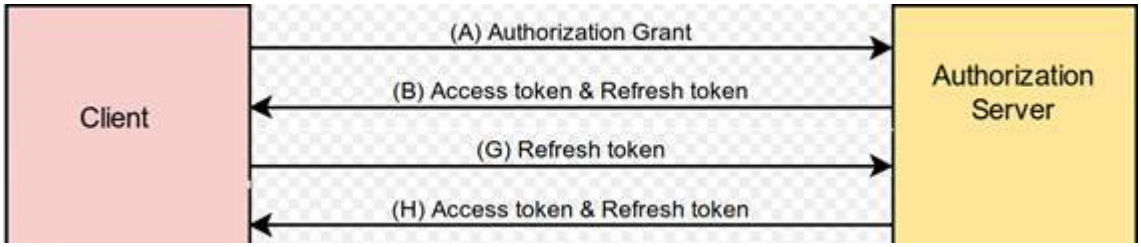
A autenticação do usuário é composta por duas etapas. Primeiro será enviado uma requisição no padrão OAuth para nosso servidor com as credencias do usuário, onde o mesmo responderá com um token valido por um período de tempo especifico. Através deste token será iniciada uma conexão websockt com a api SignalR. Através desta conexão será feita toda a comunicação Cliente / Servidor. Conforme ilustrado no exemplo abaixo:

```
adapter.login($("#usernameInput").val(), $("#passwordInput").val(), idPaOrIpOrRamal,tipoLogon)
 .then(adapter.conectarSignalr)
 .then(adapter.iniciarAtendimento)
 .fail(loginError)
 .fail(formLoginReady);
```

- **Api de Autenticação**

  A API de autenticação do EpbxManager utiliza o padrão OAuth para fazer a autenticação do usuário. Cada autenticação gera um token de acesso que é valido por 10 minutos (período configurável). Para manter a aplicação autenticada é necessário fazer o refresh token.

- **Fluxo de Autenticação da API**



- **Logon do ramal**

  Segue abaixo o exemplo de como fazer o logon do ramal.

```
$("#loginSubmit").click(function () {
        var idPaOrIpOrRamal, tipoLogon;
        formLoginLoading();

        if ($("#optionIsIP").is(":checked")) {
            idPaOrIpOrRamal = $("#ipInput").val();
            if (idPaOrIpOrRamal.length <= 4) {
                tipoLogon = adapter.TipoLogon.RamalVirtual;
            } else {
                tipoLogon = adapter.TipoLogon.Ip;
            }
        } else {
            idPaOrIpOrRamal = parseInt(parsePhone($("#idPaInput").val()));
            tipoLogon = adapter.TipoLogon.IdPa;
        }

adapter.login($("#usernameInput").val(), $("#passwordInput").val(), idPaOrIpOrRamal, tipoLogon)
          .then(adapter.conectarSignalr)
          .then(adapter.iniciarAtendimento)
          .fail(loginError)
          .fail(formLoginReady);
        return false;
});
```

Chamadas

- **Efetuando uma chamada**

Para efetuar uma chamada através da api SinalR, basta chamar o método “discar” disponível do objeto “adapter”, conforme ilustrado no exemplo abaixo.

```
// tipoDiscagem: 1 = Numero Externo, 2 = Ramal

    adapter.server.discar(numero, tipoDiscagem)
            .then(function () {
                // sucesso na chamada da api. Implementar suas regras.
            })
            .fail(function (err) {
                // Erro na execução do método. Efetuar tratamentos.
            });
```

- **Recebendo uma chamada**

Quando o ramal logado receber uma chamada, o servidor através do SignalR executa o método configurado para o evento “events.onChamada”. Se a chamada recebida for originada do discador automático, os atributos de chamada possuirá atributos adicionais. Conforme ilustrado no exemplo abaixo.

```
windowHandler.on(events.onChamada, function (event, ramal, chamada) {
    chamadaAtual = chamada;
    mostrarChamada(chamada, "Recebendo chamada");
    // recebimento de chamada normal ou de discador.
    // Quando discador, o objeto chamada contera os campos
    // codigoCliente
    // nomeCliente
    // codigoCampanha
    // telefoneCliente
    // infoAdicional1
    // infoAdicional2
    // infoAdicional3
    // infoAdicional4
    // infoAdicional5
});
```

- **Desligando uma chamada em curso**

Durante uma chamada em curso, o operador poderá desliga-la. Para implementar essa funcionalidade basta chamar o método “desligar()” disponível no objeto “adapter”, conforme ilustrado abaixo. Caso ocorra algum erro durante a execução, os detalhes estarão disponíveis no objeto de retorno na propriedade “message”

```
adapter.server.desligar().fail(console.log(err.message));
```

- **Efetuando uma consulta**

  Durante uma chamada em curso, o operador poderá realizar uma consulta “outra chamada” para outro ramal ou número externo. Ao realizar essa consulta, a chamada em curso entra em espera, onde o cliente ouvirá uma mensagem configurada na plataforma.

```
adapter.server.consultar(numero, tipoDiscagem)
        .then(function () {
          // chamada atual ficou em espera
          // tipoDiscagem: 1 = Numero Externo, 2 = Ramal

          mostrarChamadaEmEspera(emEspera);
        })
        .fail(function (err) {
          mostrarChamada(emEspera);
          resetTransferirPanels();
          commandError(err);
        });
```

- **Transferindo uma chamada em curso**

  Após efetuar uma consulta para outro ramal ou número externo, o operador poderá realizar a transferência da primeira chamada que estava em espera. Para implementar essa funcionalidade basta chamar o método “consultar(numero, tipo)” conforme exemplo ilustrado abaixo.

```
// transfere a chamada que foi colocada em espera pelo método de consulta para a chamada em curso
        adapter.server.transferir()
          .then(function () {
            resetTransferirPanels();
          })
        .fail(commandError);
```

- **Efetuando download de uma gravação**

  Quando uma chamada é encerrada, o servidor executa automaticamente o método configurado para o evento “events.onChamadaGlobalId”. Um dos parâmetros recebidos é o “GlobalId”. Esse parâmetro é utilizado para identificar a chamada. Com ele podemos gerar um link para realizar o download da gravação em mp3 conforme no exemplo abaixo:

```
windowHandler.on(events.onChamadaGlobalId, function (event, ramal, chamada, globalId) {
  var url = adapter.getUrlDownloadChamada(globalId);
  window.location.href = url;
});
```

Ex:
http://seu_servidor/Service/api/Ligacao/Download/GLOBALID?token_type=bearer&access_token=TOKEN_SESSAO

Intervalos

Todos os ramais são identificados na plataforma com um determinado intervalo. O intervalo possui como principal objetivo identificar se o ramal está trabalhando, ocioso, em pausa, banheiro, almoço, etc. E além destas finalidades, é possível determinar se determinado intervalo possui permissão para efetuar chamada e receber chamada.

Os intervalos podem ser customizados de acordo com a necessidade do cliente e com a nomenclatura que achar mais apropriada. É possível também, determinar quais grupos poderão usar determinados intervalos, ampliando desta forma a capacidade de configurações e utilização.

- **Recebendo Intervalos**

  Quando o ramal do operador efetua o logon na plataforma, o sistema dispara o evento "events.onInfoIntervaloRamal" enviando os intervalos que o ramal logado tem permissão, conforme exemplo abaixo:

```
windowHandler.on(events.onInfoIntervaloRamal, function(event, ramal, infoIntervalo) {
  intervaloOptions[infoIntervalo.RamalStatusDetalheId] = {
    Descricao: infoIntervalo.Descricao,
    Produtivo: infoIntervalo.Produtivo
  };
  if (intervaloAtual.RamalStatusDetalheId === infoIntervalo.RamalStatusDetalheId) {
    mostrarIntervaloStatus(infoIntervalo.RamalStatusDetalheId);
  }
  updateIntervalorOptions();
});
```

- **Alterando o intervalo de um ramal**

  Para alterar o intervalo atual do ramal, basta chamar o método "alterarIntervaloTipo", através do objeto "adapter".

```
Var intervalo = $("#intervaloSelect").val();
adapter.server.alterarIntervaloTipo(intervalo).fail(function(){
        console.log("Ocorreu um erro ao alterar o intervalo do ramal");
});
```

Conferência

O serviço de conferência funciona muito simples, primeiro você precisa adicionar os participantes na conferência para depois iniciar a conferência. A conferência aceita participante do tipo ramal ou tipo número externo.

- **Adicionar participante**

```
adapter.server.conferenciaAdicionar(numero), tipoDiscagem)
```

- **Iniciar conferência**

```
adapter.server.conferenciaIniciar().fail(function(){
      console.log("Ocorreu um erro ao iniciar a conferência")
 });
```

- **Eventos Callback**

```
windowHandler.on(events.onConferenciaTermino, function(event, ramal, chamada) {
   mostrarRamalLivre();
});

windowHandler.on(events.onConferenciaErro, function(event, ramal, ex) {
   console.error(ex);
   mostrarRamalLivre("Erro ao iniciar conferência");
});
```