MANUAL PARA DESENVOLVEDORES
MODULOS DE PROVISIONAMENTO / SERVIDORES

Nota: Este material foi preparado pelos desenvolvedores do HOSTMGR com o intuito de instruir outros programadores em como proceder para o desenvolvimento de módulos de servidores compatíveis com o HOSTMGR.

Neste material você encontra todas as informações necessárias ao desenvolvimento de módulos de provisionamento/integração com servidores compatíveis com o gerenciador HOSTMGR.

Compreender como o modulo é organizado dentro do sistema é o primeiro passo para desenvolver. Para tanto, tenha em mente que cada modulo deve ser representado por um diretório com nome exclusivo, escrito em minúsculo e armazenado no diretório **/modulos/servidores.**

Dentro do diretório são armazenados os arquivos do modulo, sendo os mais importantes moduloExemplo.php (nome do modulo) e funcoes.php (nome padrão).

*index.php é apenas uma proteção para o diretório

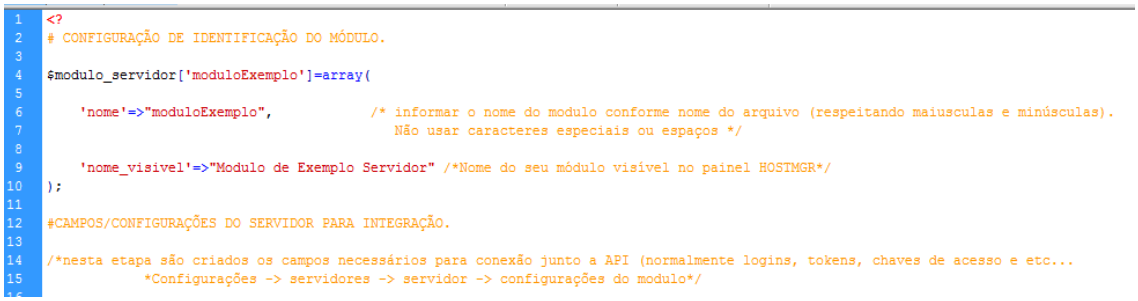
Junto desse material você recebeu os arquivos de exemplo para o desenvolvimento do primeiro modulo, que desmembramos abaixo.

Arquivo /modulos/servidores/moduloExemplo/moduloExemplo.php

Esse é o arquivo inicial do modulo, deve ser armazenado dentro do diretório do modulo que está sendo criado. Seu nome deve ser o mesmo nome do diretório acrescido de .php

Aqui se inicia a configuração do modulo.

Variável $modulo_servidor['moduloExemplo']

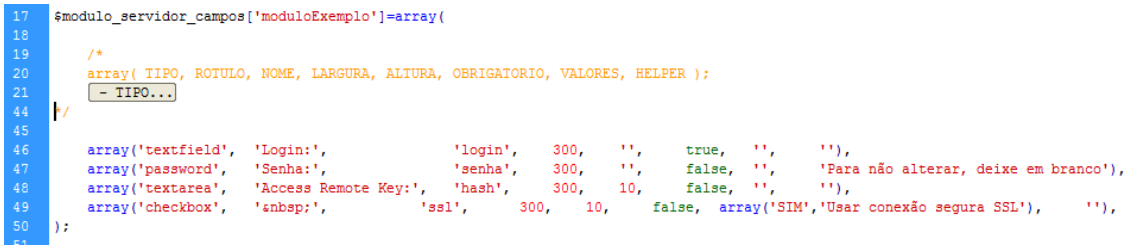


```php
<?
# CONFIGURAÇÃO DE IDENTIFICAÇÃO DO MÓDULO.

$modulo_servidor['moduloExemplo']=array(

    'nome'=>"moduloExemplo",           /* informar o nome do modulo conforme nome do arquivo (respeitando maiusculas e minúsculas).
                                          Não usar caracteres especiais ou espaços */

    'nome_visivel'=>"Modulo de Exemplo Servidor" /*Nome do seu módulo visível no painel HOSTMGR*/
);

#CAMPOS/CONFIGURAÇÕES DO SERVIDOR PARA INTEGRAÇÃO.

/*nesta etapa são criados os campos necessários para conexão junto a API (normalmente logins, tokens, chaves de acesso e etc...
          *Configurações -> servidores -> servidor -> configurações do modulo*/
```

É uma variável do tipo array que recebe informações básicas sobre o modulo. A chave desse array e o valor da chave "**nome**" devem ser o nome do seu modulo, ou seja, o nome dado ao diretório do modulo e ao arquivo editado, nesse caso, **"moduloExemplo"**.

Quanto a chave "**nome_visivel**", é o nome do modulo que será exibido ao usuário do sistema quando da utilização do modulo.

Variável $modulo_servidor_campos['moduloExemplo']



```php
$modulo_servidor_campos['moduloExemplo']=array(

    /*
    array( TIPO, ROTULO, NOME, LARGURA, ALTURA, OBRIGATORIO, VALORES, HELPER );
     - TIPO...
*/

    array('textfield',  'Login:',              'login',    300,    '',     true,   '',     ''),
    array('password',   'Senha:',              'senha',    300,    '',     false,  '',     'Para não alterar, deixe em branco'),
    array('textarea',   'Access Remote Key:',  'hash',     300,    10,     false,  '',     ''),
    array('checkbox',   ' ',          'ssl',      300,    10,     false,  array('SIM','Usar conexão segura SSL'),     ''),
);
```

Contem informações sobre os campos de configuração do modulo. Neste array são gerados os campos que o administrador deverá preencher ao criar um servidor com esse modulo de integração.

Os campos comumente cadastrados são: Email, Login, Senha, Chave de Acesso, Token, dentre outros dados configuráveis que você possa precisar para se conectar ao servidor e API que serão integrados. Eles ficam

localizados em configurações -> servidores -> servidor -> configurações do modulo.

Os campos são gerados na ordem cadastrada seguindo a regra de um array/campo por linha como no arquivo de exemplo.

Variável $modulo_servidor_produto['moduloExemplo']

```
/*Nesta etapa você deve definir os campos de configuração para os produtos, como espaço em disco, plano, usuarios, bitrate e etc...
        *Configurações -> produtos e serviços -> produto -> configurações do modulo*/

$modulo_servidor_produto['moduloExemplo']=array(
    array('textfield',  'Plano no WHM (package):',  'plano',    300,    '',     true,   '',     ''),
    array('textfield',  'Feature List (ACL):',  'featurelist',  300,    '',     false,  '',     'Se o produto for uma revenda. <strong>
Deixe em branco para "Default"</strong>'),

);
```

Contem informações sobre os campos de configuração do modulo. Neste array são gerados os campos de configuração dos produtos que utilizarão o modulo de integração. Os campos serão preenchidos ao configurar o produto.

Os campos comumente cadastrados são: espaço, plano, ouvintes, dentre outros limites que possa ser configurado para cada conta/plano específico.

Os campos ficam localizados em configurações -> produtos e serviços -> produto -> configurações do modulo.

ARQUIVO FUNCOES.PHP

Esse arquivo armazena as funções e principais comandos do modulo, seu nome é padrão e deve ser sempre armazenado dentro do diretório do modulo.

Funções auxiliares()

```php
function moduloExemplo_funcao_auxiliar($parametros){

    //seu código aqui

    $sucesso=true;

    if($sucesso){
        $retorno='xxxxxxxx';
    }else{
        $retorno='erro';
    }

    return $retorno;

}
```

As funções auxiliares são criadas por você para manuseio interno do seu modulo, sendo, portanto, funções que serão criadas de acordo com sua necessidade, e utilizadas dentro dos modulos de comando.

Seu uso não é obrigatório.

Seu nome deve ser sempre precedido pelo nome do modulo acrescido de _,
No nosso exemplo, moduloExemplo_ seguido pelo nome da função.

Nota: nunca crie funções extras dentro do modulo sem adicionar o prefixo com seu nome a fim de se evitar conflitos com outras funções do sistema ou de outros modulos.

Função moduloExemplo_acao_criar($parametros)

```php
function moduloExemplo_acao_criar($parametros){

    echo'<pre>';
    print_r($parametros);
    echo'</pre>';

    //exit();

    //Seu código para criar contas aqui

    $sucesso=true;

    if($sucesso){
        $resultado=array('status'=>'sucesso');
    }else{
        $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
    }

    return $resultado;
}
```

Essa função tem por finalidade ativar novas contas por meio da integração externa.

A função recebe todos os dados necessários para você conectar-se junto a sua API de integração, ativando a conta.

Essa função possui um nome padrão e também deve ser precedida por NOMEDOMODULO_, para seu perfeito funcionamento.

A função é executada toda vez que o operador, na tela da conta/produto, clica no botão criar, ou quando o pagamento é confirmado (caso o produto esteja configurado para tanto).

As principais informações estão armazenadas na variável **$parametros** onde estão organizados os dados distribuídos em arrays.

> **$parametros['conta']** – Array que contem os dados da conta que está sofrendo a intervenção, como id_conta, domínio, login, senha, data de cadastro, entre outras. Confira o anexo com a relação de variáveis.

$parametros['cliente'] – Array que contem os dados do cliente como (nome, empresa, telefone, email, endereço, numero, bairro, cidade, estado, cpf_cnpj)

$parametros['produto'] – Array que contem os dados referentes ao produto e ao seu modulo. São os valores dos produtos e dos campos que você criou. No nosso exemplo, **plano** e **featurelist.**

$parametros['servidor'] – Array que contem os dados do servidor em que o produto está sendo criado. Dentre os valores estão os campos criados pela variável **$modulo_servidor_campos**. No nosso exemplo, **login**, **senha**, **hash** e **ssl.**

Caso haja alguma dúvida quanto ao correto nome do campo a ser utilizado você pode listar todos os campos por meio da função print_r conforme o exemplo a seguir:

```
<?php echo'<pre>';
print_r($parametros['cliente']); //lista todos os valores do array cliente
echo'</pre>';
exit(); ?>
```

Retorno da função

IMPORTANTE: quando a função é executada com sucesso o status do produto é alterado de **pendente** para **ativo**. Para que isso ocorra, é necessário que você forneça retorno adequado da função em caso de sucesso.

```
45      $sucesso=true;
46
47      if($sucesso){
48          $resultado=array('status'=>'sucesso');
49      }else{
50          $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
51      }
52
53      return $resultado;
```

O retorno é sempre um array, sendo array('status'=>'sucesso'); em caso de sucesso, ou array('status'=>'erro','msg'=>'Mensagem de erro'); caso haja algum erro, sendo “Mensagem de erro”, qualquer informação que identifique o erro ocorrido.

Esse procedimento deve se repetir para todas as funções de comando do modulo.

Função moduloExemplo_acao_suspender($parametros)

```php
function moduloExemplo_acao_suspender($parametros){

    //Seu código para suspender contas aqui

    $sucesso=true;

    if($sucesso){
        $resultado=array('status'=>'sucesso');
    }else{
        $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
    }

    return $resultado;

}
```

Essa função tem por finalidade suspender/bloquear contas no servidor externo por meio da integração.

A função recebe todos os dados necessários para você conectar-se junto a sua API de integração e suspender a conta do cliente.

Essa função possui um nome padrão e também deve ser precedida por NOMEDOMODULO_, para seu perfeito funcionamento.

A função é executada toda vez que o operador, na tela da conta/produto, clica no botão Suspender, ou quando da execução do cron, quando identificado que o cliente está inadimplente.

Os dados são recebidos pela variável $parametros, que trás todas as informações necessárias por meio de array.

Todas as informações estão disponíveis em chaves do array $parametros (conta, cliente, produto e servidor), tudo conforme demonstrado no item *Funcao moduloExemplo_acao_criar*, que você pode conferir acima.

Retorno da função

IMPORTANTE: quando a função é executada com sucesso o status do produto é alterado de **ativo** para **suspenso**. Para que isso ocorra, é necessário que você forneça o retorno adequado da função em caso de sucesso.

```
45      $sucesso=true;
46
47      if($sucesso){
48          $resultado=array('status'=>'sucesso');
49      }else{
50          $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
51      }
52
53      return $resultado;
```

O retorno é sempre um array, sendo array('status'=>'sucesso'); em caso de sucesso, ou array('status'=>'erro','msg'=>'Mensagem de erro'); caso haja algum erro, sendo “Mensagem de erro”, qualquer informação que identifique o erro ocorrido.

Função moduloExemplo_acao_reativar($parametros)

```
57  function moduloExemplo_acao_reativar($parametros){
58
59      //Seu código para reativar contas aqui
60
61      $sucesso=true;
62
63      if($sucesso){
64          $resultado=array('status'=>'sucesso');
65      }else{
66          $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
67      }
68
69      return $resultado;
70
71  }
```

Essa função tem por finalidade reativar/desbloquear contas por meio da integração externa.

A função recebe todos os dados necessários para você conectar-se junto a sua API de integração e reativar a conta do cliente.

Essa função possui um nome padrão e também deve ser precedida por NOMEDOMODULO_, para seu perfeito funcionamento.

A função é executada toda vez que o operador, na tela da conta/produto, clica no botão Reativar, ou quando o pagamento de uma fatura é confirmado.

Os dados são recebidos pela variável $parametros, que trás todas as informações necessárias por meio de array.

Todas as informações estão disponíveis em chaves do array $parametros (conta, cliente, produto e servidor), tudo conforme demonstrado no item ***Funcao moduloExemplo_acao_criar***, que você pode conferir acima.

Retorno da função

IMPORTANTE: quando a função é executada com sucesso o status do produto é alterado de **suspenso** para **ativo**. Para que isso ocorra, é necessário que você forneça o retorno adequado da função em caso de sucesso.

```
45      $sucesso=true;
46
47      if($sucesso){
48          $resultado=array('status'=>'sucesso');
49      }else{
50          $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
51      }
52
53      return $resultado;
```

O retorno é sempre um array, sendo array('status'=>'sucesso'); em caso de sucesso, ou array('status'=>'erro','msg'=>'Mensagem de erro'); caso haja algum erro, sendo “Mensagem de erro”, qualquer informação que identifique o erro ocorrido.

Função moduloExemplo_acao_finalizar($parametros)

```php
function moduloExemplo_acao_finalizar($parametros){

    //Seu código para finalizar/excluir contas aqui

    $sucesso=true;

    if($sucesso){
        $resultado=array('status'=>'sucesso');
    }else{
        $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
    }

    return $resultado;
}
```

Essa função tem por finalidade **finalizar / excluir** contas do servidor externo por meio da integração.

A função recebe todos os dados necessários para você conectar-se junto a sua API de integração e finalizar a conta do cliente.

Essa função possui um nome padrão e também deve ser precedida por NOMEDOMODULO_, para seu perfeito funcionamento.

A função é executada toda vez que o operador, na tela da conta/produto, clica no **botão Finalizar.**

Os dados são recebidos pela variável $parametros, que trás todas as informações necessárias por meio de array.

Todas as informações estão disponíveis em chaves do array $parametros (conta, cliente, produto e servidor), tudo conforme demonstrado no item ***Funcao moduloExemplo_acao_criar***, que você pode conferir acima.

Retorno da função

IMPORTANTE: quando a função é executada com sucesso o status do produto é alterado de **ativo ou suspenso** para **encerrado**. Para que isso ocorra, é necessário que você forneça o retorno adequado da função em caso de sucesso.

```
45      $sucesso=true;
46
47      if($sucesso){
48          $resultado=array('status'=>'sucesso');
49      }else{
50          $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
51      }
52
53      return $resultado;
```

O retorno é sempre um array, sendo array('status'=>'sucesso'); em caso de sucesso, ou array('status'=>'erro','msg'=>'Mensagem de erro'); caso haja algum erro, sendo “Mensagem de erro”, qualquer informação que identifique o erro ocorrido.

AÇÕES ADICIONAIS

Foram demonstradas acima as principais funções dos modulos para servidores (criar, suspender, reativar e finalizar), contudo, por vezes será necessário adicionar outras funções/comandos em seu modulo, oque pode ser feito por meio de funções de comando adicionais conforme abaixo.

```php
function moduloExemplo_acao_alterar_senha($parametros){
    //Seu código para alterar senha aqui

    $sucesso=true;

    if($sucesso){
        $resultado=array('status'=>'sucesso');
    }else{
        $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
    }

    return $resultado;
}
```

As ações adicionais não possuem um nome padrão, contudo, assim como as funções padrão do modulo, seu nome deve ser precedido de moduloExemplo_acao_ (onde moduloExemplo é o nome do modulo).

Isso tem por escopo evitar conflitos com outras funções ou módulos do sistema.

Você definirá livremente qual a finalidade da função, seja alterar um plano, senha, ligar um VPS, obter informações e etc.

A função recebe todos os dados necessários para você conectar-se junto a sua API de integração e executar o comando.

A função é executada toda vez que o operador, na tela da conta/produto, clica no **botão** correspondente que será definido por você no **próximo tópico.**

Os dados necessários são recebidos pela variável $parametros, que trás todas as informações necessárias por meio de array.

Todas as informações estão disponíveis em chaves do array $parametros (conta, cliente, produto e servidor), tudo conforme demonstrado no item *Funcao moduloExemplo_acao_criar*, que você pode conferir acima.

Retorno da função

IMPORTANTE: ao ser executado, você precisa informar ao operador do sistema o status do comando (sucesso ou erro). Para que isso ocorra, é necessário que você forneça o retorno adequado da função em caso de sucesso.

```
45      $sucesso=true;
46
47      if($sucesso){
48          $resultado=array('status'=>'sucesso');
49      }else{
50          $resultado=array('status'=>'erro','msg'=>'Mensagem de erro');
51      }
52
53      return $resultado;
```

O retorno é sempre um array, sendo array('status'=>'sucesso'); em caso de sucesso, ou array('status'=>'erro','msg'=>'Mensagem de erro'); caso haja algum erro, sendo “Mensagem de erro”, qualquer informação que identifique o erro ocorrido.

Função moduloExemplo_comandos_adicionais()

```
118
119 /* ADICIONAR COMANDOS NO ADMIN(adicionar botões/comandos para as funções adicionais)*/
120 function moduloExemplo_comandos_adicionais() {
121     //adicionar outros comandos (criar, suspender, reativar e finalizar são definidos automaticamente)
122
123     $comandosAdmin=array(
124         array("comando"=>"Alterar Senha",   "funcao"=>"acao_alterar_senha"),
125         array("comando"=>"Alterar Plano",   "funcao"=>"acao_alterar_plano"),
126     );
127
128     return $comandosAdmin;
129 }
```

Essa função tem por finalidade adicionar ao painel admin (na tela da conta/produto) os botões necessários para executar as funções adicionais.

No caso do nosso exemplo, são adicionados dois botões “**Alterar Senha” e “Alterar Plano”,** que respectivamente chamam as funções **moduloExemplo_acao_alterar_senha($parametros)** e **moduloExemplo_acao_alterar_plano($parametros)**

O retorno da função deve ser o array que contem as características dos botões

Função moduloExemplo_comandos_cliente()

```
134 function moduloExemplo_comandos_cliente() {
135     //adicionar outros comandos
136 
137     $comandosCliente=array(
138         array("comando"=>"Resetar Senha",   "funcao"=>"acao_alterar_senha"),
139         /*comandos criar, suspender, reativar e finalizar não são permitidos*/
140 
141     );
142 
143     return $comandosCliente;
144 }
```

Essa função funciona de forma semelhante a função anterior, contudo, tem por finalidade **adicionar ao painel do cliente** (detalhes do serviço) os botões necessários ao gerenciamento de seu produto.

NOTA: não é permitido adicionar as funções, criar, suspender, reativar e/ou finalizar.

No caso do nosso exemplo, é adicionado o botão “**Resetar Senha”** que chama a função moduloExemplo_acao_alterar_senha.

O retorno da função deve ser o array que contem as características dos botões.

Função moduloExemplo_informacoes_admin

Essa função tem por finalidade imprimir códigos HTML na tela da conta/produto no admin. Pode ser utilizada para agregar painéis e comandos adicionais ou expor informações relacionadas aquela conta/serviço.

```
147
148  function moduloExemplo_informacoes_admin($parametros){
149
150      $code='<p>meu código aqui</p>';
151
152      return $code;
153  }
```

Os dados são exibidos no admin (na tela da conta/produto de seu cliente), ao final da página.

O retorno da função é o código HTML a ser exibido.

Função moduloExemplo_informacoes_cliente

Essa função funciona de forma idêntica a anterior, contudo, tem por finalidade imprimir códigos HTML na central do cliente (detalhes do produto/serviço).

Pode ser utilizada para agregar painéis e comandos adicionais ou expor informações relacionadas àquela conta.

```
157  function moduloExemplo_informacoes_cliente($parametros){
158
159      $code='<p>meu código aqui</p>';
160
161      return $code;
162  }
```

Os dados são exibidos central do cliente (na tela de detalhes do produto/serviço), ao final da página.

O retorno da função é o código HTML a ser exibido.

Com as informações apresentadas você será capaz de desenvolver módulos de integração para servidores, compatíveis com o HOSTMGR.

Opcionalmente caso queira distribuí-lo você poderá contar com nosso sistema de licenças para disponibilizar e gerenciar suas chaves e clientes.

Caso haja qualquer dúvida contate nosso suporte por meio de ticket em nossa central do cliente (http://central.hostmgr.com.br). Criamos um departamento específico para desenvolvedores.