Documentação - AbrilID

Documento referente ao **AbrilID Versão 3.3.4**
Última atualização: **06/08/2012**

Documentação - AbrilID

ATENÇÃO: a versão 3.x do Abril ID é totalmente retrocompatível com suas versões anteriores, ou seja, é possível instanciar o objeto AbrilId utilizando uma das 3 formas existentes:

Os exemplos abaixo são apenas ilustrativos, consulte a documentação para saber como usar o Abril ID

Versão 1.x

```
setApplication('ID_DO_SITE');
```

Versão 2.x

```
AbrilIdWidgetSetup.configuracao = {
 produto:'ID_DO_SITE'
};
AbrilIdWidgetSetup.initialize();
```

Versão 3.x

```
var topo = new AbrilId();
topo.initialize({
  produto: 'ID_DO_SITE'
  .
  .
  .
});
```

Contudo, apenas **UMA** dessas formas deve ser usada em uma mesma página e se for necessária a criação de mais de uma instância do AbrilID, deve-se usar obrigatoriamente a versão 3.x, umas vez que as versões 1.x e 2.x **NÃO** possuem suporte à múltiplas instâncias.

Compatibilidade

O Abril ID é compatível e está homologado para os seguinte web browsers:

- Internet Explorer 7.x, 8.x e 9.x;
- Mozilla Firefox 3.5, 3.6 e 5.x ou maior;
- Google Chrome 10.x ou maior;
- Safari 5.0 ou maior;

Caso queira garantir que o Internet Explorer 8.x do usuário abra utilizando o modo de exibição Internet Explorer 8, utilize a seguinte metatag:

```
<meta http-equiv="X-UA-Compatible" content="IE=8" />
```

Como usar

Pré-requisitos

- **NÃO** é necessário que o site cliente esteja dentro do domínio **abril.com.br**, uma vez que o Abril Id possui suporte cross-domain. Contudo, é necessário comunicar a equipe do Abril Id do uso cross-domain, para que seja feita a liberação para o domínio específico do site;
- Observar se o ambiente de desenvolvimento está impedido de acessar o ambiente de homologação do Abril ID (http://qa.id.abril.com.br). Verificar configurações de proxy ou firewall.

Instalação

Para deixar disponível as funcionalidades do Abril ID nas páginas do seu site, inclua a linha abaixo em todas as páginas onde o Abril Id for necessário:

```
<script src="http://id.abril.com.br/javascripts/widgets.js"></script>
```

ATENÇÃO: para realizar testes em nosso ambiente de homologação utilize a URL +http://qa.id.abril.com.br/javascripts/widgets.js

Configuração Mínima

Para que o Abril Id funcione propriamente em seu site, duas configurações são **obrigatórias**. Um exemplo seguido de explicação detalhada segue abaixo.

```
<script type="text/javascript">
  var configuracao = {
    produto:'ID_DO_SEU_SITE',
    container: 'ID_DA_DIV'
  };
</script>
```

Exemplo:

```
<script type="text/javascript">
  var configuracao = {
    produto:'SAUDE',
    container: 'abrilID'
  };
</script>
```

- produto: representa o ID do seu site no Abril Id. Para obter este ID, entre em contato com a equipe do Abril Id;
- container: é o ID da div em que o Abril Id será gerado. **ATENÇÃO: você deve criar essa div em seu site.**

Instanciando o Abril ID

Após criar a configuração, basta instanciar o Abril Id para começar a usá-lo. Isso é feita da maneira exemplificada abaixo.

```
<script type="text/javascript">
  var exemplo = new AbrilId();
  exemplo.initialize(configuracao);
</script>
```

IMPORTANTE: é possível instanciar um componente WIDGET, um componente INCLUDE e um componente TOPO numa mesma página. Contudo, não é possível instanciar dois componentes do mesmo tipo. Segue exemplo abaixo.*

```
<script type="text/javascript">
  var configuracao_widget = {
    produto:'site',
    container: 'div_widget',
    componente: Componente.WIDGET
  };

  var configuracao_include = {
    produto:'site',
    container:'div_include',
    componente: Componente.INCLUDE
  };

  var configuracao_topo = {
    produto:'site',
    container:'div_topo',
    componente: Componente.TOPO
  };

  var novowidget = new AbrilId();
  novowidget.initialize(configuracao_widget);

  var novoinclude = new AbrilId();
  novoinclude.initialize(configuracao_include);

  var novotopo = new AbrilId();
  novotopo.initialize(configuracao_topo);
</script>
```

ATENÇÃO: para que o Abril ID funcione propriamente no **Internet Explorer 7, 8 e 9**, é necessário que o código acima (configuração, criação da instância e inicialização) esteja na mesma página que a chamada para o widgets.js. **Não coloque esta parte do código num arquivo externo a ser carregado na página.**

Configurações Opcionais

Além das configurações mínimas, o cliente pode fornecer alguns parâmetros com funções específicas. Abaixo seguem alguns exemplos que incluem as configurações obrigatórias e as opcionais.

Provedores

```
<script type="text/javascript">
  var configuracao = {
    produto:'ID_DO_SEU_SITE',
    container: 'ID_DA_DIV',
    provedores: ['facebook', 'google']  //parâmetro para ativar/desativar login via
redes sociais
  };
</script>
```

Descrição: Este parâmetro permite a ativação/desativação do login via redes sociais (Facebook, Twitter ou Google Accounts). Na interface dos componentes Widget ou Include de login a diferença visual é a presença dos diferentes botões das redes sociais ou nenhum botão (quando desabilitados) para login alternativo no Abril ID, impossibilitando o usuário final de fazer uso deste recurso.

Padrão: caso não declare, por padrão serão exibidos os botões de todas as redes sociais disponíveis (Facebook, Twitter e Google Accounts).

Uso: dentro da configuração é necessário declarar um array contendo os provedores desejados, na ordem em que devem aparecer. Exemplo:

```
provedores: ['facebook'] //apenas facebook
provedores: ['facebook','google'] //facebook e google
provedores: ['google','twitter'] //google e twitter
provedores: [] //nenhum provedor
```

Componente

```
<script type="text/javascript">
var configuracao = {
    produto:'ID_DO_SEU_SITE',
    container: 'ID_DA_DIV',
    provedores: ['facebook', 'google'],
    componente: Componente.INCLUDE //parâmetro determinar tipo de componente
  };
</script>
```

Descrição: Este parâmetro define qual o tipo de componente que será carregado. Atualmente existem 3 valores possíveis: Componente.WIDGET, Componente.INCLUDE ou Componente.TOPO. Este parâmetro determina uma série de modificações na interface, de forma a alterar a cara dos dados de forma a ser um componente Widget ou um Include. **ATENÇÃO: alguns componentes possuem apenas certas interfaces**, como mostrado na tabela abaixo.

Interface	Componente.WIDGET	Componente.INCLUDE	Componente.TOPO
Login	possui	possui	possui
Logout	possui	não possui	possui
Recuperação de senha	possui	possui	possui
Criação de conta	possui	possui	possui
Validação de senha	possui	não possui	não possui
Edição completa de perfil	não possui	possui	não possui

Edição básica de perfil	possui	possui	não possui
Alteração de senha	possui	não possui	não possui
Coleta de dados	possui	não possui	não possui
Newsletter	não possui	possui	não possui

*Padrão: caso não declare, por padrão o componente será Componente.WIDGET.

*Uso: dentro da configuração é possível indicar qual o tipo da interface que deve ser carregada.

```
componente: Componente.WIDGET //a interface aparecerá como um Widget
componente: Componente.INCLUDE //a interface aparecerá como um Include
componente: Componente.TOPO //a interface aparecerá como um Topo
```

Seta

ATENÇÃO: essa configuração existe apenas no componente Widget.

```
<script type="text/javascript">
  var configuracao = {
    produto:'ID_DO_SEU_SITE',
    container: 'ID_DA_DIV',
    provedores: ['facebook', 'google'],
    componente: Componente.WIDGET,
    seta: 'SD' //parâmetro para posicionar uma seta indicativa no topo do componente
Widget
  };
</script>
```

Descrição: Este parâmetro permite o posicionamento de uma seta indicativa na parte superior de um componente Widget. Os valores possíveis são indicativos da posição da seta: 'SE' (superior direita), 'SC' (superior centro), 'SD' (superior direita).

Padrão: caso não declare, por padrão o componente Widget não exibirá nenhuma seta indicativa.

Uso: os valores possíveis são indicativos da posição da seta: 'SE' (superior direita), 'SC' (superior centro), 'SD' (superior direita)

```
seta: 'SD' //seta no canto superior direito
seta: 'SC' //seta no centro, na parte superior
seta: 'SE' //seta no canto superior esquerdo
```

Observação: a seta não pode ser utilizada no componente TOPO

Estilo

```
<script type="text/javascript">
  var configuracao = {
    produto:'ID_DO_SEU_SITE',
    container: 'ID_DA_DIV',
    provedores: ['facebook', 'google'],
    componente: Componente.WIDGET,
    seta: 'SD',
    estilo:{ //início da configuração de estilo
      'general': {
      'background-color':'#4f6068', //cor do background da caixa do widget//
      'border-color':'#111a33', //cor da borda da caixa do widget//
      'border-width':'1', //largura da borda da caixa do widget, sem o 'px' -
minimo: 1, maximo: 5//
      'text-color':'#DDCCBB', //cor do 'X' para fechar o widget e de todos os textos
do widget, exceto erros e botoes//
      'inner-border-color':'#010729' //cor da borda da caixa interna, do separador e
da borda dos campos//
      },
      'inner-box':{
        'background-color':'#66777f' //cor do background da caixa interna do
widget//
      },
      'button':{
        'stroke':{
          'text-color':'#ffffff' , //cor de fonte do botao com gradiente//
          'background-color':{
            'start':'#020926', //cor inicial do gradiente do botao//
            'end':'#020926' //cor final do gradiente do botao//
          },
          'border-color':'#020926' //cor de borda do botao com gradiente//
        },
        'flat':{
          'text-color':'#ffffff' , //cor de fonte do botao sem gradiente//
          'background-color':'#020926', //cor de fundo do botao sem gradiente//
          'border-color':'#020926' //cor de borda do botao sem gradiente//
        }
      },
      'error':{
        'background-color':'#66777f', //cor fundo da caixa de erro//
        'border-color':'#020926', //cor de borda da caixa de erro//
        'text-color':'#ffffff' //cor do texto da caixa de erro e da borda dos campos
indicados no erro//
      },
      'header':{
        'type': 'light' //'default': logotipo verde, assinatura cinza escuro |
'light': logotipo e assinatura brancos//
      },
      'topo': {
        'background-color': {
          'start': '#CCCCCC', //cor inicial do gradiente do fundo//
          'end': '#CCCCCC' //cor final do gradiente do fundo//
        },
        'line-color':'#888888', //cor do separador //
        'text-color':'#888888', //cor dos textos do topo //
        'border-color':'#888888', // cor da borda do topo, caso haja //
        'border-width':'2', // espessura da borda - de 2 a 5 - default 0//
```

```
        'logo':'default' // 'default' (verde)|'white' (branco)|'light grey' (cinza
claro)|'grey' (cinza)|'black' (preto)  //
      }
```

```
    }
  };
</script>
```

Descrição: este parâmetro permite a configuração de layout de praticamente todas as estruturas dos componentes Widget e Include.

Padrão: o usuário pode declarar de nenhuma a todas as configurações de layout, e qualquer atributo que não for declarado receberá seu layout padrão.

Uso: é necessário preencher os atributos com os valores cabíveis (por exemplo, uma cor hexadecimal)

Origem de evento

```
<script type="text/javascript">
  var configuracao = {
    produto:'ID_DO_SEU_SITE',
    container: 'ID_DA_DIV',
    origem_evento: 'Promoção Ganhe um iPad'
  };
</script>
```

Descrição: Este parâmetro permite identificar o ponto ou propósito da interação do usuário com o Abril ID.

Padrão: caso não declare, por padrão será recuperada a URL da pagina onde o widget estiver instalado.

Uso: o valor é um texto.

```
<script type="text/javascript">
 ...
 origem_evento: 'Um texto que descreva a origem'
 //origem_evento: 'Promoção Você em NY'
 //origem_evento: 'Promoção Tititi Premios Todo Dia'
 //origem_evento: 'Casa Curso de Decoração ed 2012'
 ...
</script>
```

Exibindo o Abril ID

Após instalar e configurar o Abril ID, você pode exibir a interface escolhida em seu site atravé de apenas um comando, que pode estar relacionado a uma ação do usuário (um click, por exemplo) ou ao carregamento da página. Segue uma tabela com o comando que chama cada um uma das interfaces.

Interface	Chamada
Login	[INSTANCIA DO SEU COMPONENTE].login.render();
Logout	[INSTANCIA DO SEU COMPONENTE].logout.render();

Recuperação de senha	[INSTANCIA DO SEU COMPONENTE].password_reset.render();
Criação de conta	[INSTANCIA DO SEU COMPONENTE].signup.render();
Validação de senha	[INSTANCIA DO SEU COMPONENTE].password_validation.render();
Edição completa de perfil	[INSTANCIA DO SEU COMPONENTE].profile_full_update.render();
Edição básica de perfil	[INSTANCIA DO SEU COMPONENTE].profile_update.render();
Alteração de senha	[INSTANCIA DO SEU COMPONENTE].password_update.render();
Coleta de dados	[INSTANCIA DO SEU COMPONENTE].profile_attribute.render();
Newsletter	[INSTANCIA DO SEU COMPONENTE].newsletter.render();

```
<script type="text/javascript">
  var configuracao = {
    produto:'site',
    container: 'div_widget',
    componente: Componente.WIDGET
  };

  var widget = new AbrilId();
  widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);" onclick="widget.login.render();">login</a>
```

Exibindo o Topo Abril ID

Para exibir o TOPO Abril ID é necessário ter o widget.js instalado no site. A instalação, regras de compatibilidade, pré-requisitos e configuração mínima aplicadas a exibição do TOPO seguem as mesmas aplicadas a exibição dos widgets/includes do Abril ID, detalhadas no começo da documentação.

Para exibição do TOPO, deve ser criada uma "div" no local da página onde o TOPO deverá ser exibido, definir as configurações necessárias e instanciar o TOPO. Em seguida, executar o método *initialize* passando as configurações como parâmetro e depois executar a função *topo_abrilid.render* para renderizar o topo na página.

Nos parâmetros de configuração o **componente** deve ser definido como "Componente.TOPO", o **container** deve ser o id da div criada no passo anterior e o **topo** pode ser definido como mini,medio ou pequeno, dependendo do modelo do topo que se deseja renderizar. Se nenhuma opção de **topo** for definida, o topo mini será renderizado como padrão.

Obs.: Se o site estiver definindo o **estilo** na configuração de renderização do widget ou do include, essas mesmas configurações devem ser passadas como parâmetro na chamada do TOPO.

Exemplo de exibição do TOPO pequeno:

```
<script type="text/javascript">
  // instânciar o TOPO
  var topo = new AbrilId();

  // inicializar o TOPO com as configurações desejadas
  topo.initialize(
    produto: 'site_login',
    container: 'div_topo',
    provedores: ['facebook', 'twitter', 'google'],
    componente: Componente.TOPO,
    topo: 'pequeno',
    estilo: { // Opcional. Se não for definido um estilo, será utilizado o estilo
padrão.
      'topo': {
        'background-color': {
   'start': '#CCCCCC', //cor inicial do gradiente do fundo//
   'end': '#CCCCCC' //cor final do gradiente do fundo//
        },
        'line-color':'#888888', //cor do separador//
        'text-color':'#888888', //cor do separador//
        'border-color':'#888888',
        'border-width':'2',
        'logo':'default'
      }
    }
 );
</script>

...

// Renderiza o TOPO na página
<script>
  topo.topo_abrilid.render();
</script>
```

Interfaces

As interfaces são os elementos visíveis dos componentes do Abril ID. É o "widget de login", ou o "include de newsletter", ou seja, é a parte que aparece e é usada pelo usuário final do site cliente. Essas interfaces interagem com site através de "callbacks", que são dados estruturados (JSON) enviados ao site após algum processo ocorrer nos componentes. Alguns callbacks são genéricos e ocorrem (ou seja, são disparados) para todos os componentes. Outros são específicos e únicos para cada componente. Esses callbacks podem ser usados para pegar dados de usuários logados, ou trazer dados do perfil do usuário, além de prover feedbacks de sucesso ou falha das ações executadas nas interfaces.

Listeners

Sempre que um callback é disparado por um componente, o site cliente precisa estar preparado para "ouvir" o callback esperado. Para isso, o Abril ID disponibiliza uma forma de registrar listener para aguardarem o disparo de callbacks específicos. Por exemplo, para ouvir sucesso de um login, deve registrar-se o seguinte listener:

```
<script>
widgetEventManager.listenTo('login:success', function(json){});
</script>
```

O primeiro parâmetro do listener diz respeito ao nome do callback esperado (veja abaixo) e o segundo parâmetro é uma função que recebe os dados estruturados (JSON) e executa quando ouvir o disparo do callback. De maneira análoga, para ouvir um callback de falha de login, utilize:

```
<script>
widgetEventManager.listenTo('login:failure', function(json){});
</script>
```

Callbacks genéricos

A partir da versão 2.0, dois novos callbacks foram implementados para todos os componentes. O primeiro callback é "widgetLoaded:success" e é acionado cada vez que um componente é renderizado. Isso vale tanto para a abertura do widget de login, por exemplo, como pro carregamento do widget com a resposta após o fluxo de login.

O segundo callback geral é o "widgetClosed:success" que é acionado cada vez que um componente é fechado através do botão de fechar (o "X" posicionado na parte superior direita dos widgets). Para ouvir estes callbacks, registre os listeners dessa maneira:

```
<script>
widgetEventManager.listenTo('widgetLoaded:success', function(json){});
widgetEventManager.listenTo('widgetClosed:success', function(json){});
</script>
```

Interface de Login

Componentes implementados

- Componente.WIDGET

Observação: a interface de Login também aceita componente INCLUDE. Contudo, o login faz parte do fluxo de outras interfaces dos componentes INCLUDE, como newsletter e edição de perfil, não sendo necessário a chamada direta desta interface.

Campo de login

O Abril ID permite ao usuário se autênticar através do e-mail ou nome de usuário (username).

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);" onclick="widget.login.render();">login</a>
```

Opções

confirmation_url

Essa opção permite ao site cliente direcionar o usuário para a página que desejar, após o processo de confirmação de cadastro.

```
<a href="javascript: void(0);"
onclick="widget.signup.render({'confirmation_url':'http://www.abril.com.br'});">cria
r conta</a>
```

image_url

Essa opção permite ao site cliente desabilite as criação de conta e no local insira uma imagem, caso uma promoção não possa receber cadastros após uma data estimada.

```
<a href="javascript: void(0);"
onclick="widget.signup.render({'image_url':'<URL_IMAGEM>'});">criar conta</a>
```

Callbacks

Nome do callback	Formato do JSON

login:success	{
	"status": 0,
	"token":
	"96f82893416953cacc3ef9cac53b8646f
	656acfc4c4bc9810ddbd6a9de7ed140",
	"person": {
	"codigo": "ID_USUARIO",
	"name": "NOME",
	"email": "EMAIL",
	"avatar_url": "URL_DO_AVATAR",
	"perfil": {
	"sexo": "SEXO",
	"data_nascimento":
	"dd/mm/aaaa"
	}
	},
	"success": true,
	"message": "MENSAGEM"
	}

login:failure	{
	"status": CODIGO_DE_ERRO,
	"person": null,
	"success": false,
	"message": "MESSAGE"
	}

logged:true	{
	"token":
	"96f82893416953cacc3ef9cac53b8646f
	656acfc4c4bc9810ddbd6a9de7ed140",
	"person": {
	"codigo": "ID_USUARIO",
	"name": "NOME",
	"email": "EMAIL",
	"avatar_url": "URL_DO_AVATAR",
	"perfil": {
	"sexo": "SEXO",
	"data_nascimento":
	"dd/mm/aaaa"
	}
	},
	"success": true,
	"message": "MENSAGEM"
	}

logged:false	<pre>{
	"token": "",
	"person": null,
	"success": false,
	"message": "MESSAGE"
	}</pre>

Exemplo de uso do callback

```
<script>
widgetEventManager.listenTo('login:success', function(json) {
msg = json.person.name + "\n";
msg += json.person.email + "\n";
msg += json.person.codigo + "\n";
msg += json.person.avatar_url + "\n";
msg += json.message + "\n";
msg += json.person.perfil.sexo + "\n";
msg += json.person.perfil.data_nascimento + "\n";
msg += json.success;
alert(msg);
});
</script>
```

Cookies

No ato do login o Abril ID gera diversos cookies. Abaixo segue uma listagem dos dados disponíveis para o site cliente através dos Cookies:

```
Host: .abril.com.br
Name: aapgPersonCookie
Path: /
Content: {"personEmail":[E-MAIL DO USUARIO],"personId":[ID DO
USUARIO],"personName":[NOME DO USUARIO],"userName":[NOME DE USUARIO]}
Expires: At end of session

Host: .abril.com.br
Name: aapgTokenCookie
Path: /
Content: {"token":[TOKEN DE SESSÃO]}
Expires: At end of session
```

Os Cookies abaixo, embora ainda gerados, são oriundos do Passaporte Abril e portanto considerados depreciados, e serão descontinuados em versão futura ainda não definida do Abril ID. **É altamente desaconselhado utilizar estes cookies e suas informações para qualquer fim**.

```
Host: .abril.com.br
Name: usuario
Path: /
Content: [CODIGO];[NOME];[E-MAIL];[SEXO (M/F)];[DATA DE NASCIMENTO (dd/mm/yyyy)]
Expires: At end of session

Host: .abril.com.br
Name: Ticket
Path: /
Content: [CODIGO]
Expires: At end of session

Host: .abril.com.br
Name: pst_login
Path: /
Content: S
Expires: At end of session
```

Interface de Logout

Componentes implementados

- Componente.WIDGET

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);" onclick="widget.logout.render();">logout</a>
```

Callbacks

Nome do callback	Formato do JSON

logout:success	<pre>{
	  "success": true,
	  "message": "MENSAGEM"
	}</pre>

logout:failure	<pre>{
	  "success": false,
	  "message": "MENSAGEM"
	}</pre>

Peculiaridades

Esta interface comporta-se de maneira diferente dependendo do site que requisita o logout. Caso a aplicação seja do tipo que aceita reconhecimento automático de login, a interface, ao ser exibida, já terá realizado o logout para todos os sites, caso contrário, exibirá a mensagem de que já realizou o logout do site e a opção para executar o logout para todos os sites da Abril.

Interface de Recuperação de Senha

Componentes implementados

- Componente.WIDGET

Observação: a interface de Recuperação de Senha também aceita componente INCLUDE. Contudo, essa interface é parte intrínseca do fluxo de login, não sendo necessária a chamada direta desta interface.

Campo de login ou nome de usuário

O Abril ID permite ao usuário recuperar a senha através do e-mail ou nome de usuário (username).

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);" onclick="widget.password_reset.render();">recuperar
senha</a>
```

Callbacks

Nome do callback	Formato do JSON
passwordReset:success	`{ "code": "CODE", "success": true, "message": 'MENSAGEM", "login": 'EMAIL' }`
passwordReset:failure	`{ "code": "CODE", "success": false, "message": "MESSAGE" }`

Interface de Criação de Conta

Componentes implementados

- Componente.WIDGET

Observação: a interface de Criação de Conta também aceita componente INCLUDE. Contudo, essa interface é parte intrínseca do fluxo de login, não sendo necessária a chamada direta desta interface.

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);" onclick="widget.signup.render();">criar conta</a>
```

Opções

confirmation_url

Essa opção permite ao site cliente direcionar o usuário para a página que desejar, após o processo de confirmação de cadastro.

```
<a href="javascript: void(0);"
onclick="widget.signup.render({'confirmation_url':'http://www.abril.com.br'});">cria
r conta</a>
```

Callbacks

Nome do callback	Formato do JSON

accountCreation:success	<pre>{
	  "success": true,
	  "code": CODE,
	  "message": "MESSAGE",
	  "application":
	"LOGIN_DA_APLICACAO",
	  "login": "EMAIL"
	}</pre>

accountCreation:failure	<pre>{
	  "success": false,
	  "code": CODE,
	  "message": "MESSAGE"
	}</pre>

Interface de Validação de Senha

Componentes implementados

- Componente.WIDGET

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);"
onclick="widget.password_validation.render();">validação de senha</a>
```

Callbacks

Nome do callback	Formato do JSON

passwordValidation:success	<pre>{
	  "success": true,
	  "message": "MESSAGE"
	}</pre>

passwordValidation:failure	<pre>{
	  "success": false,
	  "message": "MESSAGE"
	}</pre>

Interface de Edição Completa de Perfil

Componentes implementados

- Componente.INCLUDE

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.INCLUDE
};

var include = new AbrilId();
include.initialize(configuracao);
</script>

...

<a href="javascript: void(0);"
onclick="include.profile_full_update.render();">edição completa de perfil</a>
```

Callbacks

Observação: os dados de retorno presentes em "attribute_values" são apenas exemplos. Os dados que retornam são os campos atribuídos ao site cliente. Nos exemplos abaixo é mais importante observar a estrutura dos dados em "attribute_values" e atentar para o campo "public" que indica a permissão dada pelo usuário para a exibição daquele atributo em especial.

Nome do callback	Formato do JSON

profileUpdate:success	{
	"message": "Alteração realizada com sucesso!",
	"profile": {
	"avatar_url": "URL_DO_AVATAR",
	"name": "Perfil",
	"attribute_values": [
	{
	"public": "0",
	"value": "M",
	"key": "sexo"
	},
	{
	"public": "1",
	"value": "12345678900",
	"key": "cpf"
	}
	],
	"codigo": ID_DO_PERFIL,
	"application": ID_DA_APLICACAO,
	"person_id": ID_DO_USUARIO,
	"description": ""
	},
	"success": true
	}

profileUpdate:failure	{
	  "message": "Ocorreu um erro no sistema de autenticação.",
	  "profile": {
	  "avatar_url": "URL_DO_AVATAR",
	  "name": "Perfil",
	  "attribute_values": [
	  {
	  "public": "0",
	  "value": "Mãe",
	  "key": "nomemae"
	  },
	  {
	  "public": "0",
	  "value": "12345678900",
	  "key": "cpf"
	  }
	  ],
	  "codigo": ID_DO_PERFIL,
	  "application": ID_DA_APLICACAO,
	  "person_id": ID_DO_USUARIO,
	  "description": ""
	  },
	  "success": false
	}

Interface de Edição Básica de Perfil

Componentes implementados

- Componente.WIDGET
- Componente.INCLUDE

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);" onclick="widget.profile_update.render();">edição básica de perfil</a>
```

Callbacks

Nome do callback	Formato do JSON

profileUpdate:success	<pre>{
	  "success": true,
	  "message": "MENSAGEM',
	  "avatar_url": "AVATAR_URL"
	}</pre>

profileUpdate:failure	<pre>{
	  "success": false,
	  "message": "MENSAGEM'
	}</pre>

Interface de Alteração de Senha

Componentes implementados

- Componente.WIDGET

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);" onclick="widget.password_update.render();">alteração
de senha</a>
```

Callbacks

Nome do callback	Formato do JSON

passwordUpdate:success	<pre>{
	  "success": true,
	  "message": "",
	  "person": {
	  "name": "NOME",
	  "codigo": "000",
	  "email": "EMAIL"
	  }
	}</pre>

passwordUpdate:failure	<pre>{
	  "success": false,
	  "message": "MENSAGEM'
	}</pre>

Interface de Coleta de Dados

Componentes implementados

- Componente.WIDGET

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.WIDGET
};

var widget = new AbrilId();
widget.initialize(configuracao);
</script>

...

<a href="javascript: void(0);"
onclick="widget.profile_attribute.render({'attributes':['ATRIBUTO1', 'ATRIBUTO2',
'ATRIBUTO3']});">coleta de dados</a>
```

O Widget exibirá os atributos na ordem em que forem solicitados.
* Os atributos solicitados devem estar previamente cadastrados no AAPG e associados a aplicação (nessa associação é feita a configuração de obrigatoriedade dos campos).

Callbacks

Nome do callback	Formato do JSON

profileAttribute:success	<pre>{
	  "success": true,
	  "message": "",
	  "data": {
	  atributo1: true/false, //isto é, preenchido ou não preenchido
	  atributo2: true/false,
	  atributo3: true/false
	  }
	}</pre>

profileAttribute:failure	<pre>{
	  "success": false,
	  "message": "MENSAGEM'
	}</pre>

Interface de Newsletter

Componentes implementados

- Componente.INCLUDE

Implementação

```
<script type="text/javascript">
var configuracao = {
produto:'site',
container: 'div_widget',
componente: Componente.INCLUDE
};

var include = new AbrilId();
include.initialize(configuracao);
</script>

...

<a href="javascript: void(0);"
onclick="include.newsletter.render({'attributes':['ATRIBUTO1', 'ATRIBUTO2',
'ATRIBUTO3']});">login</a>
```