



SCOPE

Manual de Integração

Índice

Prefácio	11
A quem se destina.....	11
Requisitos	11
Descrição do Produto	11
Conceitos	12
Transações	12
Fluxo de estados de coleta	13
Interfaces de interação com o SCOPE Client	13
Padrões adotados neste documento	14
Instalação	15
Instalação do SCOPE Client para MS-Windows®	15
Instalação do SCOPE Client para Linux.....	16
Configuração	16
Configuração do arquivo scope.ini.....	16
Configuração do registro do MS-Windows®	24
Configuração de Porta Automática	25
Fontes de Dados do SCOPE no WoW64	26
Carregamento Dinâmico da biblioteca.....	26
Funções básicas da API do SCOPE Client.....	26
Comunicação com o ScopeSRV	27
Sessão de transação	30
Status de transação	34

Funções de configuração de ambiente	36
Configurações gerais.....	36
Configuração de PIN-Pad.....	37
Funções específicas das interfaces	38
Interface coleta	39
TEF	48
Cartão de crédito	49
Cartão de débito	66
Carteira Virtual	76
Cartão Dinheiro	83
Funções de Consulta	87
Cheque	87
AVS.....	88
Pagamento	90
Status do Cartão	91
Recarga de celular.....	92
Configurando a recarga de celular	92
Processando a recarga de celular	92
Funções de consulta para recarga de celular	102
Recarga de Celular para a rede VERO BANRISUL R17	104
Recarga de Celular para a rede BRADESCO	109
Recarga de Celular para as redes VERO BANRISUL e BRADESCO em “modo compatibilidade”..	115
Estorno de transações.....	117
Comprovantes.....	120

Comprovantes de transações	120
Reimpressão de comprovante	122
PBM – Medicamentos.....	126
Consultando medicamento	126
Compra de medicamento	128
Fornecendo Lista de Medicamentos	131
Lista de medicamentos	133
Lista de medicamentos com CRM	134
Lista de medicamentos Estendida	136
Lista de Projetos	137
Elegibilidade do Cartão PBM	138
Pré-Autorização de Medicamentos PBM	139
Cancelamento de Pré-Autorização de Medicamentos PBM	140
Operações Fidelidade	142
Tipos de Operação Fidelidade.....	142
Pinpad Compartilhado e ABECS.....	143
Funcionamento	144
Comandos de controle	144
Comandos.....	149
Totalização de TEF	167
Relatório de TEF.....	168
Transação de POS para conciliação	174
Descrição da funcionalidade	174
Transação POS	174

Redes e bandeiras.....	177
Códigos de serviços	179
Cupom CIELO X ScopeObtemCampoExt2 X arquivo	180
Cupom REDE X ScopeObtemCampoExt2 X arquivo	182
Outros cupons X ScopeObtemCampoExt2 X arquivo	185
Formatação para o arquivo de conciliação.....	186
Funções de Pagamento Recorrente	187
Cadastro de Pagamento Recorrente.....	187
Inclusão de Cartão	193
Identificação da Transação	197
PIN Impresso.....	201
Consulta Conteúdo Digital	201
Efetiva Conteúdo Digital	203
ScopeObtemListaRedesConteudoDigital	206
ScopeObtemProvedoresConteudoDigital	206
ScopeObtemProdutosConteudoDigital.....	208
Funções diversas.....	209
Dados da transação.....	209
Fornecendo informações extras para a transação.....	215
ScopeGetLastMsg.....	222
ScopeGetCheque	222
ScopeAtualizaValor.....	223
ScopeGarantiaDescontoCheque.....	224

ScopeTransacaoFinanceira.....	224
ScopeSaque	227
ScopeInvestimento	227
ScopeObtemCartaoInvestimento.....	228
ScopeResumoOperacoes	229
ScopePagamento	230
ScopeServiçoTécnico	232
ScopeAtualizaParametrosChip.....	233
ScopeVersao.....	234
ScopeAtualizaPrecosMercadorias	234
ScopeServiçosGenericos.....	235
ScopeObtemDadosAdicionais	237
ScopeObtemProdutosFrota.....	238
ScopeObtemDadosCredíarioCredito.....	239
ScopeCompraCartaoCreditoCPF	244
ScopeConsultaPgtoFatura	246
ScopeConsultaPgtoFaturaCPF	247
ScopeConsultaSaldoCreditoCPF	248
ScopeTransacaoFinanceiraCPF.....	249
ScopePagamentoCPF	251
ScopeConsultaPreAutorizacão.....	252
ScopeCreditoPagamentoFatura	253
ScopeStartLog	254

ScopeStopLog.....	254
Operações com tratamentos específicos	255
CORBAN – Pagamento de Contas.....	255
PIX – Pagamento via QRCode, PIX Saque e PIX Troco.....	260
Redes com tratamentos específicos.....	261
VeroBanrisul.....	261
Ticket Edenred	263
Cielo Premia	269
Cielo Auto	274
SAVS	278
Cielo – Transações Sem Contato (Contactless).....	282
Valecard	283
SGF.....	285
REDE V06.xx.....	294
Getnet – Transações Sem Contato (Contactless).....	298
Redes Padrão Lojista Frota	298
VeroBanrisul – Transações Sem Contato (Contactless).....	301
Ticket Log – Transações Frota	302
FIC.....	304
GetNetLAC v2.99	305
Bradescard	307
Hub	313
RAPPI	314

Cielo – Parcelado Cliente	315
Cielo – DCC Digitado	316
STIX – Fidelidade	321
Perguntas Frequentes – Geral.....	322
Apêndice A – Tabelas.....	324
Códigos de retorno	324
Formatos dos dados	338
Códigos de Fluxo	339
Código das redes	340
Código de especificação das redes.....	344
Código das bandeiras	346
Dados disponíveis das transações.....	354
Grupo de Serviços	358
Códigos dos Serviços.....	358
Convênios	362
Modo de entrada.....	362
Apêndice B – Especificação Visanet 4.1.....	363
Adequação	363
Certificação.....	364
Apêndice C – PIN-Pad Compartilhado	365
Apêndice D – Conjunto de bibliotecas do SCOPE Client.....	366
MS-WINDOWS®	366
Linux.....	367
Android.....	368

Apêndice E – Android.....	369
Configuração do projeto no Android Studio.....	369
Configurando o AndroidManifest.....	373
Diretório /scope	374
Arquivo Scope.ini	375
Captura de arquivos de log do POS	375
PIN-Pad Bluetooth	376
PIN-Pad USB	378
POS.....	382
Métodos básicos	393
Apêndice F – Identificando a versão do SCOPE Client	401
Verificando no SCOPE Server	401
Verificando no ambiente do PDV	402
Apêndice G – Formato do Código de Barras InComm.....	405
Apêndice H – iOS	406
Adicionando a biblioteca do SCOPE	406
Adicionando o SCOPEAPI.H.....	407
Importando o Framework ExternalAccessory	407
Configurando o arquivo INFO.PLIST	409
Adicionando os protocolos de hardware	409
SCOPE.PLIST.....	410
Apêndice I – POS Ingenico	411
Ferramentas para Desenvolvimento	411
Módulos SCOPE POS	411

Prefácio

A quem se destina

O manual do integrador destina-se aqueles que desenvolvem aplicação em que é necessário efetuar transações com autorizadoras, banco, etc. e para isso utilizarão o SCOPE como concentrador de TEF.

Requisitos

Plataformas Compatíveis

Todos os módulos da solução SCOPE, são compatíveis com as plataformas Windows Server 2008 ou superior.

O módulo SCOPE Client, além das plataformas acima, também é compatível com a plataforma Linux (Librix, RedHat, SUSE e outros).

Requisitos de HW

- PC com processador de 1 GHz ou mais de velocidade
- 512 megabytes (MB) de RAM ou mais são recomendados
- 1 gigabyte (GB) de espaço disponível em disco rígido ou mais são recomendados

Requisitos de SW

Banco de Dados (SGBD). Os seguintes SGBDs são suportados pela solução SCOPE:

- SQL Server 2008 ou superior
- Oracle 11g ou superior

Descrição do Produto

Objetivo

O sistema **SCOPE – Solução Completa para Pagamento Eletrônico** – permite ao software de **PDV** efetuar o pagamento através de **TEF**, por exemplo, cartões de crédito e débito, através da digitação do emboço ou leitura da tarja magnética ou chip.

Definição

O **SCOPE** foi concebido para gerenciar todas as etapas de uma TEF abstraindo a complexidade inerente à transação do aplicativo PDV.

Benefícios

- O sistema permite uma arquitetura flexível, multi-empresa, multi-filial, centralizada ou distribuída, adequando-se à necessidade do cliente.
- Facilita a gestão do negócio, pois oferece ferramentas de consulta e relatórios operacionais e gerenciais.

- Permite a gestão da sessão contábil, pelo estabelecimento ou pelo centro de processamento, de forma a indicar eventuais divergências nos créditos efetuados pelos bancos.
- Permite mecanismos de alta-disponibilidade através de solução cluster e rotas de contingência.

Conceitos

Transações

O SCOPE provê à loja diversos tipos de transações (exemplo: compra com cartão de crédito, compra com cartão de débito, recarga de celular, pagamento de contas e de fatura de cartões, etc.). Numa transação, o SCOPE Client comunica-se com o SCOPE Server baseado em mensagens, as quais seguem a norma ISO 8583 que especifica o protocolo de mensagens para transações financeiras com cartão. As mensagens básicas e as suas sequências que geralmente compõem uma transação completa e bem sucedida são demonstradas na **Figura 1** e descritas abaixo. Observe que a informação apresentada abaixo não é exposta à interface de programação, compreendendo um fluxo interno entre SCOPE Client e SCOPE Server.

- 9604 – mensagem de solicitação da pré-TEF: esse tipo de mensagem é geralmente enviado duas vezes com finalidade diferentes (primeira e segunda pré-TEF). A primeira pré-TEF contém o BIN de um cartão e um grupo de serviço além de outros dados exigidos numa mensagem ISO. A segunda abrange o produto selecionado.
- 9614 – mensagem de resposta da pré-TEF. Contém os dados solicitados pelo SCOPE Client. Como a mensagem 9604, esta diferirá de acordo com o contexto da pré-TEF. A resposta da primeira é a rede autorizadora, a bandeira, os serviços disponíveis e os atributos (ex.: limite de parcelas, data limite de agendamento, etc.) dos produtos habilitados para o cartão capturado. A da segunda define para o SCOPE Client como as mensagens das transações deverão ser montadas.
- 0200 – mensagem de solicitação da transação financeira. Neste momento, a mensagem vai até a rede autorizadora para que seja autorizada.
- 0210 – mensagem de resposta da transação financeira. A autorizadora responde a solicitação com esta mensagem que abrange a aprovação ou a rejeição da solicitação.
- 0202 – mensagem de confirmação da transação financeira. Esta mensagem é a que confirma que a transação foi realmente efetivada. Ela indica, por exemplo, que o comprovante foi fisicamente impresso e que o processo como um todo foi concluído.

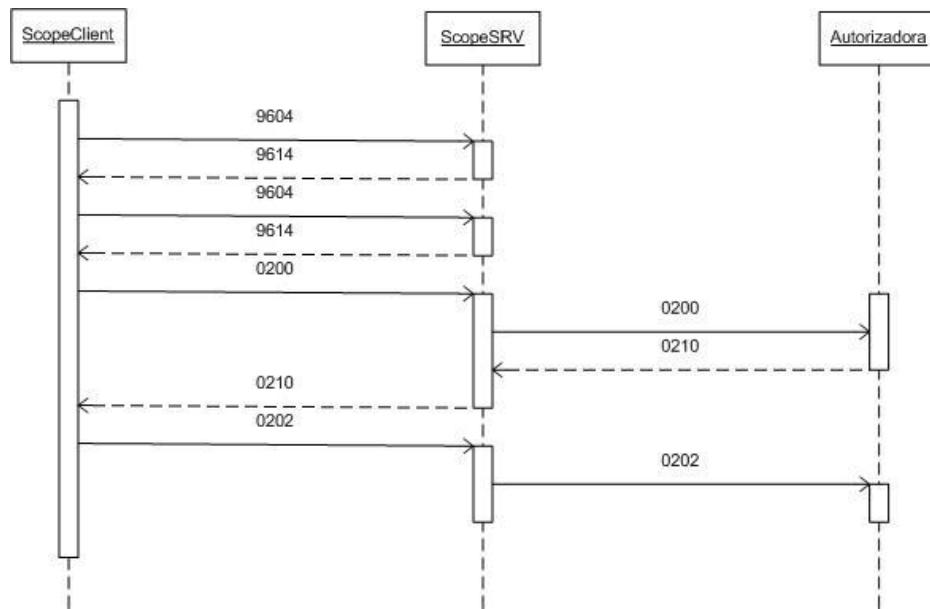


Figura 1: sequência de mensagens numa transação completa e bem sucedida.

Fluxo de estados de coleta

O funcionamento do SCOPE Client baseia-se em fluxos de estados de coleta para as diversas operações disponíveis. A cada iteração entre o aplicativo com o SCOPE Client, este informa em qual estado de coleta se encontra, qual informação deve ser coletada, qual mensagem deve ser exibida e quais opções de fluxo estão disponíveis (retornar, próximo e cancelar). Desta forma o fluxo avança até que seja possível efetuar a transação desejada.

Exemplificando: Ao iniciar uma transação de crédito, o SCOPE estará no estado de "coleta número do cartão". Uma vez obtido o cartão, o estado avança e dependendo de uma série de variáveis, poderá migrar para o estado "coleta os 4 últimos dígitos do cartão". Conforme as configurações possíveis para o tratamento de serviço de crédito, o próximo estado poderá ser "coletar se à vista ou não", o fluxo avança até a impressão do comprovante.

Interfaces de interação com o SCOPE Client

Antes de começar o desenvolvimento da integração com o SCOPE é fundamental conhecer como a aplicação irá interagir com o SCOPE Client. A interface disponível é chamada de Interface coleta e está descrita a seguir:

Interface coleta

Disponível para todos os sistemas operacionais com os quais o SCOPE Client é executado, a interface coleta do SCOPE Client é o que apresenta uma maior interação entre a aplicação de PDV e o SCOPE. Através desta interface, a aplicação se torna responsável pela coleta dos dados digitados pelo operador ou cliente conforme a solicitação a cada iteração do SCOPE. Como a aplicação coletará os dados, ela se responsabilizará pela exibição da mensagem na tela e a entrada de dados para o operador, sendo que para alguns casos, deverá

tratar também a limitação do tamanho do campo aceitável (ex.: para a coleta do número de segurança do cartão, a aplicação permitirá a entrada de um valor com no mínimo 3 e no máximo 5 dígitos).

Características da interface coleta

Interceptar, para uso da própria aplicação, os dados coletados via digitação para o SCOPE;

- Não interfere na interface gráfica do usuário;
- Disponível para qualquer linguagem e sistema operacional;
- A aplicação deve ter um maior nível de “especialização”, portanto, maior lógica na integração.

Padrões adotados neste documento

Este documento descreve funções padronizadas no seguinte formato:

```
LONG EXPORT ScopeNomeDaFuncao (argumentos...)
```

As sessões que descrevem os argumentos ou parâmetros das funções exibirão os mesmos numa tabela, cujas linhas representam cada parâmetro e as colunas indicarão os seguintes dados:

- 1^a coluna: parâmetro de entrada [in] ou saída [out];
- 2^a coluna: tipo de dado que representa o parâmetro;
- 3^a coluna: nome da variável representada no parâmetro
- 4^a coluna: significado do parâmetro.

Todos os exemplos de códigos relacionados neste documento estão seguindo a linguagem C com a seguinte formatação:

```
int main ()  
{  
    printf ("Hello, SCOPE\\'s Programmers!");  
}
```

Para compatibilidade das funções entre diversas linguagens e plataformas, algumas convenções de tipos devem ser adotadas, conforme a tabela a seguir:

Definição de Tipo:

Tipo	Tamanho	Descrição
BYTE	1 byte	Valor sem sinal de 0 a 255
WORD	2 bytes	Valor sem sinal de 0 a 65.535
SHORT	2 bytes	Valor com sinal de -32.768 a 32.767
LONG	4 bytes	Valor com sinal de -2.147.483.648 a 2.147.483.647

Analogamente, como recurso de portabilidade, algumas constantes devem ser criadas e definidas diferentemente em cada plataforma, com o objetivo de utilizar os mesmos protótipos de funções conforme a tabela a seguir:

Constantes:

Tipo	Descrição
EXPORT	Usada nos protótipos das funções que são exportadas
CALLBACK	Função cujo endereço de entrada é fornecido a outras funções, de modo que estas possam utilizar internamente da primeira

Legenda de Abreviações

Formato	Atributo
a	Caracteres alfabéticos.
n	Caracteres numéricos.
an	Caracteres alfabéticos e numéricos.
ans	Caracteres alfabéticos, numéricos e especiais.
MM	Mês.
DD	Dia.
AA	Ano.
hh	Hora.
mm	Minuto.
ss	Segundo.
LLvar	Tamanho de um campo variável. Ex. Se o conteúdo do campo = "AB1234CD", teremos para representá-lo: "08 AB1234CD".
LLLvar	Tamanho de um campo variável. Ex. Se o conteúdo do campo = "AB1234CD", teremos para representá-lo: "008 AB1234CD".
10	Tamanho fixo de 10 caracteres.
..10	Tamanho variável de até 10 caracteres.
b	Representação binária dos dados.

Instalação

Nesta sessão é comentada a localização das bibliotecas do SCOPE Client em cada sistema operacional. Quanto à composição do SCOPE Client, deve-se consultar o [Apêndice E – Conjunto de bibliotecas do SCOPE Client](#).

Instalação do SCOPE Client para MS-Windows®

As bibliotecas do SCOPE Client para MS-Windows® devem estar em um diretório acessível pela aplicação. A decisão da localização destas bibliotecas é tomada pela equipe que desenvolve a aplicação de PDV, porque a particularidade da linguagem e/ou ambiente de desenvolvimento tem impacto sobre o acesso a elas.

Normalmente, o SCOPE Client é colocado no mesmo diretório da aplicação integradora. No entanto, há casos em que ele está disponível na pasta do sistema %windir%\System32.

CUIDADO: devido à perda de controle sobre qual biblioteca o MS-Windows® faz referência, não é aconselhável que as bibliotecas estejam no diretório do MS-Windows® ou em qualquer subdiretório.

No CD de instalação do SCOPE, há possibilidade de instalar o SCOPE Client na máquina. Esta instalação executará os seguintes passos:

- criará um diretório com o SCOPE Client;
- copiará o executável do HotKey e seu atalho na área de trabalho;
- alterará o registro do MS-Windows® com os itens de configuração.

O SCOPE se utiliza de alguns arquivos de controle (no formato SXXXXYYYYZZZAAA.sc_), gerados em tempo de execução. Assim, em sistemas operacionais com conceito de multiusuários, como Linux e MS-Windows, existe a necessidade de permissão para escrita e leitura no diretório da aplicação. No caso do MS-Windows, tais arquivos serão gerados em 'C:\'.

Instalação do SCOPE Client para Linux

Para a instalação do SCOPE Client em ambiente Linux, é disponibilizado um arquivo no formato RPM. Este arquivo é um pacote do sistema RPM (Red Hat Package Manager), utilizado em várias distribuições Linux. O nome do arquivo liberado segue o padrão:

itautec-scope-<versão>-1.i386.rpm

Onde: <versão> representa a versão e o release do SCOPE Client. Para a sua instalação execute o comando:

\$ rpm -i itautec-scope-<versão>-1.i386.rpm

Este comando instalará as bibliotecas na pasta /usr/lib.

IMPORTANTE: para a execução deste comando é necessário ter direitos do usuário root.

CUIDADO: em ambiente Linux, o usuário root tem o direito de realizar comandos que podem danificar o sistema.

LEMBRETE: podem-se acrescentar os parâmetros “-vh” para exibir o progresso da instalação.

Configuração

Configuração do arquivo scope.ini

O arquivo **scope.ini** é responsável por parte da configuração da aplicação. Ele deve estar disponível com o SCOPE Client, independente do sistema operacional que a aplicação executa.

Sessão [<empresa><filial>]

Esta sessão é a única obrigatória para que o SCOPE Client possa se conectar ao ScopeSRV. Dentro dos colchetes deve haver uma sequência de 8 dígitos, sendo que os 4 primeiros representam o código da empresa e os demais, o código da filial cadastrado no SCOPE. Estes códigos devem ser iguais aos utilizados como parâmetros da função [ScopeOpen\(\)](#). Os valores das chaves configuráveis nesta sessão se encontram na tabela abaixo.

Chave	Significado	Valor	
Name	Endereço ou nome da máquina em que está o ScopeSRV.	I.P. ou nome da máquina do ScopeSRV	
Port	Porta de conexão do ScopeSRV.	Valor numérico (o padrão é 2046).	
AtualizaDataHora	Parâmetro que decide se o SCOPE Client deverá alterar a data e a hora da máquina para sincronizar com o ScopeSRV no momento da conexão.	N	se não deve sincronizar (padrão).
		S	caso deseja-se que sincronize.
MsgOperReduzida	Controla o tamanho máximo de colunas das mensagens que o SCOPE Client enviará para a aplicação para que esta exiba no teclado do operador ou display com esta finalidade.	N	40 colunas (padrão)
		s	20 colunas
		m	16 colunas
ShowCupom	Configura a exibição do cupom na tela. (Não utilizado na Interface Coleta).	n	Não exibe
		s	Exibe (padrão)
TimeOutLogon	Configura o tempo, em segundos, de espera para a conexão do SCOPE Client com o ScopeSRV.	Valor numérico entre 10 e 60 (padrão é 10).	
TimeOutAdm	Configura o tempo, em segundos, de espera da resposta da autorizadora em cada transação.	Valor numérico entre 15 e 180 (padrão é 30).	
CupomReducido	Configura se deve exibir o cupom reduzido na tela. (Não utilizado na Interface Coleta)	n	Não exibir
		s	Exibir
VersaoAutomacao	Versão da automação passada pela área de certificação da Visanet.	Sequência de caracteres no formato RRAAAACCCC, sendo que: RR à release de certificação do TEF AAAA à Nome da automação CCCC à Código do memorando xxx (o padrão é "01HOTK0000")	
WKPAN	Configura se deve habilitar a Comunicação Segura com o PIN Pad.	n	Não habilitar
		s	Habilitar
ScopeValidaSaque	Configura se o Scope deve consistir o valor do saque digitado com os valores configurados nos limites de saque do produto, localizados no ScopeCnf.	n	Não habilitar
		s	Habilitar

MascararDados	<p>Indica como o valor do PAN deve ser enviado para a aplicação.</p> <p>ATENÇÃO: a configuração do parâmetro MascararDados=n poderá acarretar em não conformidade com o PCI-DSS.</p>	n	Nenhum cartão é mascarado. Para bandeiras que estão em conformidade com o PCI-DSS, esse valor não tem efeito, ou seja, o cartão estará sempre mascarado.
		s	Todos os cartões serão mascarados com zeros.
		s(0)	Todos os cartões serão mascarados com zeros.
		s(*)	Todos os cartões serão mascarados com asteriscos.
		p	Todos os cartões que seguem o padrão PCI DSS serão mascarados com zeros.
		p(0)	Todos os cartões que seguem o padrão PCI DSS serão mascarados com zeros.
		p(*)	Todos os cartões que seguem o padrão PCI DSS serão mascarados com asteriscos.
GenTimeout	Define um tempo máximo, em segundo, para o SCOPE Client gerar as chaves de criptografia utilizado na conexão.		Valor numérico de 1 à 99 que representa o tempo em segundos. Se não tiver configurado ou o valor estiver inválido, será assumido o valor padrão é de 99 segundos. Neste caso, será praticamente impossível alcançar esse tempo e a chave ideal será gerada sempre.
RSAMaxTentativas	Configura o número máximo de tentativas para geração de chave RSA, para a Comunicação Segura com o PIN Pad.		Valor numérico entre 11 e 99, que representa o número máximo de tentativas. Se esse parâmetro não estiver configurado, o valor padrão são 10 tentativas.

As chaves mínimas exigidas são as duas primeiras (Name e Port), enquanto as outras são opcionais e dependerá da situação de cada aplicação.

Exemplo: para a configuração de um PDV cadastrado na empresa 0001 e filial 0007, que irá se conectar ao ScopeSRV no IP 10.50.9.70, o scope.ini deverá ser configurado da seguinte maneira:

```
[00010007]
Name=10.50.9.70
Port=2046
VersaoAutomacao= "01CAPS0001" ; valor fictício
```

Sessão [SCOPEAPI]

Esta sessão não é obrigatória e define os parâmetros abaixo relacionados ao ambiente:

Chave	Significado	Valor	
SaveCupom	Diretório para salvar o cupom	Diretório (padrão é "./")	
TracePin (*)	Habilita o log do PIN-pad compartilhado. Em MS-Windows, será gerado no 'C:\', nos demais sistemas, no diretório corrente da aplicação, o arquivo 'TracePin.sc_'. O sistema de log é circular, assim quando o tamanho desse arquivo for 1 MB, ele será renomeado para 'TracePin.bak' e um novo 'TracePin.sc_' será criado.	n s	Não habilita (padrão) Habilita
TraceAPI (*)	Habilita o log do SCOPE Client que mostra a integração com a automação comercial.	n s	Não habilita (padrão) Habilita
TraceSrl (*)	Habilita o log do SCOPE Client que mostra a integração com a serial. OBSERVAÇÃO: devido o conteúdo de alguns comandos possuírem dados confidenciais, parte do log é mascarado com dados fixos.	n s	Não habilita (padrão) Habilita
PBMNaoSolicitaCartao	Esse parâmetro é responsável por deixar de solicitar o número do cartão durante uma transação de consulta PBM.	n s	Solicita cartão no pin pad (padrão). Não solicita cartão no pin pad
ArqControlPath	Diretório para salvar os arquivos de controle do Scope Client. O diretório padrão é a raiz do sistema (c:\). <i>OBSERVAÇÃO: válido apenas para o sistema operacional MS-Windows</i>	Diretório	
ArqTracePath	Diretório para salvar os arquivos de log gerado pelo SCOPE Client. No MS-Windows, o diretório padrão é a raiz do sistema (c:\). No Linux, o padrão é o diretório da aplicação.	Diretório	

(*) A partir da versão 3.01.21.005, essas chaves foram substituídas por novas sessões e chaves (veja tabela abaixo) devido à utilização de uma nova biblioteca de geração de logs (csmg.dll no Windows e libcsmg.so no Linux). A aplicação Windows deverá utilizar as funções ScopeStartLog e ScopeStopLog para que a biblioteca csmg.dll seja carregada dinamicamente.

Sessão [SCOPELOGAPI]

Os parâmetros dessa sessão habilitam as mensagens do módulo scopeapi relacionadas às interfaces com a aplicação.

Chave	Significado	Valor
TraceLevel	Indica o nível de mensagens que serão impressas no arquivo de log.	O nível de varia de 0 a 8, sendo 0 o que imprime nenhuma

	ATENÇÃO: a configuração do parâmetro TraceLevel=0 poderá acarretar em não conformidade com o PCI-DSS	mensagem até 8 que imprime todas as mensagens que estão no programa.
LogPath	Indica o caminho do local onde serão gerados os arquivos de log.	No windows, utilizar o backslash '\\'. No Linux, utilizar o forwardslash '/'
LogFiles	Quantidade de arquivos a serem criados	Valor inteiro de 1 a n. Utilizar em conjunto com a chave LogSize.
LogSize	Tamanho máximo de um arquivo de log gerado. Ao atingir o tamanho máximo, o programa começa um novo arquivo ou sobrepõe um existente.	Valor inteiro em bytes. Utilizar a estratégia de colocar um tamanho suficiente grande que comporte as transações de 1 dia. O valor depende da situação, porém evitar que tenha um tamanho muito grande. Utilize em conjunto com a chave LogFiles.

O(s) arquivo(s) gerado(s) possue(m) o formato:

ScopeLogAPI.txt.xxx

Onde xxx pode assumir os valores 000 até o número indicado em LogFiles – 1.

Sessão [SCOPELOGPRF]

Os parâmetros dessa sessão habilitam as mensagens do módulo scopeprf relacionadas aos comandos da biblioteca de interface com o pinpad.

Chave	Significado	Valor
TraceLevel	Indica o nível de mensagens que serão impressas no arquivo de log. ATENÇÃO: a configuração do parâmetro TraceLevel=0 poderá acarretar em não conformidade com o PCI-DSS	O nível de varia de 0 a 8, sendo 0 o que imprime nenhuma mensagem até 8 que imprime todas as mensagens que estão no programa.
LogPath	Indica o caminho do local onde serão gerados os arquivos de log.	No windows, utilizar o backslash '\\'. No Linux, utilizar o forwardslash '/'
LogFiles	Quantidade de arquivos a serem criados	Valor inteiro de 1 a n. Utilizar em conjunto com a chave LogSize.

LogSize	Tamanho máximo de um arquivo de log gerado. Ao atingir o tamanho máximo, o programa começa um novo arquivo ou sobrepõe um existente.	Valor inteiro em bytes. Utilizar a estratégia de colocar um tamanho suficiente grande que comporte as transações de 1 dia. O valor depende da situação, porém evitar que tenha um tamanho muito grande. Utilize em conjunto com a chave LogFiles.
---------	--	---

O(s) arquivo(s) gerado(s) possue(m) o formato:

ScopeLogPRF.txt.xxx

Onde xxx pode assumir os valores 000 até o número indicado em LogFiles – 1.

Sessão [SCOPELOGSRL]

Os parâmetros dessa sessão habilitam as mensagens do módulo scopeprf relacionadas aos comandos de interface de comunicação com o firmware do pinpad.

Chave	Significado	Valor
TraceLevel	Indica o nível de mensagens que serão impressas no arquivo de log. ATENÇÃO: a configuração do parâmetro TraceLevel=0 poderá acarretar em não conformidade com o PCI-DSS	O nível de varia de 0 a 8, sendo 0 o que imprime nenhuma mensagem até 8 que imprime todas as mensagens que estão no programa.
LogPath	Indica o caminho do local onde serão gerados os arquivos de log.	No windows, utilizar o backslash '\\'. No Linux, utilizar o forwardslash '/'
LogFiles	Quantidade de arquivos a serem criados	Valor inteiro de 1 a n. Utilizar em conjunto com a chave LogSize.
LogSize	Tamanho máximo de um arquivo de log gerado. Ao atingir o tamanho máximo, o programa começa um novo arquivo ou sobrepõe um existente.	Valor inteiro em bytes. Utilizar a estratégia de colocar um tamanho suficiente grande que comporte as transações de 1 dia. O valor depende da situação, porém evitar que tenha um tamanho muito grande. Utilize em conjunto com a chave LogFiles.

O(s) arquivo(s) gerado(s) possue(m) o formato:

ScopeLogSRL_eeeeffffppp.txt.xxx

Onde:

xxx pode assumir os valores 000 até o número indicado em LogFiles – 1

eeee é código numérico da empresa

ffff é o código numérico da filial

ppp é o código numérico do terminal

Sessão [PPCOMP]

Aqui, configuram-se itens relacionados ao PIN-Pad compartilhado

Chave	Significado	Valor	
NaoAbrirDigitado	Configura para não abrir o digitado se cancelada a leitura do cartão no PIN-Pad compartilhado.	s	não abre digitação (padrão)

Configuração de porta serial para Linux

Atualmente, o SCOPE Client é preparado para trabalhar apenas com PIN-Pad conectados à porta serial. No entanto, para ambiente Linux, é possível conectar um PIN-Pad serial numa porta USB, utilizando um conversor SerialBàUSB. Normalmente os drivers existentes para esse conversor criam dispositivos (em /dev) nomeados **ttyUSBx**, ao invés dos **ttySx**. Para o redirecionamento da porta de comunicação no SCOPE, existem duas chaves a serem colocadas no arquivo scope.ini, configuradas conforme tabela abaixo.

Chave	Significado	Valor
SerialNumPorts	Número de portas a ser configuradas.	Valor entre 1 e 6 (o padrão é 6)
SerialPortx	Dispositivo com novo direcionamento para a porta (x-1).	Valores de 0 a 5.

LEMBRETE: a configuração pode ser alternada entre as diversas portas.

Exemplo: suponhamos que possuímos 3 equipamentos seriais, mas o computador que rodará a aplicação possui uma porta serial e duas USB. Será necessário que dois equipamentos utilizem conversores SerialBàUSB, que ao serem conectados, são criados dois novos dispositivos: ttyUSB0 e ttyUSB1 (os números terminais – 0 e 1 – podem variar). Portanto, em algum ponto do arquivo scope.ini, deve ser colocado a configuração abaixo para atender esta máquina:

```
SerialNumPorts=3
SerialPort0=/dev/ttyS0 ; porta 1 serial
SerialPort1=/dev/ttyUSB0 ; porta 2 USB
SerialPort2=/dev/ttyUSB1 ; porta 3 USB
```

CUIDADO: de acordo com a configuração acima, conectamos um equipamento na porta 1 e os outros dois, nas portas 2 e 3.

Sessão [PINPAD]

Aqui, configura-se item relacionado ao tamanho mínimo de dados permitidos para serem lidos pelo PIN-Pad.

Chave	Significado	Valor	
TamMinDados	Tamanho mínimo de entrada de dados permitidos pelo PIN-Pad.	Valor numérico acima de 1 (o padrão é 1).	
PPTLIOTE	Indica se o Scope realizará o tratamento de envio da carga de tabelas para o pinpad de forma otimizada, isto é, quando um registro da carga é perdido devido a problema de comunicação com o server, a retransmissão ocorra a partir do registro que deu erro.	n	Não utiliza o recurso (default)
		s	Utiliza o recurso

CUIDADO: Caso o PIN Pad utilize a Biblioteca Compartilhada 1.07 e PCI 2.0 a quantidade mínima de dígitos permitidos para serem lidos por coleta deve ser 4 dígitos, sendo assim, TamMinDados deve estar parametrizado com um valor de no mínimo 4.

Seção [SCOPEAPIPOS]

Essa seção permite configurações gerais, como para habilitar a digitação dos códigos e acrescentar redes e/ou bandeiras.

Chave	Significado	Valor	
DigitacaoCodigo	Configura se dá a opção de outra rede. ATENÇÃO: essa chave é ignorada se a chave Complementar existir.	n	não pergunta se é outra rede
		s	pergunta se é outra e coleta o código da rede e da bandeira
Complementar	Configura se acrescenta rede à tabela padrão ou as substitui. ATENÇÃO: a existência dessa chave desconsidera a configuração chave DigitacaoCodigo, pois ela é prioritária.	n	substitui as tabelas padrão pelas que estão configuradas
		s	acrescenta a configuração de redes e bandeiras que estão no arquivo a listagem padrão
Redes	Lista de até 7 códigos de redes distintas para perguntar no fluxo	Números de até 3 dígitos cada, separados por vírgula	
Bandeiras	Lista de até 14 códigos de bandeiras distintas para perguntar no fluxo	Números de até 3 dígitos cada, separados por vírgula	
NomeBandeiraYYY	Nome da bandeira YYY que será exibida no fluxo, onde YYY é uma das bandeiras listadas na chave Bandeiras. ATENÇÃO: se faltar a chave para alguma das bandeiras, será exibido código da bandeira para o operador.	String de até 20 caracteres	
ColetasOpcionais	Lista de estados de coleta que serão executados de forma opcional durante o fluxo de coleta.	Valores válidos (separados por vírgula): data-de-validade nome-do-portador	

Seção [SCOPEAPIPOS-RXXX]

Essa seção permite configurações de uma das redes configuradas na seção detalhada anteriormente, onde XXX é uma das redes listas na chave Redes. Portanto, deve haver uma seção para cada rede da lista.

ATENÇÃO: se não houver a seção para alguma das redes listadas, será exibido o código da rede como opção e todas as bandeiras listadas estarão associadas à rede.

Chave	Significado	Valor
Nome	Nome da rede XXX que será exibida no fluxo	String de até 20 caracteres
Bandeiras	Lista de bandeiras associadas a esta rede. A bandeira deve existir na chave Bandeiras da seção [SCOPEAPIPOS].	Números de até 3 dígitos cada, separados por vírgula
ServicoBXXX	Lista de códigos de serviços permitidos para essa bandeira, associada a essa rede, onde XXX é o código com 3 dígitos (zeros à esquerda) de uma das bandeiras listada no campo Bandeiras. Se houver bandeiras sem configuração de serviço, serão associados todos os serviços previstos.	Números de até 3 dígitos cada, separados por vírgula

Configuração do registro do MS-Windows®

No sistema operacional MS-Windows® é possível configurar o SCOPE Client utilizando o registro do sistema operacional. Abrindo o editor de registro, as configurações do SCOPE devem estar em **HKEY_LOCAL_MACHINE\SOFTWARE\Scope**:

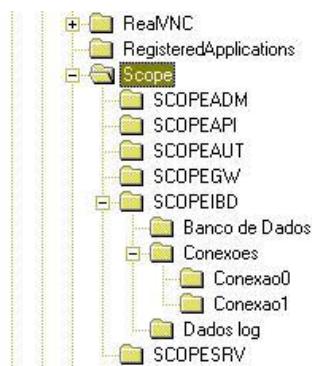


Figura 2: árvore do SCOPE no registro do MS-Windows®

Em sistemas operacionais MS-Windows® de 64 bits (x64), o SCOPE é instalado sobre o subsistema Wow64 para funcionar corretamente. No WoW64 as configurações do SCOPE são armazenadas em **HKEY_LOCAL_MACHINE/SOFTWARE/Wow6432Node/Scope**:

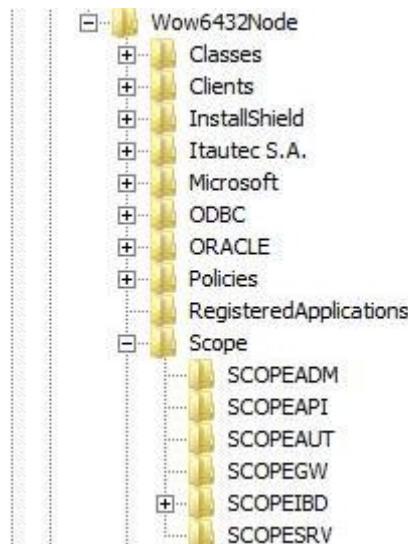


Figura 3: árvore do SCOPE no registro do Wow64®

Para maiores informações sobre o Wow64, consulte a sessão "**Instalação do SCOPE em Sistemas Operacionais Microsoft Windows® de 64-Bits**", no documento "Scope – Manual de Instalação e Configuração".

Relacionado na tabela abaixo, estão as chaves do scope.ini com o registro do MS-Windows®.

Chave do arquivo scope.ini	Localização no editor do registro
AtualizaDataHora	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\AtualizaDataHora
CupomReduzido	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\CupomReduzido
MsgOperReduzida	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\MsgOperReduzida
Name	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPESRV\Name
NaoAbrirDigitado	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\NaoAbrirDigitado
Port	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPESRV\Port
ShowCupom	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI>ShowCupom
TimeOutAdm	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\TimeOutAdm
TimeOutLogon	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\TimeOutLogon
VersaoAutomacao	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\VersaoAutomacao
ScopeValidaSaque	HKEY_LOCAL_MACHINE\SOFTWARE\Scope\SCOPEAPI\ScopeValidaSaque

Configuração de Porta Automática

A partir da versão 3.01.19.xxx, é possível realizar a configuração do Scope de maneira que ele reconheça automaticamente em qual porta do computador o PinPad se conecta. Para isso, é preciso satisfazer as seguintes condições:

Indicar no ScopePSW na tela de configuração de perfil de hardware e escolher a opção AUTO para a porta do pinpad selecionado.

Realizar a chamada da nova função [ScopePPGetCOMPort](#) disponibilizada nessa versão do Scope.

Fontes de Dados do SCOPE no WoW64

No WoW64, as fontes de dados (ou DSNs) são criadas por um Administrador de Fontes de Dados ODBC compatível com 32 bits do WoW64 e não o de 64 bits que é acessível por padrão a partir do Painel de Controle do Windows. Assim, ao instalar o Scope são criados links com as bases de dados que podem depois ser alterados se necessário em **%SystemRoot%\SysWOW64\odbcad32.exe**.

Para maiores informações sobre o Wow64, consulte a sessão "**Instalação do SCOPE em Sistemas Operacionais Microsoft Windows® de 64-Bits**", no documento "NCR Manual de Instalação Scope"

Carregamento Dinâmico da biblioteca

A partir da versão do Scope **3.01.23.138**, é possível realizar o carregamento dinâmico da biblioteca principal do Scope Client, o módulo scopeapi.dll. Além disso, é possível colocar o arquivo de configuração scope.ini na mesma pasta onde se localiza a DLL. Para isso, é preciso satisfazer as seguintes condições:

1. É permitido realizar somente no sistema operacional Windows.
2. A aplicação precisa indicar o caminho desejado da localização dos módulos através da API Windows **SetDllDirectory** antes de chamar a função **LoadLibrary**.

Exemplo em C/C++:

```
{  
    char szDLLPath[256] = { 0 };  
    HMODULE hModule;  
  
    // Atribuir em szDLLPath a localização da pasta onde foram instalados os módulos do  
    // Scope Client.  
  
    ...  
  
    SetDllDirectory(szDLLPath);  
  
    hModule = LoadLibrary(TEXT("scopeapi.dll"));  
  
    ...  
}
```

Funções básicas da API do SCOPE Client

Comunicação com o ScopeSRV

O funcionamento correto da maior parte das funções do SCOPE Client demanda a conexão com o ScopeSRV. Esta conexão possui a finalidade de carregar inicialmente todos os parâmetros de configuração da aplicação, para que o SCOPE Client consiga realizar as transações básicas. Toda mensagem trocada entre o SCOPE Client e o ScopeSRV deve utilizar o protocolo TCP/IP.

Conexão

A função ScopeOpen abre comunicação com o Scope Server e inicia o SCOPE Client, procedendo a alocação dos recursos necessários.

Recomenda-se que a conexão com o Scope Server seja realizada sob demanda, ou seja, apenas quando a automação for realizar a(s) TEF(s) para o atendimento de um determinado cliente.

Logo após realizar as TEF para o atendimento desse cliente, a comunicação com o Scope Server deve ser fechada, através da função ScopeClose, que será explicada a seguir.

Antes de enviar o logon para o SCOPE Server, o client gera um par de chaves pública e privada. Dependendo da configuração de hardware do PDV, esse processo pode levar alguns segundos. Isso é necessário para gerar uma chave de tamanho seguro e assegurar a conformidade com o PCI-DSS. Entretanto, o estabelecimento pode definir um tempo máximo de para gerar essas chaves, configurando o parâmetro GenTimeout no arquivo scope.ini do client (ver tópico Sessão [*<empresa><filial>*]).

ATENÇÃO: a configuração do parâmetro GenTimeout poderá acarretar em não conformidade com o PCI-DSS.

Protótipo

```
LONG EXPORT ScopeOpen (char *Modo, char *Empresa, char *Filial, char *PDV)
```

Parâmetros

[in]	String (constante igual à "2")	Modo	Modo de operação
[in]	String com quatro dígitos	Empresa	Código de identificação da empresa conforme cadastrado no ScopeCNF
[in]	String com quatro dígitos	Filial	Código de identificação da filial conforme cadastrado no ScopeCNF
[in]	String com três dígitos	PDV	Número do PDV conforme cadastrado no ScopeCNF

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFE02	65026	SCOPE API já foi inicializada corretamente
0xFE09	65033	Servidor não configurado no arquivo scope.ini
0xFF45	65349	Verificar o erro retornado no log do ScopeSrv

OxFF96	65430	O ScopeSrv retornou um erro específico indicando que não conseguiu montar a lista de prioridade/aid - Verifique a configuração do perfil do pdv
--------	-------	--

Exemplo

```

...
char empresa[] = "0001";
char filial[] = "0001";
char pdv[] = "0001";
...
retorno = ScopeOpen ("2", empresa, filial, pdv);
if (retorno != 0)
{
    printf("Erro ao conectar com o SCOPE Server. Erro(%d)", retorno);
    exit(0);
}
...

```

A função ScopeOpenVerify surgiu para suprir a necessidade de automações comerciais que desejam verificar se o scope.ini foi alterado pelo estabelecimento ao longo do tempo. A automação comercial fica responsável por gerar o hash do arquivo e armazená-lo, o hash será passado para a função ScopeOpenVerify, o SCOPE irá gerar o hash do scope.ini no momento que a função foi chamada. A função irá comparar o hashes, caso estejam diferentes, irá retornar erro e não se conectará ao ScopeSRV.

Protótipo

```
LONG EXPORT ScopeOpenVerify (char *Empresa, char *Filial, char *PDV, char *hashScopeini)
```

Parâmetros

[in]	String com quatro dígitos	Empresa	Código de identificação da empresa conforme cadastrado no ScopeCNF
[in]	String com quatro dígitos	Filial	Código de identificação da filial conforme cadastrado no ScopeCNF
[in]	String com três dígitos	PDV	Número do PDV conforme cadastrado no ScopeCNF
[in]	String com 20 dígitos	hashScopeini	Hash do scope.ini gerado com algoritmo sha1.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno			Significado
Hexadecim al	Decimal		
0xFA01	64001	Parâmetro 1 inválido	
0xFA02	64002	Parâmetro 2 inválido	
0xFA03	64003	Parâmetro 3 inválido	
0xFA04	64004	Parâmetro 4 inválido	
0xFE02	65026	SCOPE API já foi inicializada corretamente	

Exemplo

```

...
char empresa[] = "0001";
char filial[] = "0001";
char pdv[] = "0001";
// calcula hash do scope.ini
...
retorno = ScopeOpenVerify (empresa, filial, PDV, "EE44A73531FB35BF7009");
if (retorno != 0)
{
    printf("Erro ao conectar com o SCOPE Server. Erro(%d)", retorno);
    exit(0);
}
...

```

Desconexão

A função ScopeClose encerra a comunicação com o Scope Server, procedendo à liberação de todos os recursos alocados.

Como explicado no tópico anterior relacionado à função ScopeOpen, essa função deve ser chamada logo após realizar a(s) TEF(s) relacionadas ao atendimento de um determinado cliente. Desta forma, a conexão com o Scope Server se encerra, evitando ociosidade nessa comunicação.

Protótipo

LONG EXPORT ScopeClose (void)

Parâmetros

Não há parâmetro.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno			Significado
Hexadecim al	Decimal		
OxFE01	65025	SCOPE API não foi inicializada corretamente	
OxFE00	65024	A transação em andamento – a aplicação deve aguardar	

Exemplo

```

...
retorno = ScopeClose ();
...

```

Sessão de transação

Sessão de TEF

O conceito de sessão de TEF existe para garantir a integridade de uma operação. Desta forma, entre a abertura e encerramento da sessão, todas as atividades incluídas na respectiva operação (autorização da transação, impressão de comprovante) deverão ser completamente realizadas. Este mecanismo permite que em caso de falha (queda de energia), a operação possa ser desfeita. Isso corresponde a imprimir corretamente o cupom de TEF e garantir que a falha ocorrida não atrapalhe no funcionamento da aplicação.

Sendo multi-TEF, o SCOPE implementa o conceito de sessão de TEF com o intuito de garantir que todas as transações numa mesma sessão serão aprovadas ou desfeitas. Com isto, o cliente pode efetuar o pagamento parcialmente de diversas maneiras (exemplo: 40% do valor da venda serão pagos em débito e o restante com o cartão de crédito), e para validar a venda, todas as transações deverão ser aprovadas, caso contrário, as transações não podem ser concluídas (não tem sentido que a venda seja considerada concluída apenas com a aprovação de parte do valor da compra).

Em cada venda realizada, a aplicação deve abrir a sessão, realizar as diversas transações e finalmente fechar a sessão, confirmando ou desfazendo todas as transações desta venda. A estrutura do fluxo básico de funcionamento do SCOPE Client encontra-se na Figura 4.

IMPORTANTE: numa sessão de multi-TEF em que há várias transações, não é possível desfazer uma ou outra transação. Para isso, o operador deverá cancelar todas daquela sessão ou confirmá-las.

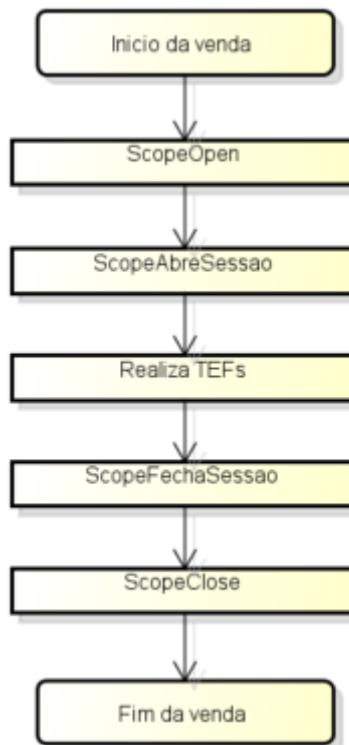


Figura 4: visão geral do funcionamento do SCOPE Client.

Abrindo uma sessão

A função ScopeAbreSessaoTEF() informa ao SCOPE para iniciar uma sessão de TEF (ciclo com uma ou mais transações TEF). Ela deve ser invocada ao finalizar a venda e antes de selecionar o meio de pagamento.

Protótipo

```
LONG EXPORT ScopeAbreSessaoTEF (void)
```

Parâmetros

Não há parâmetro.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE00	65024	A transação em andamento – a aplicação deve aguardar

Exemplo

```
...
// Conecta ao servidor do SCOPE
...
// Realiza a venda
...
retorno = ScopeAbreSessaoTEF();
if (retorno != 0)
{
    // Trata o erro e interrompe a sequência.
}
else
{
    // Inicia a(s) transação(s) desejada(s)
}
```

Encerrando a sessão

Aciona o SCOPE para finalizar uma sessão de TEF (ciclo com uma ou mais transações de TEF), ou seja, confirmar ou desfazer as transações da sessão em aberto, após encerrar o processamento da transação.

Protótipo

```
LONG EXPORT ScopeFechaSessaoTEF(BYTE Acao, BYTE *DesfezTEFAposQueda)
```

Parâmetros

[in]	Byte (0:Desfaz; 1:Confirma)	Acao	Informa o SCOPE para confirmar ou desfazer a(s) transação(s) da sessão de TEF atual
[out]	Ponteiro para byte	DesfezTEFAposQueda	Retorna se a(s) transação(s) da sessão de TEF foram desfeitas após uma queda de energia.

Retorno

Caso retorne sucesso (0x0000), significa que o SCOPE conseguiu com êxito confirmar ou desfazer a(s) transação(s) de uma sessão de TEF. Caso contrário, ocorreu algum problema na confirmação ou desfazimento da(s) transação(s). Para maiores detalhes dos códigos relacionados, ver [tabela de códigos de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFB08	64264	Erro no arquivo de controle utilizado finalização no ciclo multi-TEF
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente

Exemplo

```
BYTE acao, desfez;
...
// Realiza a(s) transação(s) desejada(s)
...
acao = 1; // confirmar
retorno = ScopeFechaSessaoTEF(acao, &desfez);
if (retorno != 0)
{
    // Erro no encerramento da sessão
}
...

```

Tratando queda de energia

A função que encerra a sessão também é utilizada para desfazer as transações pendentes de uma sessão de TEF após uma queda de energia no PDV. Se o parâmetro que permite que a aplicação interfira na conclusão de transações em caso de quedas de energia estiver habilitado (ver tabela de configurações gerais), é realizado o desfazimento ou a confirmação das transações pendentes, de acordo com o parâmetro de entrada escolhido pelo operador. Se o parâmetro estiver desabilitado, após a aplicação ter restabelecido a conexão com o servidor SCOPE, ela deverá chamar essa função para finalizar uma possível sessão que ainda esteja em aberto. Nesse caso, é realizado o desfazimento das transações pendentes, independentemente, do parâmetro de entrada.

Um detalhe importante é que uma sessão de TEF é considerada finalizada ao iniciar a execução dessa função. Desta forma, mesmo que ocorra queda de energia durante a execução dessa função, o SCOPE considera como finalizado a sessão de TEF e procederá para realizar a ação solicitada.

O segundo parâmetro faz o papel principal no tratamento em queda de energia. Este parâmetro é independente do primeiro parâmetro e somente tem funcionalidade nesta situação. Caso este parâmetro seja devolvido com o valor 1 (True), a aplicação deve exibir a mensagem **"A transação TEF anterior foi desfeita (cancelada). Reter o cupom TEF"**, e se possível, acionar um supervisor para verificar a situação.

Exemplo

```

...
BYTE acao, desfez;
...
// Conecta ao servidor do SCOPE
...
ScopeFechaSessaoTEF(acao, &desfez);
if (desfez)
{
    printf("A transação TEF anterior foi desfeita (cancelada).");
    printf("\n Reter o cupom TEF.");
}
...

```

IMPORTANTE: sempre que tratar uma possível queda de energia e o segundo parâmetro da função ScopeFechaSessaoTEF() retornar o valor 1, deve-se imprimir a mensagem:

"A transação TEF anterior foi desfeita (cancelada). Reter o cupom TEF"

Deixando transação pendente na queda de energia

É possível também deixar a transação pendente para depois realizar o acerto de pendência manual no módulo de pendência SCOPENPND. Este acerto só deverá ser realizado por um gerente, alguém de finanças ou quem a empresa deposite confiança e seja responsável para tal procedimento.

O aplicativo deve chamar a função ScopeMTEFOnOff(), antes do tratamento de queda de energia, informando se deseja desfazer ou deixar pendentes as transações.

Protótipo

LONG EXPORT ScopeMTEFOnOff (BYTE Trata)

Parâmetros

[in]	Byte	Trata	Informa o SCOPE para desfazer (0) possíveis transações interrompidas ou deixar pendente (1) para o posterior acerto manual.
------	------	-------	---

Retorno

- 0x0000 – definiu com êxito a ação a ser tomada conforme o parâmetro passado.

- 0xFE00 – não foi aberta a conexão com o SCOPE Server.
- 0xFE01 – tem transação em sendo processado.

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente

Exemplo

```

...
BYTE acao, defez;
...
// Conecta ao servidor do SCOPE
...
ScopeMTEFOnOff(1); // deixa pendente
ScopeFechaSessaoTEF(acao, &defez);
if (defez)
{
    printf("A transação TEF anterior foi desfeita (cancelada).");
    printf("\n Reter o cupom TEF.");
}
...

```

Status de transação

Durante o processamento da transação, a aplicação pode consultar o estado em que a transação está e tomar alguma ação baseado no valor dela. Esta ação pode estar entre uma das situações abaixo, conforme o intervalo de códigos retornado pelo SCOPE Client:

- valor igual 0: transação finalizada com sucesso;
- valores entre 3 (3h) e 92 (5Bh): erro de algum parâmetro passado para o SCOPE;
- valores entre 64001 (FA01h) e 64005 (FA05h): erro de algum parâmetro passado para o SCOPE;
- valores entre 64257 (FB01h) e 64267 (FB0Bh): erro interno do SCOPE;
- valores entre 64512 (FC00h) e 64767 (FCFFh): coleta de algum dado necessário à transação;
- valores entre 65025 (FE01h) e 65535 (FFFFh): erro reportado pelo SCOPE referente à transação;

O código de status 65024 (FE00_h) indica que a transação está em andamento e a aplicação deve aguardar. Ou seja, a aplicação deverá aguardar algum tempo para solicitar novamente o status até que um código de status diferente de 65024 seja fornecido.

Consultando o status

A consulta do status da transação deve ocorrer enquanto uma transação (cartão de crédito, cartão de débito, recarga de celular, etc.) está em processamento.

Protótipo

```
LONG EXPORT ScopeStatus (void)
```

Parâmetros

Não há parâmetro.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimai	
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente

Exemplo

```
...
// Abre sessão
retorno = ScopeCompraCartaoCredito(valor, taxa_servico);
if (retorno == 0)
{
    do
    {
        retorno = ScopeStatus();
        // trata conforme retorno
        if (retorno == 65024)
        {
            sleep (1000);
            continue;
        }
        else
            if ((retorno <= 0xFC00) || (retorno >= 0xFF))
                processando == 0; //saiu da faixa de coleta
                // trata a coleta de algum dado
    } while (processando == 1);
```

```

    } // fim do if
}
{
    // Trata erro retornado na solicitação da transação
}
...

```

Funções de configuração de ambiente

O SCOPE Client carrega diversos parâmetros de configuração para seu funcionamento no ambiente de PDV em que está executando. Estes dados são carregados na conexão com o ScopeSRV e na leitura do arquivo scope.ini.

Além destas duas fontes de configuração, há também aquela que a aplicação pode fazer por meio de algumas funções disponíveis nas bibliotecas do SCOPE Client. Dependendo do tipo de configuração a que nos referimos, ela permite a aplicação ativar ou desativar as configurações durante a execução do programa.

Configurações gerais

No momento de execução do aplicativo, a aplicação de frente de loja poderá configurar os seguintes itens:

Código de identificação	Descrição
1	Permite que a aplicação cancele a transação que está sendo executada no PIN-Pad quando este esteja coletando algo. Por padrão, a aplicação não tem conhecimento do momento em que o SCOPE está coletando o cartão, a senha ou outra informação no PIN-Pad, pois ele retorna o estado indicando processamento (65024), o qual não é um estado de coleta, e, portanto, inviabilizando o uso da função ScopeResumeParm() para o possível cancelamento.
2	Habilitado este item, o SCOPE retornará o estado para obter os serviços (64644) durante o fluxo de TEF para que a aplicação de PDV possa obter os serviços e parâmetros configurados no ScopeCNF.
4	Se habilitado, não abrirá o digitado na leitura do cartão com o PIN-Pad Compartilhado. Por padrão, ao pressionar a tecla cancela no PIN-Pad, o SCOPE Client retornará um estado para coletar o PAN do cartão.
8	Normalmente, ao solicitar a digitação da senha no PIN-Pad pela função ScopePPStartGetPIN(), a função ScopePPGetPIN() retorna para a aplicação a senha digitada no PIN-Pad descriptografada. Habilitando este item, a senha retornada estará criptografada pela MasterKey da Itautec.
16	PDVs com impressora com papel de carbono podem habilitar este item para não imprimir a informação de 1ª Via Cliente ou 2ª Via Estabelecimento nos respectivos cupons.
32	Permite a gravação em arquivo dos dados da coleta para ser recuperado em caso de queda de energia.

128	Permite que a aplicação interfira na conclusão de transações que não foram finalizadas apropriadamente em caso de quedas de energia, confirmando ou desfazendo as transações de acordo com a decisão do operador.
256	Se habilitado, permitirá a coleta de saque em operações de débito à vista. Atualmente só é usado pela rede autorizadora Cielo.

IMPORTANTE: este tipo de configuração só é permitido quando a aplicação está utilizando a interface coleta.

Configurando em tempo de execução

A função ScopeConfigura(), responsável para a configuração da aplicação de PDV citada acima, poderá ser executada apenas uma única vez após a inicialização do SCOPE ou antes das operações TEF.

Protótipo

```
LONG EXPORT ScopeConfigura (LONG Id, LONG Param)
```

Parâmetros

[in]	LONG	Id	Identifica o parâmetro a ser configurado conforme tabela acima
[in]	LONG	Param	Informa se deve habilitar ou desabilitar o item. Para habilitar a aplicação deverá passar o valor 1 e para desabilitar, o valor 0.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFF59	65369	Função de uso exclusivo na interface coleta

Exemplo

```
...
// conexão com o Scope server bem sucedida
...
// trabalhando com impressora carbonada
retorno = ScopeConfigura(16, 1);
...
...
```

Configuração de PIN-Pad

Para evitar a substituição de PIN-Pad por um tipo incompatível com a configuração do ScopePSW, há a possibilidade de informar o SCOPE o tipo do PIN-Pad que está conectado e proibir a execução de transação. A validação está baseada nas seguintes opções:

- valor 0: não possui PIN-Pad conectado;
- valor 1: utilizando PIN-Pad via biblioteca VISANET;
- valor 2: utilizando PIN-Pad com biblioteca compartilhada.

Validando a interface de PIN-Pad

A validação da interface utilizada pela aplicação de PDV versus a configurada no ScopePSW somente ocorrerá na chamada à função ScopeOpen(). Sendo assim, a aplicação de PDV deve solicitar a validação antes da conexão com o servidor do SCOPE.

Protótipo

```
LONG EXPORT ScopeValidalInterfacePP (BYTE IntPP)
```

Parâmetros

[in]	BYTE	IntPP	informa ao SCOPE a interface de acesso ao PIN-Pad compartilhada utilizada pela Aplicação PDV.
------	------	-------	---

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar

Exemplo

```
...
// PIN-Pad compartilhado conectado
retorno = ScopeValidalInterfacePP(2);
// conecta ao ScopeSRV
...
...
```

LEMBRETE: a chamada a essa função deve ser feita antes de chamar a função [ScopeOpen\(\)](#).

Funções específicas das interfaces

Dependendo da interface sobre a qual a aplicação de PDV foi desenvolvida, a aplicação deverá tratar funções específicas da interface escolhida.

Interface coleta

Como a aplicação será responsável pela coleta (entrada de dados do usuário), esta deve obter do SCOPE Client informações para solicitar os dados. Da mesma maneira, a aplicação deve devolver o que foi coletado para o SCOPE Client. Antes de tudo, a aplicação necessita de um meio para informar ao SCOPE Client que o modo de interação será pela interface coleta. O fluxo da aplicação, quando esta utiliza a interface coleta, está esboçado no diagrama da Figura 5Figura 4. Nos tópicos subsequentes explicamos os mecanismos para realizar estas operações.

Na interface coleta, o SCOPE Client retornará, através da função [ScopeStatus\(\)](#), um código (ver códigos na tabela [Coleta de dados](#)) sempre que necessitar de uma atuação do aplicativo, como por exemplo, coletar dados, imprimir cupons e cheques, mostrar mensagens para o operador e/ou cliente, entre outros.

Definindo a interface coleta

Antes de realizar qualquer transação, a aplicação deve definir a interface coleta como meio de interação, o que é realizado pela função ScopeSetApiColeta(). Isto deverá ser feito após a conexão do SCOPE Client com o servidor.

Protótipo

```
LONG EXPORT ScopeSetApiColeta(void)
```

Parâmetros

Não há parâmetro.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Hexadecim al	Decimal	Significado
0xFE01	65025		SCOPE API não foi inicializada corretamente

Exemplo

```
...
// Conexão ao ScopeSRV bem sucedida
...
retorno = ScopeSetApiColeta();
...
```

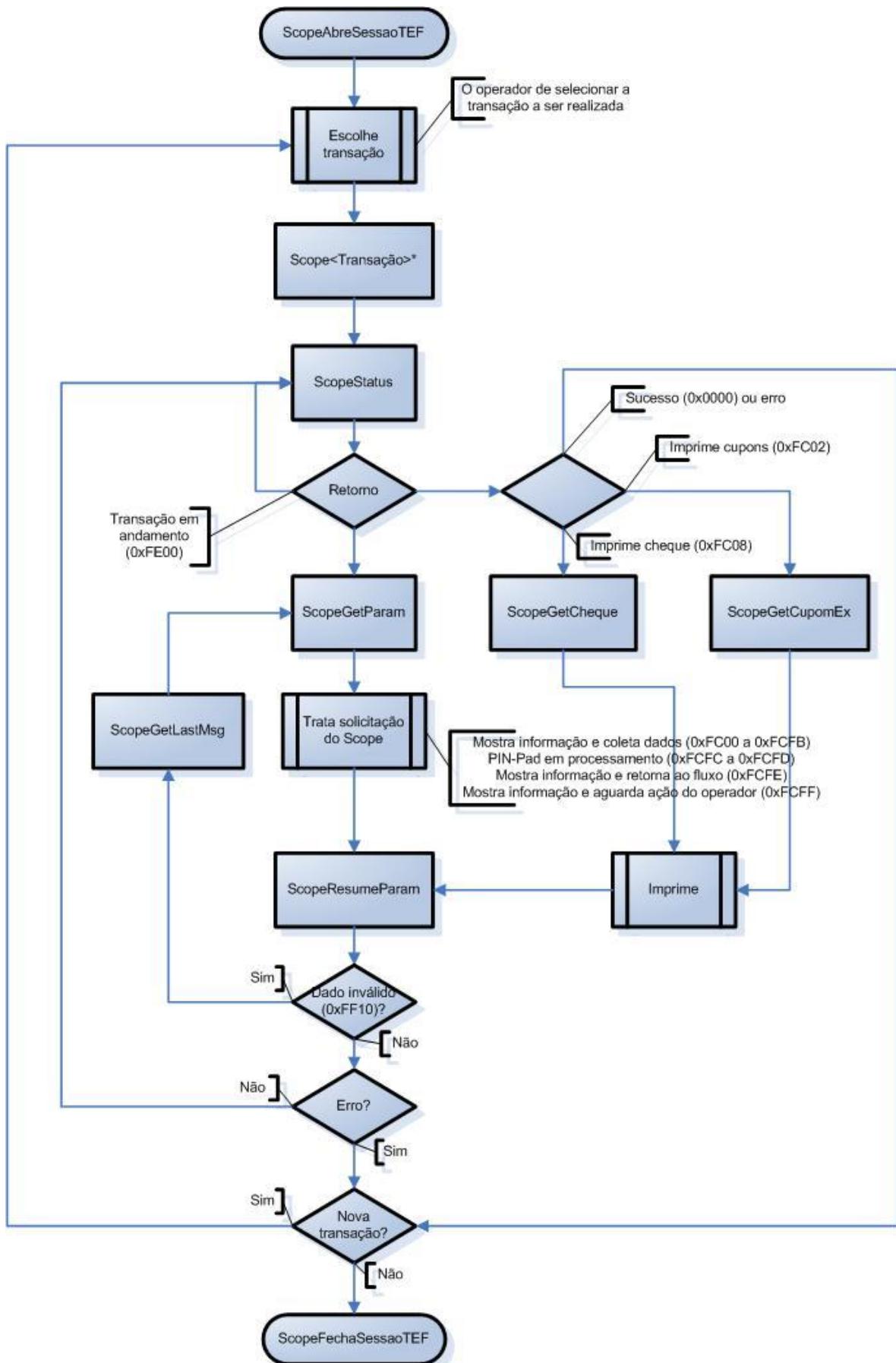


Figura 5: fluxo de processamento de uma transação com a interface coleta.

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Obtendo os parâmetros da transação

Durante a iteração do processamento de uma transação, a aplicação deve obter informações necessárias para solicitar a coleta do dado para o operador e/ou cliente. A função [ScopeGetParam\(\)](#), responsável em descrever os dados a coletar, deve ser chamada sempre que o SCOPE Client retornar um estado de coleta correspondente a uma coleta de dados (ver [Status de transação](#)). Com o retorno do SCOPE Client, a aplicação de PDV deverá atualizar as mensagens das telas do operador e/ou cliente, disponibilizar um meio de entrada de dado, validar a entrada, entre outras ações. A aplicação deve prover o operador de algum meio que ele possa cancelar, retornar e continuar a coleta de dados da transação.

As mensagens que o SCOPE Client disponibiliza para serem exibidas são fundamentadas no fato de que no PDV haja dois visores (display): um que é apresentado ao cliente da loja e outro, apresentado ao operador do PDV. Além disso, por padrão, considera-se que cada um desses visores possua 2 linhas de exibição de mensagens com largura de 40 caracteres. É possível configurar o tamanho máximo de colunas para até 20 ou 16 (ver o capítulo [Configuração](#)).

Estruturas de apoio

Para a aplicação receber do SCOPE Client as informações para coleta, ela deverá passar um endereço de memória de um buffer que é representado, em linguagem C, pela estrutura ptPARAM_COLETA definida no arquivo `ScopeApi.h`, cuja declaração é:

```
typedef struct _stPARAM_COLETA
{
    WORD Bandeira;
    WORD FormatoDado;
    WORD HabTeclas;
    char MsgOp1[64];
    char MsgOp2[64];
    char MsgCl1[64];
    char MsgCl2[64];
    char WrkKey[16+1];
#ifndef __linux__
    char filler;
#endif
    WORD PosMasterKey;
    char PAN[19+1];
    BYTE UsaCriptoPinpad;
    BYTE IdModoPagto;
    BYTE AceitaCartaoDigitado;
    char Reservado[105];
} stPARAM_COLETA, *ptPARAM_COLETA;
```

A descrição de cada campo encontra-se abaixo:

- Bandeira: a bandeira do cartão relativa à operação em andamento (ver Código das bandeiras);
- FormatoDado: código respectivo ao formato do dado a ser coletado (ver tabela de códigos em Formato dos dados);
- HabTeclas: teclas que estão disponíveis para a aplicação. É um campo com combinação de bits. Exemplo: se o campo estiver com o valor 6 ($2 + 4 = 6$), significa que as teclas para prosseguir (valor 2) e retornar (valor 4) a coleta devem estar habilitadas, desabilitando a tecla cancelar (valor 1) (ver códigos relacionados na tabela Código das teclas);
- MsgOp1: mensagem para exibição na linha 1 do visor do operador (string);
- MsgOp2: mensagem para exibição na linha 2 do visor do operador (string);

- MsgCl1: mensagem para exibição na linha 1 do visor do cliente (string);
- MsgCl2: mensagem para exibição na linha 2 do visor do cliente (string);
- WrkKey: chave de trabalho utilizada pela função PP_iGetPIN() que é responsável pela coleta de senha criptografada no PIN-Pad (string). Este campo é utilizado apenas com a biblioteca da VISANET para PIN-Pad;
- filler: para alinhamento em 8 bytes nos ambientes executados sobre o sistema operacional Linux;
- PosMasterKey: posição da Master Key utilizada pela função PP_iGetPIN(). Este campo é utilizado apenas com a biblioteca da VISANET para PIN-Pad;
- PAN: número do cartão utilizado pela função PP_iGetPIN(). Este campo é utilizado apenas com a biblioteca da VISANET para PIN-Pad;

UsaCriptoPinpad: indica se a aplicação deverá coletar senha com criptografia no PIN-Pad (função PP_iGetPIN()) ou sem criptografia no PIN-Pad (função PP_iGetString()). Para o valor 1 (um), deve coletar com criptografia no PIN-Pad e para o valor 0 (zero), sem criptografia. Este campo é utilizado apenas com a biblioteca da VISANET para PIN-Pad;

- IdModoPagto: modalidade de pagamento (crédito, débito ou outros). Parâmetro utilizado na chamada da função PP_iGetCard(). Este campo é utilizado apenas com a biblioteca da VISANET para PIN-Pad;
- AceitaCartaoDigitado: indica se aceita a entrada digitada do número do cartão. Informação disponível no estado de coleta do cartão;
- Reservado: utilização futura.

Protótipo

```
LONG EXPORT ScopeGetParam (LONG tipoParam, ptPARAM_COLETA lpParam)
```

Parâmetros

[in]	LONG	TipoParam	Aplicação informa que tipo de parâmetro (retornado pelo ScopeStatus()) deseja-se obter, isto é, os parâmetros para uma coleta de dados e/ou exibição de uma mensagem.
[out]	ptPARAM_COLETA	lpParam	Conjunto de dados para a coleta detalhado acima

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE01	65025	SCOPE API não foi inicializada corretamente

Exemplo

```
...
// executa transação (crédito, débito, etc.)
stPARAM_COLETA Pcoleta;
```

```

...
// inicia iteração
retorno = ScopeStatus();
if(retorno != 0xFE00)
{
    ...
    memset(&PColeta, '\0', sizeof(PColeta));
    retorno = ScopeGetParam(retorno, &PColeta);
    // atualiza as teclas retornar, avançar e cancelar
    // coleta os dados
    ...
    // entrega o que coletou ao SCOPE
    ...
}
// finaliza iteração

```

Notas: A partir da versão 3.01.16.001 do SCOPE foi criada a possibilidade de “Coleta Especial”, conforme descrito abaixo. Tal recurso foi implementado para viabilizar as “coletas dinâmicas” do cartão Frota da rede Ticket Log.

A partir da versão (3.01.18.004), foi implementada a “Coleta Estendida”, identificada pelo estado de coleta TC_COLETA_EXT (0xFCFB). Isso foi necessário devido limite do range 0xFC00 a 0xFCFF.

Coleta Especial

Caso o estado retornado pelo SCOPE seja **TC_COLETA_DADO_ESPECIAL**, a aplicação deverá usar a função estendida ScopeGetParamExt, cujo formato está descrito abaixo.

Coleta Estendida

Caso o estado retornado pelo SCOPE seja **TC_COLETA_EXT**, a aplicação deverá usar a função estendida ScopeGetParamExt, cujo formato está descrito abaixo. Neste caso, o campo **IdColetaExt** pode ser usado para identificar a coleta estendida, conforme tabela “Estados de Coleta Estendidos” do Apêndice A – Tabelas.

Protótipo

```
LONG EXPORT ScopeGetParamExt (LONG tipoParam, ptPARAM_COLETA_EXT lpParam)
```

Parâmetros

[in]	LONG	TipoParam	Aplicação informa que tipo de parâmetro (retornado pelo ScopeStatus()) deseja-se obter, que no caso é 0xFCF4.
[out]	ptPARAM_COLETA_EXT	lpParam	Conjunto de dados para a coleta detalhado abaixo.

Retornos

Ver [tabela de código de retorno](#).

Exemplo

```

...
// executa transação (crédito, débito, etc.)
stPARAM_COLETA_EXT PcoletaExt;

...
// inicia iteração
retorno = ScopeStatus();
if(retorno = 0xFEF4)
{
    ...
    memset(&PColeta, '\0', sizeof(PColeta));
    retorno = ScopeGetParamExt(retorno, &PColetaExt);
    // atualiza as teclas retornar, avançar e cancelar
    // coleta os dados
    ...
    // entrega o que coletou ao SCOPE
    ...
}
// finaliza iteração

```

Estrutura do Parâmetro

```

typedef struct _stDADOS_EXT
{
    char Sigla[3];      // Identificação da coleta extendida, se existir
    char Rotulo[40 + 1]; // Rotulo a ser exibido
    BYTE AceitaVazio;  // Aceita ou não dado vazio
    BYTE QtdCasasDecimais; // qtd de casas decimais, se pertinente
    char TamMin[2];     // Tamanho mínimo do campo
    char TamMax[2];     // Tamanho máximo do campo
} stDADOS_EXT, *ptDADOS_EXT;

typedef struct _stDADOS_LIM
{
    char Inferior[12]; // Limite inferior (uso futuro)
    char Superior[12]; // Limite superior (uso futuro)
} stDADOS_LIM, *ptDADOS_LIM;

typedef struct _stPARAM_COLETA_EXT
{
    BYTE FormatoDados;
    BYTE HabTeclas;
    char CodBandeira[3];
    char CodRede[3];
    char MsgOp1[64];
    char MsgOp2[64];
    char MsgCl1[64];
    char MsgCl2[64];
    BYTE UsaExt;        // Indica se deve ou não usar os campos sExt abaixo
    stDADOS_EXT sExt;   // Dados extendidos
    BYTE UsaLimites;    // Indica se deve ou não usar os campos sLimite abaixo
}

```

```

stDADOS_LIM sLimite; // Limites inferior e superior: uso futuro
WORD   IdColetaExt; // Identificação da coleta estendida
char    Reservado[97];
} stPARAM_COLETA_EXT, *ptPARAM_COLETA_EXT;

```

Formatos de Dado

```

typedef enum {
    TM_DDMMAA,           // Data no formato DDMMAA
    TM_DDMM,             // Data no formato DDMM
    TM_MMAA,             // Data no formato MMAA
    TM_HHMMSS,           // Hora no formato HHMMAA
    TM_NUM,               // Número inteiro
    TM_SENHA,             // Senha (interno ao SCOPE)
    TM_ULTIMOS_DIGITOS, // Últimos dígitos
    TM_ALFANUMERICO,     // Alfanumérico
    TM_DDMMAAAA,          // Data no formato DDMMAAAA
    TM_CONFIRMACAO,       // Display para exibição (não há coleta)
    TM_MMAAAA,             // Data no formato MMAAAA
    TM_MASCARADO,          // Exibir '*' na tela, mas enviar em claro
    TM_HHMM,               // Hora no formato HHMM
    TM_BOOL,                // Booleano, a resposta deve ser 0=Não ou 1=Sim
    TM_VALOR_MONETARIO,    // Valor monetário, de tam=10+2 casas decimais, total=12
    TM_NUM_DECIMAL,         // Número não inteiro (com casas decimais)
    TM_SELECAO,              // Seleção de opção (Menu)
    TM_PAN                  // PAN do cartão
} TIPO_MASCARA;

```

Formato de Dado para “Seleção”

Caso o formato seja TM_SELECAO, deve-se usar as funções de obtenção de Menu Dinâmico do SCOPE, ScopeMenuRecuperaltense ScopeMenuSelecionalItem. O tipo de tabela a ser usado é o MNU_TAB_TIPO_SELECAO_ESPECIAL = 5.

Passando o dado da coleta ao SCOPE Client

Realizando a coleta de dados solicitada pelo SCOPE via função [ScopeStatus\(\)](#), a aplicação devolverá o que foi coletado e alguma ação ao SCOPE. A ação se refere à interrupção do processamento, retorno para um estado anterior ou continuação para o próximo estado de coleta. Os dados coletados são entregues ao SCOPE por meio da função [ScopeResumeParam\(\)](#).

Protótipo

```
LONG EXPORT ScopeResumeParam (LONG codTipoColeta, char *dados, WORD dadosParam, eACAO_APL acao)
```

Parâmetros

[in]	LONG	codTipoColeta	Código do estado, obtido pela função ScopeStatus() , para qual o dado foi coletado
[in]	String	dados	Dado coletado pela aplicação
[in]	WORD	dadosParam	Modo de captura do dado. Informa se a captura foi feita pelo teclado (0x0004) ou pela leitura magnética (0x0020) ou pela interface que suporta tanto a leitura magnética como a leitura do Chip (0x0080) ou pela leitura de CMC7 (0x0010)
[in]	eACAO_APL	Ação	Ação tomada pelo operador (ver Códigos de Fluxo)

Retorno

Ver [tabela de código de retorno](#).

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE04	65028	Não existe transação suspensa
0xFF10	65296	Dado inválido

Exemplo

```

...
// executa transação (crédito, débito, etc.)
char *dados[128];
WORD modo_entrada;
...

// inicia iteração
cod_coleta = ScopeStatus();
if(cod_coleta != 0xFE00)
{
    ...
    // obtém os parâmetros para a coleta
    // atualiza as teclas retornar, avançar e cancelar
    // coleta os dados e armazena em na variável 'dados'
    modo_entrada = 0x0004; // pelo teclado
    // obtém a ação que o usuário escolheu e coloca em 'acao'
    ...
    // entrega o que coletou ao SCOPE na função abaixo
    ScopeResumeParam(cod_coleta, dados, modo_entrada, acao);
    ...
}
// finaliza iteração

```

Para resolver uma necessidade de alguns clientes que desejam manipular os itens do “Menu Dinâmico” montados através da Inicialização de Tabelas pelas redes VISANET-4.1 e CIELO, foram definidas duas novas funções exportadas pelo SCOPE Client, conforme abaixo. A ideia é que, ao receber o estado TC_EXIBE_MENU (que já existe hoje), o PDV possa opcionalmente chamar a função ScopeMenuRecuperaltens do SCOPE para receber os itens do Menu a serem exibidos e em seguida chamar a ScopeMenuSelecionaltem para indicar ao SCOPE qual item foi selecionado.

Desta forma, o PDV poderá, por exemplo:

- Exibir todos os itens na mesma tela (hoje, pelo fluxo do SCOPE Client, é apresentado um item por vez);
- Excluir algum item da lista que eventualmente não deva ser exibido ao operador;
- Alterar a descrição de algum item para melhor entendimento do operador;

- Fazer a seleção automática de um determinado item sem apresentá-lo ao operador;

OBSERVAÇÃO: O índice do item selecionado deve ser equivalente ao item da tabela fornecida pelo Scope, independente de ter sido exibido para o usuário ou não. Por exemplo, se o menu tem 10 itens, alguns deles não foram exibidos e o operador seleciona o último item do menu, o item a ser selecionado continua sendo o índice 10 (valor a ser passado pelo ScopeMenuSelecionaItem).

Protótipo

```
dllScopeAPI ScopeMenuRecuperaltens(BYTE _TipoTabela, char *_Buffer, WORD _TamBuffer);
```

Parâmetros

[in]	BYTE	_TipoTabela	Vide Tabela Abaixo;
[in]	String	_Buffer	Ponteiro para área alocada pela aplicação, para receber os itens do menu, formato abaixo.
[in]	WORD	_TamBuffer	Tamanho da área alocada disponível;

Parâmetro 1 – Tipo de Tabela

1	MNU_TAB_TIPO_DINAMICO	Menu Dinâmico da CIELO
2	MNU_TAB_TIPO_GENERICO	Menu Genérico para Transação POS
3	MNU_TAB_TIPO_SAVS	Menu de Itens da Plataforma de Serviço
4	MNU_TAB_TIPO_GENERICO_DINAMICO	Menu Genérico para Transação POS Dinâmico
5	MNU_TAB_TIPO_SELECAO_ESPECIAL	Menu de Seleção, usado quando ScopeGetParamExt retornar TM_SELECAO em FormatoDado da stPARAM_COLETA_EXT

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFF6B	65387	Não Encontrado (para quando não veio menu dinâmico na carga de tabelas).

Estruturas do Menu Dinâmico

```
Menu Dinâmico da CIELO (_TipoTabela = 1)
typedef struct {
    char CodFuncao[4+1];
    char Descricao[40+1];
    char CodGrupoServiço[2+1];
    char CodFluxoPDV[3+1];
    char CodRede[3+1];
    char CodBandeira[3+1];
    char CodFuncaoRede[4+1];
} stScopeItemMenuDinamico, *LPScopeItemMenuDinamico;
```

```

typedef struct {
    char   TipoTabela;
    char   QtdeItens;
    stScopeItemMenuDinamico sItem[15];
} stScopeMenuDinamico, *LPScopeMenuDinamico;

```

Seleção Especial da GetParamExt (_TipoTabela = 5)
#define QTD_MAX_ITENS_PERG_DIN_SEL 20
typedef struct
{
char Sigla[3]; // Sigla da seleção (informativo)
char Rotulo[40 + 1]; // Rótulo da seleção (para display)
} stPergDinSellItem, *LPPergDinSellItem;
typedef struct
{
char QtdeItens[2]; // Qtd de itens de seleção
stPergDinSellItem sItem[QTD_MAX_ITENS_PERG_DIN_SEL];
} stPergDinSel, *LPPergDinSel;

Protótipo

```
dllScopeAPI ScopeMenuSelecionaItem(char _Item);
```

Parâmetros

[in]	BYTE	_Item	Índice do item selecionado
------	------	-------	----------------------------

Retorno

Ver [tabela de código de retorno](#).

O uso destas novas funções é opcional. Caso não utilizadas o tratamento do Menu Dinâmico será da mesma forma sem a chamada (todos os itens exibidos).

As configurações relativas ao Menu Dinâmico existentes hoje no SCOPE.INI do ScopeSRV serão mantidas.

Caso seja necessário alterar algum item ou comportamento do menu, recomendamos antes entrar em contato com o representante da Cielo para solicitar aprovação das alterações e verificar a necessidade de recertificação do software de frente de loja.

TEF

Nas seções seguintes, trataremos sobre as transações com cartão crédito e débito, desde compras a saldo até as próprias consultas de saldo.

Cartão de crédito

Uma das operações mais realizadas numa loja e muito incentivada pelas empresas de cartões, o pagamento de compra com o cartão de crédito apresenta diversos serviços relacionados: compra à vista, parcelada pela loja, parcelada pela administradora de cartão, consulta de saldo e de financiamento e cancelamento (que dedicamos um capítulo exclusivo).

Compra com cartão de crédito

Embora haja diversos serviços relacionados ao cartão de crédito, apenas uma função está disponível: ScopeCompraCartaoCredito(). Ela se refere ao grupo de serviço de crédito e trata os seus serviços. O que determinará cada um destes serviços será o fluxo do SCOPE de acordo com o que este solicitar e o que o usuário devolver. Assim, a transação poderá terminar como à vista, parcelada pelo estabelecimento ou parcelada pela administradora.

Protótipo

```
LONG EXPORT ScopeCompraCartaoCredito (char *Valor, char *TxServico)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta.

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFC00	64512	Coletar cartão
OxFC01	64513	Coletar validade do cartão
OxFC02	64514	Imprime Cupom
OxFC03	64515	Coletar CGC ou CPF
OxFC09	64521	Coletar se a transação será à vista ou não
OxFC0E	64526	Coletar quantidade de parcelas
OxFC0A	64522	Coletar se a transação será parcelada pela administradora ou pelo estabelecimento
OxFC18	64536	Coleta os 4 últimos dígitos do cartão
OxFC1F	64543	Coletar quantidade de dias
OxFC24	64548	Coletar número do endereço
OxFC26	64550	Coletar plano de pagamento
OxFC28	64552	Coletar número do item (Fininvest)
OxFC34	64564	Coleta o valor da transação
OxFC53	64595	Coleta código do material
OxFC6D	64621	Coleta a data de emissão do cartão

OxFC6E	64622	Coleta o plano Infocards
OxFC6F	64623	Coleta número do cupom fiscal
OxFC0E	64526	Coletar quantidade de parcelas
OxFC15	64533	Coletar valor de entrada
OxFC1B	64539	Imprime consulta
OxFC20	64544	Coletar o número da pré-autorização
OxFC23	64547	Coletar CEP
OxFC33	64563	Coleta valor da taxa de serviço
OxFC64	64612	Coleta RG
OxFC7E	64638	Go On Chip
OxFC80	64640	Coleta o valor da taxa de embarque
OxFC84	64644	Obtém os serviços
OxFC85	64645	Coleta o cartão digitado
OxFC8A	64650	Coleta a data quando o cliente aderiu ao cartão
OxFC29	64553	Coletar código de segurança
OxFC2F	64559	Coleta código da localidade do telefone
OxFC30	64560	Coleta número do telefone
OxFC51	64593	Imprime cupom promocional
OxFCA5	64677	Coleta o número do Voucher
OxFCDF	64735	Coleta de Código para Identificação pagamento de Fatura (Exemplos de código: CPF, Código de Barras e Cartão)
OxFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
OxFCFF	64767	Mostrar Informações e aguardar confirmação do operador
OxFCF2	64754	Coleta se a transação deverá realizar uma consulta de planos de parcelamento
OxFC3D	64573	Coleta código de autorização
OxFC66	64614	Coleta CPF no teclado operador
OxFCF6	64758	Coleta CPF no PINPad
OxFCF9	64761	Coleta parcela grátis (1-Sim 0-Não)
OxFCF4	64756	Coleta dado especial

Possíveis Retornos de Erros

Códigos Retorno	Hexadecim al	Significado
Hexadecim al	Decimal	Significado
OxFA01	64001	Parâmetro 1 inválido
OxFA02	64002	Parâmetro 2 inválido
OxFB01	64257	Não foi possível criar a "thread" na coleta de dados
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um handle sem ter feito nenhuma transação desde última conexão com o ScopeSRV
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).

OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF0C	65292	Transação não implementada
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
OxFF3A	65338	Erro interno na execução da coleta
OxFF60	65376	Função indisponível
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
char valor[12 + 1];
char taxa[12 + 1];
...

// obtém o valor da compra e armazena em 'valor'
// obtém a taxa de serviço e armazena em 'taxa'
// abre sessão
...

retorno = ScopeCompraCartaoCredito(valor, taxa);
...

// processa a transação
...

// fecha a sessão
...

```

Consulta a financiamento de cartão de crédito

Antes de realizar uma compra financiada pela administradora de cartão de crédito, é possível consulta das parcelas do financiamento. A consulta seguirá os moldes de uma transação de compra, sendo que em nenhum momento será efetivada. Portanto, após a consulta, é necessário realizar a compra, que solicitará os mesmos dados da consulta.

Protótipo

```
LONG EXPORT ScopeConsultaCredito (char *Valor, char *TxServico)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFC84	64644	Obtém os serviços
0xFC18	64536	Coleta os 4 últimos dígitos do cartão
0xFC01	64513	Coletar validade do cartão
0xFC3D	64573	Coleta código de autorização
0xFC26	64550	Coletar plano de pagamento
0xFC29	64553	Coletar código de segurança
0xFC2A	64554	Coleta se código de segurança ausente ou ilegível
0xFC0E	64526	Coletar quantidade de parcelas
0xFC1F	64543	Coletar quantidade de dias
0xFC33	64563	Coleta valor da taxa de serviço
0xFC7E	64638	Go On Chip
0xFCFF	64767	Mostrar informações e aguardar confirmação do operador
0xFC50	64592	Coleta se a transação é com senha
0xFC11	64529	Coletar a senha
0xFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
0xFC1B	64539	Imprime consulta
0xFC02	64514	Imprime Cupom
0xFC7D	64637	Coleta se cancela ou não a transação
0xFCFD	64765	Coleta em andamento
0xFCFC	64764	Coleta cartão em andamento

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line

0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
char valor[12 + 1];
char taxa[12 + 1];
...

// obtém o valor da compra e armazena em 'valor'
// obtém a taxa de serviço e armazena em 'taxa'
// abre sessão

...
retorno = ScopeConsultaCredito(valor, taxa);
...

// processa a transação
...

// fecha a sessão
...

```

Consulta a saldo de cartão de crédito

Um dos motivos que uma transação pode não ser aprovada é a falta de saldo suficiente do cliente no cartão de crédito. Alguns cartões permitem esta consulta.

Protótipo

```
LONG EXPORT ScopeConsultaSaldoCredito (void)
```

Parâmetros

Não há parâmetros.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado

OxFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
*OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
// abre sessão
...
retorno = ScopeConsultaSaldoCredito();
...
// processa a transação
...
// fecha a sessão
...

```

Pré-autorização de crédito

Como a consulta de saldo do cartão de crédito não está disponível para todos os cartões e por representar certo risco (cartões roubados poderão ter todo o seu limite gasto, sendo conhecido o seu saldo), outras administradoras disponibilizam a opção de pré-autorização. Assim, antes de fazer a compra, é possível realizar uma pré-autorização para verificar se a compra procederá com sucesso.

Protótipo

```
LONG EXPORT ScopePreAutorizacaoCredito (char *Valor, char *TxServico)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido

OxFA02	64002	Parâmetro 2 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
char valor[12 + 1];
char taxa[12 + 1];
...
// obtém o valor da compra e armazena em 'valor'
// obtém a taxa de serviço e armazena em 'taxa'
// abre sessão
...
retorno = ScopePreAutorizacaoCredito(valor, taxa);
...
// processa a transação
...
// fecha a sessão
...

```

Captura de pré-autorização com cartão de crédito

Transação utilizada para realizar a captura (confirmação) de uma pré-autorização de crédito realizada anteriormente. A forma de utilização desta função é semelhante à de compra crédito.

Protótipo

```
LONG EXPORT ScopeCapturaPreAutorizacaoCredito (char *Valor, char *TxServico)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta.

Códigos Retorno Hexadecim al	Decimal	Significado
0xFC00	64512	Coletar cartão
0xFC01	64513	Coletar validade do cartão
0xFC02	64514	Imprime Cupom
0xFC09	64521	Coletar se a transação será a vista ou não
0xFC0E	64526	Coletar quantidade de parcelas
0FC0A	64522	Coletar se a transação será parcelada pela administradora ou pelo estabelecimento
0xFC18	64536	Coleta os 4 últimos dígitos do cartão
0xFC12	64530	Coleta controle da transação de pré-autorização
0xFCA5	64677	Coleta número do voucher
0xFC20	64544	Coleta pré-autorização
0FC3A	64570	Coleta NSU Host
0xFC8C	64652	Coleta data da transação original (pré-autorização)
0FCC0	64704	Coleta hora da transação original (pré-autorização)
0FC7C	64636	Coleta dados adicionais
0FC0E	64526	Coletar quantidade de parcelas
0FC20	64544	Coletar o número da pré-autorização
0FC33	64563	Coleta valor da taxa de serviço
0FC7E	64638	Go On Chip
0FC80	64640	Coleta o valor da taxa de embarque
0FC84	64644	Obtém os serviços
0FC85	64645	Coleta o cartão digitado
0FC29	64553	Coletar código de segurança
0FCA5	64677	Coleta o número do Voucher
0FCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
0FCFF	64767	Mostrar Informações e aguardar confirmação do operador

Possíveis Retornos de Erros

Códigos Retorno Hexadecim al	Decimal	Significado
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido

OxFB01	64257	Não foi possível criar a "thread" na coleta de dados
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF0C	65292	Transação não implementada
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
OxFF3A	65338	Erro interno na execução da coleta
OxFF60	65376	Função indisponível
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
char valor[12 + 1];
char taxa[12 + 1];

...
// obtém o valor da compra e armazena em 'valor'
// obtém a taxa de serviço e armazena em 'taxa'
// abre sessão
...
retorno = ScopeCapturaPreAutorizacaoCredito(valor, taxa);
...
// processa a transação
...
// fecha a sessão
...

```

Alteração de Pré-autorização de crédito

O valor de uma pré-autorização previamente realizada com sucesso pode ser alterado a qualquer momento através da realização de uma transação de alteração de valor de uma pré-autorização.

Protótipo

```
LONG EXPORT ScopeAlteraPreAutorizacaoCredito (char *Valor)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Significado	
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF05	65285	Transação de pré-autorização inexistente.
0xFF0A	65290	Banco de dados off-line.
0xFF85	65413	Cartão utilizado difere do utilizado na transação de pré-autorização.
0xFF86	65414	Identificador do terminal origem obrigatório.
0xFF87	65415	Transação de pré-autorização foi cancelada.
0xFF88	65416	Transação de pré-autorização foi confirmada anteriormente.
0xFF89	65417	Identificador do terminal origem é inválido ou não corresponde ao utilizado na transação de pré-autorização
0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```
...
char valor[12 + 1];
```

```
char taxa[12 + 1];  
...  
// obtém o valor da pré-autorização a será alterado e armazena em 'valor'  
// abre sessão  
...  
retorno = ScopeAlteraPreAutorizacaoCredito(valor);  
...  
// processa a transação  
...  
// fecha a sessão  
...
```

Venda IATA

Vendas IATA - *International Air Transportation Association* - são geralmente transações de crédito onde há a possibilidade de ser coletada a Taxa de Embarque, um valor geralmente cobrado em compras de viagens.

Não há um serviço/produto específico no Scope para este tipo de transação IATA. Somente estará habilitada de acordo com a configuração do Serviço de Crédito específico pelo SCOPECNF.

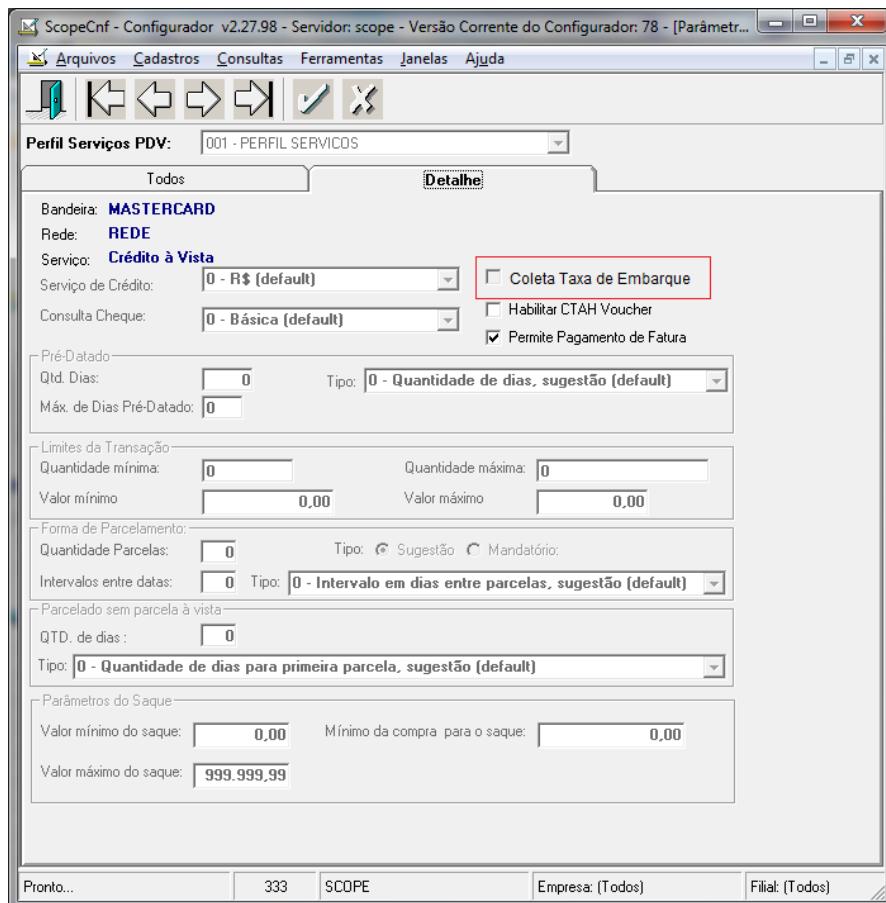
As informações a seguir descrevem os cenários para a REDE L06.01 e Cielo R2014 e Getnet LAC V02.92.

Configurações da Venda IATA

Configuração via SCOPECNF

A coleta de Taxa de Embarque é um dos estados do fluxo de transação de crédito, porém somente será solicitada a entrada para o operador caso a bandeira esteja parametrizada e com a Coleta de Taxa de Embarque habilitada.

A configuração deve ser feita pelo SCOPECNF – Perfil – Serviços – Parâmetros:



Na figura anterior, vemos que a bandeira MASTERCARD, da REDE, para o serviço de Crédito à Vista está com a Coleta de Taxa de Embarque não habilitada. Isso quer dizer que neste serviço a coleta da Taxa não será solicitada.

A parametrização pode ser feita por Serviço: Crédito à Vista, Crédito Parcelado Administradora e Crédito Parcelado Estabelecimento, de forma independente, ou seja, se desejar o estabelecimento pode habilitar a coleta da Taxa para serviços à vista, e manter desabilitado para serviços parcelados por exemplo.

IMPORTANTE: Para algumas autorizadoras, a coleta de Taxa de Embarque pode estar desabilitada para o estabelecimento via carga online de configuração. Este é caso da Cielo e da Getnet, por exemplo. A taxa somente será solicitada se estiver habilitada tanto pelo SCOPECNF quanto pela carga (se houver esta opção).

Configuração de coleta “somente” de Taxa de Embarque

As autorizadoras podem permitir que seja feita uma transação somente com o valor da Taxa de Embarque, ou seja, onde o valor da compra seja zero.

IMPORTANTE: Por padrão não é permitido que uma transação inicie com valor de compra zerado. As autorizadoras podem recusar a transação somente com Taxa de Embarque por limitações da própria autorizadora. Caso a transação seja negada pela autorizadora, verificar se esta funcionalidade é permitida por ela.

IMPORTANTE: A Getnet LAC somente permite coleta da Taxa de Embarque para o serviço de crédito parcelado loja.

Para que Scope aceite que uma transação de crédito (uma Venda IATA) seja iniciada com valor zero, é necessário configurar o Scope via [ScopeForneceCampo\(\)](#) antes de iniciar a transação:

```
...
retorno = ScopeForneceCampo(SCOPE_AUTOMACAO_PERMITE_SOMENTE_TX_EMBARQUE, "1");
...
retorno = ScopeCompraCartaoCredito("0", "0");
...
```

Valores válidos para o segundo parâmetro desta opção:

"0" ou **"N"** – para não permitir valores de compra zerados

"1" ou **"S"** – para permitir valores de compra zerados

Esta opção do [ScopeForneceCampo\(\)](#) somente pode ser usado para modificar entrada do valor principal para compras com [Cartão de Crédito](#). Não tem função para outros serviços (Débito, Pré-autorização etc).

IMPORTANTE:

- Esta informação será válida até que seja novamente chamada esta função com outro valor.
- Não é permitido uma transação ter valor total zerado. Caso uma transação inicie com valor de compra zerado, e a soma de valores + taxas seja igual a zero, a transação não será efetivada.
- A Getnet LAC não permite a realização de uma transação somente com o Valor de Embarque.

Coleta de Taxa de Embarque

A coleta de Taxa de Embarque é condicionada às configurações explicadas anteriormente, e também de que a autorizadora respondendo pela transação também tenha a funcionalidade de Venda IATA.

A transação deve ser iniciada normalmente como uma transação de crédito [ScopeCompraCartaoCredito\(\)](#). Caso o cartão utilizado seja da Cielo ou da REDE, no fluxo de crédito pode ser solicitada a coleta de Taxa de Embarque:

Se a automação decide responder automaticamente a coleta de Taxa, ela deve responder ao estado **TC_COLETA_VALOR_TAXA_EMBARQUE (0xFC80)**.

Para informar a Taxa de Embarque:

1. Tratar o estado **TC_COLETA_VALOR_TAXA_EMBARQUE**;
2. Chamar a função [ScopeResumeParam\(\)](#) informando a Taxa de Embarque que a automação já tem;

Exemplo:

```
...
switch (RC)
{
    case TC_COLETA_VALOR_TAXA_EMBARQUE:
```

```
// entrega a taxa coletada ao SCOPE na função abaixo
ScopeResumeParam(cod_coleta, dados, modo_entrada, acao);

...
}
```

Coleta do Valor de Entrada

A coleta do Valor de Entrada é solicitada após a coleta da Taxa de Embarque durante o fluxo de crédito, sendo que o valor deve ser maior que zero.

Se a automação decide responder automaticamente a coleta do Valor de Entrada, ela deve responder ao estado **TC_VALOR_ENTRADA (0xFC15)**.

Para informar o Valor de Entrada:

1. Tratar o estado TC_VALOR_ENTRADA;
2. Chamar a função ScopeResumeParam() informando o Valor de Entrada que a automação já tem;

Exemplo

```
...
switch (RC)
{
    case TC_VALOR_ENTRADA:

        // entrega o valor coletada ao SCOPE na função abaixo
        ScopeResumeParam(cod_coleta, dados, modo_entrada, acao);

        ...
}
```

IMPORTANTE: O Valor de Entrada é somente solicitado pela Getnet.

Obtendo dados da transação IATA

A automação pode verificar os valores da transação IATA através da função [ObtemCampoExt2\(\)](#).

O campo relativo à Taxa de Embarque é:

Máscara 2	Valor da taxa de embarque	0x00100000
-----------	---------------------------	------------

O campo relativo ao Valor de Entrada é:

Máscara 3	Valor de Entrada	0x00800000
-----------	------------------	------------

Veja outros dados sobre a transação no tópico [ObtemCampoExt2\(\)](#)

Cancelamento de uma transação IATA

A operação de cancelamento de uma transação IATA deve ser feita como um estorno comum de uma transação de crédito. Para isso deve ser usada a função [ScopeCancelamento\(\)](#). A documentação desta função esta descrita no capítulo Estorno de transações.

O valor a ser fornecido para o estorno deve ser o valor total da transação, ou seja, valor da compra somado com taxas.

Exemplo:

Valor da Compra:	R\$ 1000,00
Taxa de Embarque	R\$ 20,00
Valor Final:	R\$ 1020,00
Valor a ser Estornado:	R\$ 1020,00

No exemplo anterior, o valor a ser informado no fluxo de Cancelamento deve ser o valor final de **R\$ 1020,00**.

Transação IATA da Getnet

Atualmente a Getnet prevê somente um tipo de transação IATA:

- Crédito Parcelado sem juros

A Aplicação de Automação Comercial utiliza-se da função ScopeCompraCartaoCredito que segue os passos de coleta de uma transação de crédito parcelado lojista com a solicitação de duas novas coletas:

- Valor da Taxa de Embarque
- Valor de Entrada

O valor final da compra deverá ser o valor total a ser pago pelo portador do cartão, ou seja, o valor que deve ser passado na função ScopeCompraCartaoCredito deverá ser a soma do Valor da Passagem Aérea com o Valor da Taxa de Embarque.

Exemplo:

Valor da Passagem Aérea:	R\$ 1000,00
Valor da Taxa de Embarque:	R\$ 20,00
Valor da Compra:	R\$ 1020,00

```
...
char valor[12 + 1];
char taxa[12 + 1];
...
// 'valor' contém o Valor da Compra
// taxa de serviço não utilizado e 'taxa' com valor zerado
// abre sessão
...
retorno = ScopeCompraCartaoCredito(valor, taxa);
...
// processa a transação
...
```

```
// fecha a sessão
...

```

Simulação Crediário (Crédito)

Esta transação é de uso exclusivo de algumas redes que suportam a modalidade de negócio Crediário definida pela ABECS para compras com cartão de crédito.

Protótipo

LONG EXPORT ScopeSimulacaoCreditorio (WORD CodGrupoServico, char *Valor, char *TxServico)

Parâmetros

[in]	Numérico	CodGrupoServico	Código do Grupo de Serviço. Para uso desta funcionalidade através de cartões de Crédito usar fixo 2 (Crédito).
[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço (se houver).

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta.

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFC00	64512	Coletar cartão
OxFC01	64513	Coletar validade do cartão
OxFC02	64514	Imprime Cupom
OxFC0E	64526	Coletar quantidade de parcelas
OxFC18	64536	Coleta os 4 últimos dígitos do cartão
OxFC6D	64621	Coleta a data de emissão do cartão
OxFC1B	64539	Imprime consulta
Oxfc7d	64637	Coleta cancela transação
OxFC85	64645	Coleta o cartão digitado
OxFCF5	64757	Coleta se a transação será Crediário ou não
OxFCF8	64760	Coleta se uma nova consulta deve ser realizada ou não
OxFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
OxFCFF	64767	Mostrar Informações e aguardar confirmação do operador

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE06	65030	Logon duplicado

OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF0C	65292	Transação não implementada
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
char valor[12 + 1];
...

// obtém o valor da compra e armazena em 'valor'
// abre sessão
...

retorno = ScopeSimulacaoCreditorio(2, valor, "");
...

// processa a transação
...

// fecha a sessão
...

```

LEMBRETE: A transação de Simulação de Crediário pode ser efetuada no fluxo de crédito, neste caso, deverá ser chamada a função ScopeCompraCartaoCrédito().

Quando configurada na inicialização da rede, o SCOPE irá habilitar os serviços de crediário crédito (173) e Simulação de Crediário Crédito (174).

Cartão de débito

Abrangendo diversos serviços, como o cartão de débito, é uma das transações mais realizadas nos estabelecimentos. Falando em serviços, nos referimos a débito à vista, débito pré-datado, débito parcelado com a 1ª parcela agendada ou à vista, débito Voucher, saque e CDC.

Compra com cartão de débito

Para os serviços de débito à vista, débito pré-datado, débito parcelado com a 1ª parcela agendada ou à vista e débito Voucher, deve-se utilizar a função ScopeCompraCartaoDebito(). Como as outras transações, o fluxo de processamento é similar.

Protótipo

```
LONG EXPORT ScopeCompraCartaoDebito (char *Valor)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta

Códigos Retorno	Significado	
Hexadecimal	Decimal	
0xFC00	64512	Coletar cartão
0xFC01	64513	Coletar validade do cartão
0xFC02	64514	Imprime Cupom
0xFC03	64515	Coletar CGC ou CPF
0xFC09	64521	Coletar se a transação será à vista ou não
0xFC0B	64523	Coletar se a transação será pré-datada
0xFC0C	64524	Coletar se a parcela será à vista
0xFC0D	64525	Coletar quantidade de dias entre parcelas
0xFC0E	64526	Coletar quantidade de parcelas
0xFC10	64528	Coletar o dia e o mês (DDMM)
0xFC11	64529	Coletar a senha
0xFC13	64531	Coletar a forma de pagamento
0xFC14	64532	Coletar data do primeiro vencimento
0xFC15	64533	Coletar valor de entrada
0xFC18	64536	Coletar últimos dígitos do cartão
0xFC1A	64538	Coletar se deseja consultar parcelas
0xFC1B	64539	Imprime consulta
0xFC22	64546	Imprime nota promissória
0xFC23	64547	Coletar CEP
0xFC24	64548	Coletar número do endereço
0xFC25	64549	Coletar parte numérica do complemento
0xFC2B	64555	Coleta se é com ou sem garantia de pré-datado
0xFC2C	64556	Coleta se aceita ou não risco
0xFC30	64560	Coleta número do telefone

OxFC32	64562	Coleta data (formato DDMMAA)
OxFC33	64563	Coleta valor da taxa de serviço
OxFC47	64583	Coleta quantidade de parcelas e aceita 1 parcela
OxFC51	64593	Imprime cupom promocional
OxFC52	64594	Coleta se utiliza saldo
OxFC64	64612	Coleta RG
OxFC66	64614	Coleta somente CPF
OxFC7B	64635	Coleta o valor da primeira parcela
OxFC7D	64637	Coleta se cancela ou não a transação
OxFC7E	64638	Go On Chip
OxFC7F	64639	Retira o cartão
OxFC80	64640	Coleta o valor da taxa de embarque
OxFC85	64645	Coleta o cartão digitado
OxFC87	64647	Exibe o menu
OxFC95	64661	Coleta o valor das parcelas
OxFC96	64662	Coleta se a primeira parcela é para 30 ou 60 dias
OxFC9F	64671	Coleta a placa do veículo (transação de convênio combustível Banrisul)
OxFCA5	64677	Coleta o número do Voucher
OxFCBE	64702	Coleta dados ECF
OXFCC1	64705	Coleta opção de pagamento de carnê/fatura no serviço de débito a vista
OxFCCB	64715	*Coleta da Lista de Mercadorias Consumidas
OxFCDF	64735	Coleta de Código para Identificação pagamento de Fatura (Exemplos de código: CPF, Código de Barras e Cartão)
OxFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
OxFCFF	64767	Mostrar Informações e aguardar confirmação do operador

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```
...
char valor[12 + 1];
...
```

```

// obtém o valor da compra e armazena em 'valor'
...
// abre sessão
...
retorno = ScopeCompraCartaoDebito(valor);
...
// processa a transação
...
// fecha a sessão
...

```

Compra CDC (Crédito Direto ao Consumidor)

Apesar de existir uma função ScopeCompraCDC() responsável por essa modalidade de serviço, esta transação foi incorporada ao fluxo da transação ScopeCompraCartaoDebito() devido exigência da especificação Visanet 4.1. O que difere no fluxo de um pagamento com cartão de débito é que um CDC é um débito parcelado pela administradora.

Caso no fluxo de compra CDC for realizado uma consulta, os cupons da consulta não poderão ser impressos devido a problemas de controle fiscal. Estes cupons deverão ser exibidos no visor para o cliente. Somente os cupons da compra é que deverão ser impressos caso haja interesse em continuar a compra após a consulta.

Consulta CDC

Para uma simples consulta CDC, independente da compra, deve-se utilizar a função específica para a transação: ScopeConsultaCDC(). O resultado desta transação será um cupom com valores do parcelamento com as taxas.

Quando a consulta for realizada por esta função, os cupons da consulta deverão ser enviados para a impressora.

Protótipo

LONG EXPORT ScopeConsultaCDC (char *Valor, char *TxServico)

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE06	65030	Logon duplicado

OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF0C	65292	Transação não implementada
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
char valor[12 + 1];
char taxa[12 + 1];
...

// obtém o valor da compra e armazena em 'valor'
// obtém a taxa de serviço e armazena em 'taxa'
// abre sessão
...

retorno = ScopeConsultaCDC(valor, taxa);
...

// processa a transação
...

// fecha a sessão
...

```

LEMBRETE: a compra CDC é efetuada no fluxo de débito, ou seja, deverá ser chamada a função ScopeCompraCartaoDebito() para a realização dessa compra.

Consulta a saldo de cartão de débito

Um dos motivos que uma transação pode não ser aprovada é a falta de saldo suficiente do cliente no cartão de débito. Alguns cartões permitem esta consulta.

Protótipo

```
LONG EXPORT ScopeConsultaSaldoDebito (char *Valor)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Exemplo

```
...
char valor[12 + 1];
...
// obtém o valor da compra e armazena em 'valor'
...
// abre sessão
...
retorno = ScopeConsultaSaldoDebito(valor);
...
// processa a transação
...
// fecha a sessão
...
```

Débito para pagamento de fatura

Para os serviços de “Débito para Pagamento de Fatura” deve-se utilizar a função ScopeDebitoPagamentoFatura. Trata-se de uma transação de débito, utilizada especificamente para pagamento de uma fatura. Essa transação substitui a pergunta “Pagamento de Fatura?” (estado de coleta 0xFCC1 = 64705) que pode ocorrer no fluxo de débito à vista. O restante do fluxo de processamento é similar ao de débito. No caso da rede CIELO, essa função afeta o “Menu Dinâmico”, que será filtrado, excluindo-se os elementos especificamente de débito da lista, mantendo-se apenas os elementos relacionados com pagamento de fatura/carnê. Caso restar apenas um elemento após o filtro, este será selecionado, e o menu não será exibido.

Embora essa transação seja de débito, o grupo de serviço ao qual ela pertence é separado (Débito para Pagamento). Desta forma é possível definir um lista de prioridade diferente da definida pelo grupo Débito. Assim, pode-se definir que uma transação de Débito para Pagamento de Fatura seja direcionada para uma rede diferente da definida para a transação de Débito da mesma bandeira. O parâmetro “Permite Pagamento de Fatura” no perfil de serviços para os produtos de débito foi mantido por questão de compatibilidade com o legado. Portanto, para o correto funcionamento desta transação, que está diretamente relacionada com a transação de débito à vista, deve-se configurar adequadamente o perfil de serviços, desmarcando-se esta opção nas transações de débito para evitar que a pergunta “Pagamento de Fatura?” seja exibida no fluxo de Débito.

Protótipo em C/C++

```
LONG EXPORT ScopeDebitoPagamentoFatura (char *Valor)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta

Códigos Retorno	Significado	
Hexadecim al	Decimal	
0xFC00	64512	Coletar cartão
0xFC01	64513	Coletar validade do cartão
0xFC02	64514	Imprime Cupom
0xFC03	64515	Coletar CGC ou CPF
0xFC09	64521	Coletar se a transação será a vista ou não
0xFC0B	64523	Coletar se a transação será pré-datada
0xFC0C	64524	Coletar se a parcela será à vista
0xFC0D	64525	Coletar quantidade de dias entre parcelas
0xFC0E	64526	Coletar quantidade de parcelas
0xFC10	64528	Coletar o dia e o mês (DDMM)
0xFC11	64529	Coletar a senha
0xFC13	64531	Coletar a forma de pagamento
0xFC14	64532	Coletar data do primeiro vencimento
0xFC15	64533	Coletar valor de entrada
0xFC18	64536	Coletar últimos dígitos do cartão
0xFC1A	64538	Coletar se deseja consultar parcelas
0xFC1B	64539	Imprime consulta
0xFC22	64546	Imprime nota promissória
0xFC23	64547	Coletar CEP
0xFC24	64548	Coletar número do endereço
0xFC25	64549	Coletar parte numérica do complemento
0xFC2B	64555	Coleta se é com ou sem garantia de pré-datado
0xFC2C	64556	Coleta se aceita ou não risco
0xFC30	64560	Coleta número do telefone
0xFC32	64562	Coleta data (formato DDMMAA)
0xFC33	64563	Coleta valor da taxa de serviço
0xFC47	64583	Coleta quantidade de parcelas e aceita 1 parcela
0xFC51	64593	Imprime cupom promocional
0xFC52	64594	Coleta se utiliza saldo
0xFC64	64612	Coleta RG
0xFC66	64614	Coleta somente CPF
0xFC7B	64635	Coleta o valor da primeira parcela
0xFC7D	64637	Coleta se cancela ou não a transação

0xFC7E	64638	Go On Chip
0xFC7F	64639	Retira o cartão
0xFC80	64640	Coleta o valor da taxa de embarque
0xFC85	64645	Coleta o cartão digitado
0xFC87	64647	Exibe o menu
0xFC95	64661	Coleta o valor das parcelas
0xFC96	64662	Coleta se a primeira parcela é para 30 ou 60 dias
0xFC9F	64671	Coleta a placa do veículo (transação de convênio combustível Banrisul)
0xFCA5	64677	Coleta o número do Voucher
0xFCBE	64702	Coleta dados ECF
0xFCCB	64715	Coleta da Lista de Mercadorias Consumidas
0xFCDF	64735	Coleta de Código para Identificação pagamento de Fatura (Exemplos de código: CPF, Código de Barras e Cartão)
0xFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
0xFCFF	64767	Mostrar Informações e aguardar confirmação do operador

Possíveis Retornos de Erros

Códigos Retorno	Hexadecim al	Significado
Hexadecim al	Decimal	Significado
OxFA01	64001	Parâmetro 1 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo em C/C++

```

...
char valor[12 + 1];

...
// obtém o valor da compra e armazena em 'valor'
...

// abre sessão
...

retorno = ScopeDebitoPagamentoFatura(valor);
...

// processa a transação

```

```

...
// fecha a sessão
...

```

Protótipo em Java

```
public int debitoPagamentoFatura (String valor)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta

Códigos Retorno Hexadecim al	Decimal	Significado
0xFC00	64512	Coletar cartão
0xFC01	64513	Coletar validade do cartão
0xFC02	64514	Imprime Cupom
0xFC03	64515	Coletar CGC ou CPF
0xFC09	64521	Coletar se a transação será a vista ou não
0xFC0B	64523	Coletar se a transação será pré-datada
0xFC0C	64524	Coletar se a parcela será à vista
0xFC0D	64525	Coletar quantidade de dias entre parcelas
0xFC0E	64526	Coletar quantidade de parcelas
0xFC10	64528	Coletar o dia e o mês (DDMM)
0xFC11	64529	Coletar a senha
0xFC13	64531	Coletar a forma de pagamento
0xFC14	64532	Coletar data do primeiro vencimento
0xFC15	64533	Coletar valor de entrada
0xFC18	64536	Coletar últimos dígitos do cartão
0xFC1A	64538	Coletar se deseja consultar parcelas
0xFC1B	64539	Imprime consulta
0xFC22	64546	Imprime nota promissória
0xFC23	64547	Coletar CEP
0xFC24	64548	Coletar número do endereço
0xFC25	64549	Coletar parte numérica do complemento
0xFC2B	64555	Coleta se é com ou sem garantia de pré-datado
0xFC2C	64556	Coleta se aceita ou não risco
0xFC30	64560	Coleta número do telefone
0xFC32	64562	Coleta data (formato DDMMAA)
0xFC33	64563	Coleta valor da taxa de serviço
0xFC47	64583	Coleta quantidade de parcelas e aceita 1 parcela
0xFC51	64593	Imprime cupom promocional

OxFC52	64594	Coleta se utiliza saldo
OxFC64	64612	Coleta RG
OxFC66	64614	Coleta somente CPF
OxFC7B	64635	Coleta o valor da primeira parcela
OxFC7D	64637	Coleta se cancela ou não a transação
OxFC7E	64638	Go On Chip
OxFC7F	64639	Retira o cartão
OxFC80	64640	Coleta o valor da taxa de embarque
OxFC85	64645	Coleta o cartão digitado
OxFC87	64647	Exibe o menu
OxFC95	64661	Coleta o valor das parcelas
OxFC96	64662	Coleta se a primeira parcela é para 30 ou 60 dias
OxFC9F	64671	Coleta a placa do veículo (transação de convênio combustível Banrisul)
OxFCA5	64677	Coleta o número do Voucher
OxFCBE	64702	Coleta dados ECF
OxFCCB	64715	Coleta da Lista de Mercadorias Consumidas
OxFCDF	64735	Coleta de Código para Identificação pagamento de Fatura (Exemplos de código: CPF, Código de Barras e Cartão)
OxFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
OxFCFF	64767	Mostrar Informações e aguardar confirmação do operador

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo em Java

```

...
int retorno;
String valor;
...
// obtém o valor da compra e armazena em 'valor'
...
// abre sessão

```

```
...
Scope scope = new Scope();
retorno = scope.debitoPagamentoFatura(valor);
...
// processa a transação
...
// fecha a sessão
```

Carteira Virtual

Também conhecido como pagamento via QR Code, já é utilizado por várias operadoras e atualmente apresenta apenas um serviço: compra à vista, não sendo possível o parcelamento da mesma.

Obtém a Lista de Carteiras Virtuais Disponíveis

A função **ScopeObtemListaCarteiraVirtual** está obsoleta, mas continua sendo suportada. A função atual é **ScopeObtemListaCarteiraVirtualEx** utilizada pela aplicação para conhecer todas as carteiras virtuais disponíveis e fazer um filtro antes da exibição dessa informações na tela de opções.

Protótipo

```
LONG EXPORT ScopeObtemListaCarteiraVirtualEx (char *_Versao, WORD _Servico, char *_Buffer,
WORD _TamBuffer)
```

```
dllScopeAPI ScopeObtemListaCarteiraVirtualEx(char *_Versao, WORD _Servico, char *_Buffer, WORD
_TamBuffer);
```

Parâmetros

[in]	Char[2]	_Versao	Versão do formato desejado. Fixo "01".
[in]	WORD	Servico	Usar um dos seguintes serviços <ul style="list-style-type: none"> • 168: S_QR_CODE_COMPRADOR • 169: S_QR_CODE_VENDEDOR • 176: S_SAQUE_VIA_QRCODE Caso deseje obter a lista de todos os produtos dos serviços acima, passar 0 neste campo.
[out]	char *	_Buffer	Buffer de saída, conterá um ou mais registros com itens encontrados. A variável deverá ser pré alocada pelo chamador e possuir tamanho suficiente. Verificar o formato dos dados no tópico <i>Formato dados</i> .
[in]	WORD	_TamBuffer	Indica o tamanho de _Buffer.

Formato dos dados devolvidos no Buffer de saída

NOME	Formato	Descrição
Versao	AN2	Fixo '01'
QTD_PRODUTOS	AN2	Quantidade de itens (máximo 24)
		Descrição dos itens:
COD_BANDEIRA	AN3	Ref.: Apêndice A: Código das bandeiras
NOME_BANDEIRA	ANS..41	Nome da bandeira (para display)
COD_REDE	AN3	Ref.: Apêndice A: Código das redes
NOME_REDE	ANS..41	Nome da rede

Exemplo

```
#define CARTVIRT_MAX_PRODUTOS 24
```

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

```

typedef struct
{
    char Versao[2]; // Fixo '01' (para viabilizar eventuais evoluções futuras)
    char QtdProdutos[2];
    stCV_BandRede sProduto[CARTVIRT_MAX_PRODUTOS];
} stCV_ListaBandRede, *LPCV_ListaBandRede;

typedef struct
{
    char CodBandeira[3];
    char szNomeBandeira[40 + 1];
    char CodRede[3];
    char szNomeRede[40 + 1];
} stCV_BandRede, *LPCV_BandRede;

```

Retorno

Ver [tabela de código de retorno](#).

Compra com a Carteira Virtual

A função ScopeCompraCartaoVirtual está obsoleta, mas continua sendo suportada. A função atual para o módulo de Carteira Virtual, para todos os serviços disponíveis devem ser acessador por: **ScopeCarteiraVirtualEx()** ou **ScopeCarteiraVirtualEx2()**, descritas abaixo.

Protótipo

```
LONG EXPORT ScopeCarteiraVirtualEx (WORD _CodBandeira, WORD _CodRede)
```

Parâmetros

CodBandeira = 0 (indefinido, o SCOPE retornará todas as bandeiras existentes)
!= 0 (o SCOPE retornará apenas os itens que correspondem com essa bandeira)

CodRede = 0 (indefinido, o SCOPE retornará todas as redes existentes)
!= 0 (o SCOPE retornará apenas os itens que correspondem com essa rede)

Essas informações podem ser obtidas ao chamar a função: ScopeObtemListaCarteiraVirtualEx

Protótipo

```
LONG EXPORT ScopeCarteiraVirtualEx2(char *_Valor, WORD _CodBandeira, WORD _CodRede, WORD _CodServico)
```

Parâmetros

Valor	= Valor da Transação. Se igual a (0)zero, o valor será solicitado posteriormente CodBandeira = 0 (indefinido, o SCOPE retornará todas as bandeiras existentes) != 0 (o SCOPE retornará apenas os itens que correspondem com essa bandeira)
CodRede	= 0 (indefinido, o SCOPE retornará todas as redes existentes) != 0 (o SCOPE retornará apenas os itens que correspondem com essa rede)
CodServico	= 0 (o SCOPE retornará os serviços disponíveis para a transação) != 0 O código do Serviço desejado.
Essas informações podem ser obtidas ao chamar a função: ScopeObtemListaCarteiraVirtualEx	

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta.

Códigos Retorno			Significado
Estado	Hexa	Decimal	
TC_OBTEM_SERVICOS	0xFC84	64644	Serviços disponíveis para a transação. Será retornado somente se o parâmetro CFG_OBTER_SERVICOS estiver habilitado.
TC_EXIBE_MENU	0xFC87	64647	Exibe o menu das carteira virtuais disponíveis
TC_OBTEM_QRCODE	0xFCF3		Quando a string de QRCode estiver disponível, será obtida através da função ScopeObtemCampoExt3
TC_IMPRIME_CUPOM	0xFC02	64514	Impressão do cupom.

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim	Decimal	
0xEE01	60929	Timeout Server-NA
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA02	64003	Parâmetro 3 inválido
0xFA02	64004	Parâmetro 4 inválido
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF

OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF0C	65292	Transação não implementada
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
OxFF3A	65338	Erro interno na execução da coleta
OxFF60	65376	Função indisponível
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
// abre sessão
...
retorno = ScopeCarteiraVirtualEx (0,0);
...
// processa a transação
...
// fecha a sessão
...

```

Caso já estejam definidos o código da Rede e/ou Bandeira a serem utilizados, assim como o valor da Transação e o código do Serviço, esses parâmetros podem ser passados diretamente na chamada da função ScopeCarteiraVirtualEx2(). Por exemplo (CodRede = 154 (Mercado Pago), CodBandeira = 432 (PIX) e CodServiço = 0 (indefinido)):

Exemplo

```

...
// abre sessão
...
retorno = ScopeCarteiraVirtualEx (123456,154,432,0);

```

```

...
// processa a transação
...
// fecha a sessão
...

```

Exemplo para exibição do Menu, caso não seja definida a bandeira/rede

```

...
// Tratamento dos estados de coleta
switch(RC)
{
...
case TC_EXIBE_MENU
    ScopeMenuRecuperaltens(MNU_TAB_TIPO_GENERICO, Buffer, TamBuffer)
    // Retorna em Buffer uma lista de opções
    // Exibe a lista de opções
    ....
    break;
}
...

```

Para indicar ao SCOPE qual item foi selecionado, chamar a

ScopeMenuSelecionaltem.

Detalhamento para utilizar a função ScopeMenuRecuperaltens:

Parâmetros:

MNU_TAB_TIPO_GENERICO: definido com o valor 2

- Buffer: recebe a resposta no seguinte formato:

```

struct TipoScopeMenuDinamico {
    char TipoTabela;      // MNU_TAB_TIPO_GENERICO = 2
    char QtldItens;
    TipoScopeMenuGenerico stItemGenerico;
};

struct TipoScopeMenuGenerico {
    char TipoMenu;        //MNU_GENERICO_TIPO_OPCOES = 4
    char strItem[15][40+1]; // Máximo 15 itens em formato "string" de no máximo 40 caracteres
};

```

- TamBuffer: passar um tamanho de buffer de no mínimo 1024 bytes.

Exemplo para exibir a imagem do QRcode na tela:

```
// Tratamento dos estados de coleta
switch(RC)
{
...
case TC_OBTEM_QRCODE
    ScopeObtemCampoExt3 (Handle,0,0,0,Masc4.QRCode, 0, StringQRcode)
//Máscara4: String para gerar o QRCode=0x00000020
...
//Converter StringQRcodeem imagem qrcode
...
break;
}
```

Para exibir a imagem do QRcode no pinpad:

```
// Tratamento dos estados de coleta
switch(RC)
{
...
case TC_OBTEM_QRCODE
    ScopeObtemCampoExt3 (Handle,0,0,0,Masc4.QRCode, 0, StringQRcode)
//Máscara4: String para gerar o QRCode=0x00000020
...
//Converter StringQRcodeem arquivo de imagem (png, jpg...)
...
// Utilizar as funções Multimídia do pinpad detalhadas
// no tópico: Pinpad Compartilhado e ABECS – Comandos

ScopePPMMFileLoad ...
ScopePPMMDisplayImage ...
ScopePPMMFileDelete ...

break;
}
```

Algumas redes de carteira virtual fornecem a forma de pagamento que foi realizada no app de celular.

Para obter informações sobre modo de pagamento:

Se essa informação for disponibilizada pelo adquirente, ela será obtida através da função ScopeObtemCampoExt2 ou ScopeObtemCampoExt3

Máscara 3	Nome da máscara	Descrição
0x20000000	Modo_Pagamento	Modo de Pagamento: "00" : não especificado "03" : debito conta (carteira digital) "0 4": cartao credito "0 5":pix

Observação: a informação **Modo_Pagamento** só está disponível no final do fluxo da função "Compra com a Carteira Virtual".

Em transações de PIX, é possível obter o número do ISPB da instituição financeira utilizada para realizar o pagamento.

Se essa informação for disponibilizada pelo adquirente, ela será obtida através da função **ScopeObtemCampoExt3**, com o parâmetro **Ispb_Fonte_Pagadora** na máscara 4.

Máscara 4	Nome da máscara	Descrição
0x00200000	Ispb_Fonte_Pagadora	Retorna os 8 números que é referente o ISPB da instituição utilizada para realizar o pagamento do PIX.

Observação: a informação **Ispb_Fonte_Pagadora** só estará disponível no final do fluxo da função "Compra com a Carteira Virtual", ou seja, depois que a transação é aprovada. Então a automação deve fazer a chamada da função **ScopeObtemCampoExt3**. Exemplo:

```
char szBuffer[128 + 1]; ScopeObtemCampoExt3(Handle, 0x00, 0x00, 0x00, Ispb_Fonte_Pagadora, '#', szBuffer);
```

Saque com a Carteira Virtual

Aciona o SCOPE Client para efetuar um Saque através de QRCode. Esta função é equivalente à chamar a ScopeTransacaoFinanceira com código de serviço = "176 – Saque via QRCode".

Protótipo

```
LONG EXPORT ScopeSaqueCarteiraVirtual (char *_Valor, WORD _CodBandeira, WORD _CodRede)
```

Parâmetros

[in]	String	Valor	Valor da transação
[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que não deseja uma bandeira específica
[in]	WORD	CodRede	Código da rede do produto a ser executado. Use 0 (zero) para indicar que não deseja uma rede específica.

Retorno

Ver [tabela de código de retorno](#).

Cartão Dinheiro

São cartões ao portador, protegido por senha, semelhante ao cartão de débito que é adquirido (comprado) com um valor fixo, e posteriormente pode ser recarregado. É utilizado principalmente para efetuar compras, debitando esse valor do saldo do cartão, cujo objetivo é a substituição de dinheiro por este cartão. Também, é conhecido como Cartão Presente ou Gift Card.

Operações

As operações (transações) que podem ser efetuadas com o cartão dinheiro são:

- Compra/Carga de cartão
- Compra usando cartão (débito)
- Consulta a saldo (e valores permitidos para carga)
- Estornos

Compra/Carga do cartão dinheiro

Para realizar a operação de recarga, e também, a compra (carga inicial) do cartão dinheiro, deve-se utilizar a função ScopeCartaoDinheiroEx(), como descrito abaixo.

Como as outras transações, o fluxo de processamento é similar.

Protótipo

```
LONG EXPORT ScopeCartaoDinheiroEx (WORD servico, char *Valor)
```

Parâmetros

[in]	WORD	Serviço	Para efetuar a operação de Compra/Carga de cartão dinheiro usar o serviço 110 S_CARGA_CARTAO_DINHEIRO
[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Significado	
Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFF0C	65292	Transação não implementada
OxFF25	65317	Valor inválido

Exemplo

```

...
Long servico = S_CARGA_CARTAO_DINHEIRO; // valor de 110
char valor[12 + 1];
...

// obtém o valor da compra e armazena em 'valor'
// abre sessão
...

retorno = ScopeCartaoDinheiroEx(servico, valor);
...

// processa a transação
...

// fecha a sessão
...

```

Estados de coleta

A tabela abaixo mostra alguns estados de coleta que o fluxo poderá retornar, no modo coleta. A maioria dos estados não necessitam tratamentos específicos pela automação. Basta que mostre a mensagem na tela do operador e aguarde a digitação. Quando necessário, está descrito abaixo o tratamento específico que a automação poderá fazer.

ESTADO DE COLETA		DESCRÍÇÃO	TRATAMENTO
HEX	DECIMAL		
0xFC34	64564	Coleta valor de carga Pode ocorrer após o estado 0xFC5F, caso optado por não efetuar a consulta saldo ou 0xFC8B, caso tenha ocorrido a consulta saldo	Padrão
0xFC5F	64607	Coleta se deve efetuar a consulta saldo	Padrão
0xFC8B	64651	Disponibiliza valores da consulta saldo (Indica que a Consulta saldo foi efetuada)	A partir deste estado podem ser obtidos Dados da resposta de Consulta de valor da EPAY ou Gift Card
0xFCFF	64767	Mostrar Informações e aguardar confirmação do operador Caso ocorra imediatamente após o estado 0xFC5F, refere-se a confirmação do valor fixo retornado pela consulta.	Padrão

É possível realizar a Carga de Cartão Dinheiro utilizando-se a tarja magnética ou o código de barras. Assim, ao ser solicitada a função de carga (ScopeCartaoDinheiroEx()), o Scope irá solicitar primeiramente o código de barras. Se o operador desejar passar o cartão ao invés de digitar o código de barras, o mesmo terá que clicar em "cancelar" na tela de coleta do código de barras e assim será solicitada a entrada do cartão.

O valor definido na chamada da função ScopeCartaoDinheiroEx() pode influenciar o fluxo de coleta, para algumas redes autorizadoras, permitindo que seja efetuada a transação de Consulta a saldo com a rede autorizadora, antes de realizar a transação de Carga. Sendo assim, caso o valor chamado seja superior a R\$ 0,00 ("000"), apenas a transação de Carga é efetuada. Caso o valor chamado seja igual a R\$ 0,00 ("000"), permite efetuar a transação de Consulta do cartão ou prosseguir sem efetuar a consulta. De qualquer forma, será solicitado que seja coletado ou confirmado o valor, sendo que a confirmação ocorre apenas quando a consulta é efetuada e retorna valor fixo para a carga.

IMPORTANTE: a função ScopeCartaoDinheiroEx() substitui a anterior ScopeCartaoDinheiro(). No entanto, esta última ainda é mantida por compatibilidade.

Compra usando o cartão dinheiro (débito)

Para realizar a operação compra usando o cartão dinheiro, isto é, debitar o valor da compra do saldo disponível, pode ser feito de duas formas:

1. Usar a função ScopeCompraCartaoDebito(), como se fosse um cartão de débito normal (vide item Compra com cartão de débito).
2. Usar a função ScopeCartaoDinheiroEx(), como descrito abaixo.

Como as outras transações, o fluxo de processamento é similar.

Protótipo

```
LONG EXPORT ScopeCartaoDinheiroEx (WORD servico, char *Valor)
```

Parâmetros

[in]	WORD	Serviço	Para efetuar a operação de Compra/Carga de cartão dinheiro usar o serviço 98 S_COMPRA_CARTAO_DINHEIRO
[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")

Retorno

Ver [tabela de código de retorno](#).

Exemplo

```
...
Long servico = S_COMPRA_CARTAO_DINHEIRO; // valor de 98
char valor[12 + 1];
...
// obtém o valor da compra e armazena em 'valor'
// abre sessão
...
retorno = ScopeCartaoDinheiroEx(servico, valor);
...
// processa a transação
...
// fecha a sessão
```

| ...

IMPORTANTE: a função ScopeCartaoDinheiroEx() substitui a anterior ScopeCartaoDinheiro(). No entanto, esta última ainda é mantida por compatibilidade.

Consulta ao saldo de cartão dinheiro

A consulta ao saldo de cartão dinheiro e a respectiva impressão de cupom deve ser feita usando a função ScopeConsultaCartaoDinheiro(), como descrito abaixo.

Protótipo

```
LONG EXPORT ScopeConsultaCartaoDinheiro (void)
```

Parâmetros

Não há parâmetros.

Retorno

Ver [tabela de código de retorno](#).

Exemplo

```
...
// abre sessão
...
retorno = ScopeConsultaCartaoDinheiro();
...
// processa a transação
...
// fecha a sessão
...
```

Houve mudanças no fluxo da consulta, hoje é possível realizar a mesma utilizando-se o código de barras. Assim, ao ser solicitada a função de carga (ScopeConsultaCartaoDinheiro()), o Scope irá solicitar primeiramente o código de barras. Se o operador desejar passar o cartão ao invés de digitar o código de barras, o mesmo terá que clicar em "cancelar" na tela de coleta do código de barras e assim será solicitada a entrada do cartão.

Estornos

Tanto para realizar a operação de estorno de compra/carga do cartão dinheiro como a compra usando o cartão dinheiro (débito) deve se usar a função genérica de estorno chamada ScopeCancelamento(). A documentação desta função esta descrita no capítulo Estorno de transações.

Se a transação original de uma carga foi feita através do "código de barras", no fluxo de estorno será solicitado somente o "código de controle" para recuperar os dados da transação original. Nessa situação o Scope recuperará o número do cartão do banco de dados.

Se a transação original de uma carga foi feita através da entrada do cartão, o fluxo se mantém, ou seja, durante o estorno será solicitado o código de controle mais a entrada do cartão.

Funções de Consulta

Neste capítulo falamos sobre as transações de consultas que não se referem aos cartões.

Cheque

Outra funcionalidade que o SCOPE dispõe é a consulta de cheques no Serasa ou na ACSP – Associação Comercial de São Paulo. Nas consultas de cheques, via PDVs, as respostas são baseadas no número do documento (CPF/CGC), banco, número do cheque do comprador, valor e data de vencimento da compra.

O número do cheque é composto por seis caracteres numéricos, por definição do Banco Central do Brasil com base na resolução 885, de 22/12/1983.

Consulta de cheques

A transação de consulta a cheque é iniciada pela função ScopeConsultaCheque(). O fluxo de processamento é similar às outras transações.

Protótipo

```
LONG EXPORT ScopeConsultaCheque (char *Valor)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado

OxFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
char valor[12 + 1];
...
// obtém o valor do cheque e armazena em 'valor'
// abre sessão
...
retorno = ScopeConsultaCheque(valor);
...
// processa a transação
...
// fecha a sessão
...

```

AVS

Termo em inglês "Address Verification Service", também conhecido por outras autorizadoras como VAS – Verificação Automática de Endereço, é um serviço de verificação de endereço do portador do cartão utilizado para comparar o endereço que o cliente forneceu com o que está cadastrado junto com a administradora de cartão. O AVS é uma transação não financeira e não é atrelada a nenhuma transação de venda, isto é, a consulta não garante que a pessoa que esteja realizando a transação seja realmente o portador do cartão.

CUIDADO:Este tipo de informação recebida pela consulta AVS tem o intuito apenas de oferecer suporte para a decisão do estabelecimento comercial, quanto à realização ou não da venda.

Consulta AVS

A função ScopeConsultaAVS() aciona o SCOPE Client para efetuar uma transação de consulta AVS para confirmar os dados do endereço fornecido com o cadastrado na administradora.

Protótipo

LONG EXPORT ScopeConsultaAVS (void)

Parâmetros

Não há parâmetros.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF0C	65292	Transação não implementada
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
// abre sessão
...
retorno = ScopeConsultaAVS();
...
// processa a transação
...
// fecha a sessão
...

```

Pagamento

Consulta Pagamento

A função ScopeConsultaPagamento() aciona o SCOPE Client para efetuar uma transação de consulta de pagamento de títulos e convênios para confirmar se os dados do pagamento são válidos permitindo que se realize uma transação de pagamento posteriormente.

Protótipo

```
LONG EXPORT ScopeConsultaPagamento (WORDCodBandeira)
```

Parâmetros

[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.
------	------	-------------	---

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF0C	65292	Transação não implementada
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISSO
0xFF7D	65405	Chamada da função não permitida

Exemplo

```
...
// abre sessão
...
retorno = ScopeConsultaPagamento( (WORD) CódigoBandeira);
...
// processa a transação
...
```

```
// fecha a sessão
...

```

Status do Cartão

Consulta Cartão Online

A função ScopeConsultaCartaoOnline() aciona o SCOPE Client para efetuar uma transação de consulta que validará os dados capturados do cartão e retornará se o cartão é/está válido ou não. Não é necessário uso de valor para esta consulta, já que o objetivo é validar se o cartão é válido e não efetivar uma transação com ele. Utilizado geralmente em ambiente e-commerce, ou seja, sem PIN-pad configurado.

Utilizado no momento somente pela autorizadora *Global Payments* e neste caso pode ser utilizado somente com cartões de crédito.

O uso desta função com outros cartões ou mesmo de outras autorizadoras retornará sempre código de erro.

Na consulta, os seguintes dados podem ser solicitados:

- Número do cartão – modo digitado
- Últimos 4 dígitos – se a autorizadora solicitar
- Data Validade do cartão
- CVV/CVC – se a autorizadora solicitar

O resultado desta operação será indicado pelo resultado retornado da transação.

Retorno	Resultado
0	Sucesso - Dados Válidos – Cartão válido.
Diferente de zero	Dados inválidos (cartão com problema), ou cartão incorreto, ou serviço indisponível. Outros códigos podem ser retornados de acordo com tabela de código de retorno.

Esta consulta não gera comprovante.

Protótipo

```
LONG EXPORT ScopeConsultaCartaoOnline (void)
```

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Significado
-----------------	-------------

Hexadecimal	Decimal	
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFF6A	65386	Permite somente digitado

Exemplo

```

...
// abre sessão
...
retorno = ScopeConsultaCartaoOnline();
...
// processa a transação
...
// fecha a sessão
...

```

Recarga de celular

Com o crescimento número de pessoas que utilizam celulares do tipo pré-pago, a necessidade de disponibilizar pontos de recarga tem sido incentivada pelas operadoras de celulares. O SCOPE contribui com este “movimento” provendo facilidades para esse tipo de serviço.

Configurando a recarga de celular

Para a Recarga de Celular devem ser efetuadas as configurações a seguir:

- ScopeCNF: ver no documento ‘Scope – Manual de Instalação e Configuração.doc’ cadastrando os atributos para a Recarga de Celular.
- ScopelNI: O tamanho mínimo de dígitos permitidos para serem lidos pelo PIN-Pad deve ser configurado como descrito no item “Configuração do arquivo scope.ini” seção “Sessão [PINPAD]” deste documento.

Processando a recarga de celular

Apesar de o processamento seguir de maneira similar às outras transações, nesta modalidade existem códigos de coleta específicos. Durante a iteração do processamento da transação de recarga de celular, a aplicação de

PDV deverá estar preparada para receber, da função [ScopeStatus\(\)](#), no caso de uso da interface Coleta, os dois códigos de coleta abaixo, além de outros comuns a outras transações:

- Código 64624 (0xFC70): neste momento, a aplicação deverá recuperar do SCOPE a lista de operadoras disponíveis;
- Código 64558 (0xFC2E): este código representa que a aplicação deve obter a lista de valores disponíveis de recarga.

Iniciando a transação de recarga

O início desta transação é dado pela chamada à função [ScopeRecargaCelular\(\)](#). Esta transação como todas as outras, deve estar numa sessão de TEF (ver [Sessão de transação](#)).

Protótipo

```
LONG EXPORT ScopeRecargaCelular (void)
```

Parâmetros

Não há parâmetros.

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFC02	64514	Imprime Cupom
0xFC2E	64558	Coleta valor da recarga de celular pré-pago
0xFC2F	64559	Coleta código da localidade do telefone
0xFC30	64560	Coleta número do telefone
0xFC70	64624	Coleta a operadora de recarga de celular pré-pago
0xFC90	64656	Coleta o DDD no PINPad
0xFCB5	64693	Coleta redigitação do DDD
0xFCBC	64700	Coleta DDD + Telefone no PIN Pad
0xFCBD	64701	Redigita DDD + Telefone no PIN Pad
0xFCF0	64752	Decide pagamento com cartão ou dinheiro

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida

OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
// abre sessão
...
retorno = ScopeRecargaCelular();
...
// processa a transação
...
// fecha a sessão
...

```

Obtendo operadoras disponíveis

Recebendo o código de coleta 64624 da função [ScopeStatus\(\)](#), a aplicação deverá obter as operadoras disponíveis, através da função ScopeRecuperaOperadorasRecCel(). A lista disponibilizada no segundo parâmetro da função contém o formato abaixo:

Posição	Formato	Descrição
01 até 02	Short	Quantidade de operadoras disponíveis
03	Byte	Código da operadora 1
04 até 25	String	Nome da operadora 1
26	Byte	Código da operadora 2
27 até 48	String	Nome da operadora 2
...
$Pos_n = (n - 1) \times 23 + 3$	Byte	Código da operadora n
$(Pos_n + 1)$ até $(Pos_n + 21)$	String	Nome da operadora n

Exemplo: Quando houver 5 operadoras, as posições dos códigos de cada operadora no buffer serão:

$$Pos_1 = (1 - 1) \times 23 + 3 = 3$$

$$Pos_2 = (2 - 1) \times 23 + 3 = 26$$

$$Pos_3 = (3 - 1) \times 23 + 3 = 49$$

$$Pos_4 = (4 - 1) \times 23 + 3 = 72$$

$$Pos_5 = (5 - 1) \times 23 + 3 = 95$$

Portanto, codificando de maneira simples em linguagem C, temos:

```

...
short *qtd_op;
char buffer[2002];
int operadora, ind_op;
long retorno;

```

```

...
retorno = ScopeRecuperaOperadorasRecCel (2, buffer, sizeof(buffer));
if(retorno == 0)
{
    qtd_op = (short *) buffer;
    for(operadora = 0; operadora < *qtd_op; operadora++)
    {
        ind_op = (operadora - 1) * 23 + 3;
        printf("\nCod %d - Oper. %.21s",
               (int) buffer[ind_op], buffer[ind_op+1]);
    }
}
...

```

No entanto, de outra maneira mais iterativa e sem utilizar fórmulas matemáticas, podemos obter a lista de operadoras como segue no exemplo abaixo:

```

...
short *qtd_op;
char buffer[2002];
int operadora, ind_op;
long retorno;
...

retorno = ScopeRecuperaOperadorasRecCel (2, buffer, sizeof(buffer));
if(retorno == 0)
{
    qtd_op = (short *) buffer;
ind_op = 3;
    for(operadora = 0; operadora < *qtd_op; operadora++)
    {
        printf("\nCod %d - Oper. %.21s",
               (int) buffer[ind_op], buffer[ind_op+1]);
ind_op = ind_op + 23;
    }
}
...

```

Estruturas de apoio

Aplicações escritas em linguagem C podem usufruir de estruturas definidas no arquivo de cabeçalho *ScopeApi.h*. O buffer recebido contendo a lista de operadoras resume-se na estrutura abaixo:

<pre> typedef struct {</pre>

```

short NumOperCel;
char OperCel[2000];
} stREC_CEL_OPERADORAS, *ptREC_CEL_OPERADORAS;
```

onde:

- NumOperCel: número de operadoras de celular retornadas nesta transação;
- OperCel: tabela de operadoras.

No entanto, o membro OperCel da estrutura acima contém a lista desejada e para percorrê-la pode-se utilizar a estrutura:

```

typedef struct
{
    unsigned short CodOperCel;
    char NomeOperCel[TAM_NOME_OP];
} stREC_CEL_ID_OPERADORA_M3, *ptREC_CEL_ID_OPERADORA_M3;
```

onde:

CodOperCel: código, da operadora, que será escolhido pelo operador e devolvido para o SCOPE;

NomeOperCel: nome da operadora representada por uma string.

NOTA: A estrutura anterior stREC_CEL_ID_OPERADORA está obsoleta e foi mantida apenas por compatibilidade com o legado. A nova estrutura stREC_CEL_ID_OPERADORA_M3 permite que o campo CodOperCel seja maior que 255, caso das bandeiras 380 – VERO BANRISUL e 527 – BRADESCO CELULAR. Portanto, para uso da Recarga de Celular das redes VEROBANRISUL R17 e BRADESCO a estrutura stREC_CEL_ID_OPERADORA_M3 (Modelo 3) é mandatória.

Reescrevendo o código exemplificado acima, obtemos:

```

int i;
stREC_CEL_OPERADORAS ListaOper;
stREC_CEL_ID_OPERADORA_M3 Oper;
long retorno;
...
retorno = ScopeRecuperaOperadorasRecCel(REC_CEL_OPERADORAS_MODELO_3,
                                         (char *)&ListaOper, sizeof(ListaOper));
if(retorno == 0)
{
    printf("\n\n LISTA DE OPERADORAS:\n\n");
    /* Exibe as operadoras */
    for(i = 0; i < (int)ListaOper.NumOperCel; i++)
    {
        // inicializa estruturas
        memcpy(&Oper, &ListaOper.OperCel[i*sizeof(stREC_CEL_ID_OPERADORA_M3)],
               sizeof(stREC_CEL_ID_OPERADORA_M3));
```

```

    printf("\nCod %d - Oper. %.2ls",
          (int)buffer[ind_op], buffer[ind_op+1]);
}
}

```

Protótipo

LONG EXPORT ScopeRecuperaOperadorasRecCel (BYTE TpTab, char *buffer, WORD TamBuf)

Parâmetros

[in]	BYTE	TpTab	Informa o formato da tabela que a aplicação deseja receber a lista de operadoras.
[out]	String	Buffer	Buffer com a lista de operadoras.
[in]	WORD	TamBuf	Tamanho do buffer anterior, previamente alocado, que deve ser maior ou igual a 2002 bytes.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido

Exemplo

```

...
stREC_CEL_OPERADORAS ListaOper;
...

// obtém status do SCOPE
switch(status_scope)
{
...
case 64624:
    ScopeRecuperaOperadorasRecCel(2, (char *)&ListaOper, sizeof(buffer));
    ApresentaListaAoOperador(ListaOper);
    // Obtém código da operadora escolhido pelo cliente
    break;
...
}

// passa a informação coletada para o SCOPE
...

```

Obtendo valores de recarga

Com o status do SCOPE igual a 64558, a aplicação deverá informar as opções de recarga disponíveis para a operadora escolhida, que são disponibilizadas pela função ScopeRecuperaValoresRecCel() de maneira similar à função que recupera as operadoras. A estrutura do buffer recebido é a seguinte:

1byte	Um caractere representando o tipo de valor.								
	<table border="1"> <thead> <tr> <th>Valor (ASCII)</th><th>Significado</th></tr> </thead> <tbody> <tr> <td>'V'</td><td>Valor variável</td></tr> <tr> <td>'F'</td><td>Valor Fixo</td></tr> <tr> <td>'T'</td><td>Todos valores</td></tr> </tbody> </table>	Valor (ASCII)	Significado	'V'	Valor variável	'F'	Valor Fixo	'T'	Todos valores
Valor (ASCII)	Significado								
'V'	Valor variável								
'F'	Valor Fixo								
'T'	Todos valores								
12 bytes	Valor mínimo representado por string								
12 bytes	Valor máximo representado por string								
1byte	Total de valores fixos da tabela								
12 bytes	Valor fixo 1 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 1								
12 bytes	Custo da recarga para o valor 1								
12 bytes	Valor fixo 2 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 2								
12 bytes	Custo da recarga para o valor 2								
12 bytes	Valor fixo 3 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 3								
12 bytes	Custo da recarga para o valor 3								
12 bytes	Valor fixo 4 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 4								
12 bytes	Custo da recarga para o valor 4								
12 bytes	Valor fixo 5 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 5								
12 bytes	Custo da recarga para o valor 5								
12 bytes	Valor fixo 6 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 6								
12 bytes	Custo da recarga para o valor 6								
12 bytes	Valor fixo 7 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 7								
12 bytes	Custo da recarga para o valor 7								
12 bytes	Valor fixo 8 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 8								
12 bytes	Custo da recarga para o valor 8								
12 bytes	Valor fixo 9 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 9								
12 bytes	Custo da recarga para o valor 9								
12 bytes	Valor fixo 10 da recarga, com 2 casas decimais								
12 bytes	Bônus da recarga para o valor 10								
12 bytes	Custo da recarga para o valor 10								
41 bytes	Mensagem promocional a ser exibida com os valores								

Estruturas de apoio

Definidas no arquivo de cabeçalho `ScopeApi.h`, estão as estruturas abaixo, que são utilizadas para o recebimento do buffer de valores. A primeira delas:

```

typedef struct
{
    char TipoValor;
    char ValorMinimo[12];
    char ValorMaximo[12];
    char Totvalor;
    stREC_CEL_VALOR TabValores[10];
    char MsgPromocional[41];
} stREC_CEL_VALORES, *ptREC_CEL_VALORES;

```

, onde:

- TipoValor: caractere que representa o tipo de valor permitido para a operadora e que define quais valores serão exibidos. O domínio de valores para este campo é:
 - 'V' – valor variável entre o que está indicado no campo ValorMinimo e ValorMaximo desta mesma estrutura
 - 'F' – valores fixos que estão no campo TabValores
 - 'T' – disponível tanto o valor variável quanto o fixo
- ValorMinimo: este campo é uma cadeia com 12 caracteres numéricos que representam o valor mínimo do intervalo de valores variáveis com a vírgula implícita (exemplo: caso o dado neste campo seja 000000001000, então ele está representando o valor R\$10,00);
- ValorMaximo: este campo é uma cadeia com 12 caracteres numéricos que representam o valor máximo do intervalo de valores variáveis com a vírgula implícita (exemplo: caso o dado neste campo seja 000000001500, então ele está representando o valor R\$15,00);
- Totvalor: este campo é binário e indica a quantidade de valores fixos que estão no campo TabValores;
- TabValores: cadeia de registros, descrita abaixo, com todos os valores fixos disponíveis para a recarga desta transação que, embora seja fixo com 10 registros, deve-se sempre observar a quantidade válida representada pelo campo anterior;
- MsgPromocional: string contendo uma mensagem promocional a ser exibida com os valores de recarga.

CUIDADO: como a estrutura de valores é sempre fixa com dez registros de valores e por esses vários valores do domínio do campo de tipo de valor, é extremamente recomendável que a aplicação sempre verifique a quantidade de valores fixos, informada pelo campo TotValor, disponíveis.

Como se pode notar pela definição, o campo TabValores é do tipo da estrutura stREC_CEL_VALOR, descrita abaixo.

```

typedef struct
{
    char Valor[12];
    char Bonus[12];
    char Custo[12];
} stREC_CEL_VALOR, *ptREC_CEL_VALOR;

```

- Valor: este campo é uma cadeia com 12 caracteres numéricos que representam um valor fixo, para escolha do cliente, com a vírgula implícita (exemplo: caso o dado neste campo seja 000000001500, então ele está representando o valor R\$15,00);
- Bônus: este campo é uma cadeia com 12 caracteres numéricos que representam o bônus que o cliente ganha ao escolher o valor fixo deste registro com a vírgula implícita (exemplo: caso o dado neste campo seja 000000010000, então ele está representando o valor R\$100,00);
- Custo: este campo é uma cadeia com 12 caracteres numéricos que representam o custo da recarga ao escolher o valor fixo deste registro com a vírgula implícita (exemplo: caso o dado neste campo seja 000000005000, então ele está representando o valor R\$50,00);

```
typedef struct
{
    char TipoValor;
    char ValorMinimo[10];
    char ValorMaximo[10];
    char Totvalor;
    stREC_CEL_VALOR TabValores[10];
    char MsgPromocional[41];

    char TotFaixaValores;
    stREC_CEL_FAIXA_VALORES_2 TabFaixaValores[10];
} stREC_CEL_VALORES_MODELO_4, *ptREC_CEL_VALORES_MODELO_4;
```

A rede GWCel permite que o cliente faça uma recarga, utilizando uma faixa de valores, ao invés de valores pré-definidos. A maioria dos membros dessa estrutura são os mesmos da stREC_CEL_VALORES, sendo adicionais:

- TotFaixaValores: indica a quantidade total de faixas de valores disponíveis;
- TabFaixaValores: cadeia de registros, descrita abaixo, com todos as faixas de valores disponíveis para a recarga desta transação que, embora seja fixo com 10 registros, deve-se sempre observar a quantidade válida representada pelo campo anterior;

Existe uma diferença na estrutura dos dados enviados pela GWCel dependendo a versão de sua especificação.

Para a rede GWCel versão 003 a estrutura de Valores de Faixa é a seguinte:

```
typedef struct
{
    char ValorMin[10];
    char ValorMax[10];
} stREC_CEL_FAIXA_VALORES, *ptREC_CEL_FAIXA_VALORES;
```

- ValorMin: indica o valor mínimo aceitável para a recarga;

- ValorMax: indica o valor máximo aceitável para a recarga.

Para a Gwcel versão 005, as estruturas de Valores de Faixa também possuem os campos Bônus e Custo, antes só presentes na estrutura de valores fixos:

```
typedef struct
{
    char ValorMin[12];
    char ValorMax[12];
    char Bonus[12];
    char Custo[12];

} stREC_CEL_FAIXA_VALORES_2, *ptREC_CEL_FAIXA_VALORES_2;
```

- ValorMin: indica o valor mínimo aceitável para a recarga;
- ValorMax: indica o valor máximo aceitável para a recarga.
- Bonus: Indica o valor do bônus que o cliente receberá ao escolher a faixa.
- Custo: Indica o custo da recarga.

Protótipo

```
LONG EXPORT ScopeRecuperaValoresRecCel (BYTE TpTab, char *buffer, WORD TamBuf)
```

Parâmetros

[in]	BYTE	TpTab	Informa o formato da tabela que a aplicação deseja receber a lista de operadoras. Aceita apenas o valor 2.
[out]	String	Buffer	Buffer com a lista de valores de recarga disponíveis.
[in]	WORD	TamBuf	Tamanho do array buffer (segundo parâmetro) alocado que deve ser igual a 427 bytes.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido

Exemplo

```
...
stREC_CEL_VALORES tabVal;
...
// obtém status do SCOPE
switch(status_scope)
```

```
{
...
case 64558:
    ScopeRecuperaValoresRecCel(2, (char *)&tabVal, sizeof(buffer));
ApresentaValoresAoOperador(tabVal);
    // Obtém o valor de recarga escolhido pelo cliente
    break;
...
}
// passa a informação coletada para o SCOPE
...
```

CUIDADO: os campos de valores representados nas estruturas de recarga não são *strings* terminadas com o caractere *nulo*, mas são cadeias de caracteres de tamanho fixo igual a 12 com zeros à esquerda.

Funções de consulta para recarga de celular

O Scope Client não disponibilizava funções de consulta a planos de recarga de celular, úteis para sistemas de retaguarda que desejavam vender pacotes de recarga antecipadamente dentro de um pré-pedido. Para contornar isso, foram adicionadas duas funções, uma para retornar as operadoras e outra para retornar os valores disponíveis para recarga de uma operadora de uma localidade específica.

Obtendo operadoras disponíveis fora do fluxo de recarga

Para obter as operadoras disponíveis, deve ser feita uma chamada a função [ScopeObtemOperadorasRecCelOffTEF\(\)](#), ela retornará um buffer com as operadoras disponíveis.

Protótipo

LONG EXPORT ScopeObtemOperadorasRecCelOffTEF (BYTE _TipoTabela, char *_Buffer, WORD _TamBuffer)

Parâmetros

[in]	BYTE	_TipoTabela	Informa o formato da tabela que a aplicação deseja receber a lista de operadoras.
[out]	String	_Buffer	Buffer com a lista de operadoras.
[in]	WORD	_TamBuffer	Tamanho do buffer anterior, previamente alocado, que deve ser maior ou igual a 2002 bytes.

Retorno

Ver [tabela de código de retorno](#).

Exemplo

```
...
LONG RC;
stREC_CEL_OPERADORAS      ListaOper      = {0};
```

```

stREC_CEL_ID_OPERADORA_M3 OPER;
...
// Obtem as operadoras.
RC = ScopeObtemOperadorasRecCelOffTEF(REC_CEL_OPERADORAS_MODELO_3,(P_CHAR)
&ListaOper, sizeof(ListaOper);
if(RC != RCS_SUCESSO)
{
    return RC;
}
...
// exibe as operadoras

```

Obtendo valores de recarga disponíveis fora do fluxo de recarga

Para obter os valores disponíveis para recarga, deve ser feita uma chamada a função [ScopeRecuperaValoresRecCelOffTEF\(\)](#) ela retornará um buffer com a lista de valores disponíveis para uma operadora de uma localidade específica.

Protótipo

LONG EXPORT ScopeRecuperaValoresRecCelOffTEF(BYTE _TipoTabela, char *_Buffer, WORD _TamBuffer, char* _CodOperadora, char* _CodLocalidade)

Parâmetros

[in]	BYTE	_TipoTabela	Informa o formato da tabela que a aplicação deseja receber a lista de operadoras. Aceita apenas o valor 2.
[out]	String	_Buffer	Buffer com a lista de valores de recarga disponíveis.
[in]	WORD	_TamBuffer	Tamanho do array buffer (segundo parâmetro) alocado que deve ser igual a 427 bytes.
[in]	String	_CodOperadora	Código da bandeira da operadora
[in]	String	_CodLocalidade	Código da localidade (DDD)

Retorno

Ver [tabela de código de retorno](#).

Exemplo

```

...
LONG RC;
char codOperadora[3 + 1];
char codLocalidade[2 + 1];
stREC_CEL_VALORES ListaValores = {0};
...
```

```
// Entra com o código da operadora e a localidade

RC=ScopeRecuperaValoresRecCelOffTEF((BYTE)REC_CEL_VALORES_MODELO_2,(P_CHAR)
&ListaValores, sizeof(ListaValores), codOperadora, codLocalidade);
if ( RC == RCS_SUCESSO || RC == RCS_ACQUIRER_OFF)
{
    // Exibe os valores
}
...
...
```

Recarga de Celular para a rede VERO BANRISUL R17

A Recarga de Celular da Rede VEROBANRISUL R17 (Release 2017) possui algumas particularidades que a diferem das outras redes:

1. O pagamento também pode ser feito através do cartão BANRICOMPRA. Portanto, durante o fluxo de pagamento, o SCOPE irá perguntar “Pag. com Cartão? 1-Sim 0-Não”, cujo estado de coleta é TC_DECIDE_CARTAO_OU_DINHEIRO (0xFCF0 = 64752). Caso a resposta seja afirmativa, o SCOPE irá solicitar “Insira ou Passe o Cartão”;
2. As listas de “Operadoras de Recarga” e “Valores Disponíveis” são transações “online” e possuem formatos próprios, conforme estruturas descritas mais abaixo;
3. Para manter a compatibilidade do fluxo operacional com outras redes de Recarga de Celular, a própria rede VEROBANRISUL é considerada pelo SCOPE como uma Operadora/Bandeira = 380 – VERO BANRISUL. Assim, o fluxo de coleta da Recarga de Celular foi mantido o mesmo até o ponto em que o PDV solicita a lista de Operadoras disponíveis. Caso disponível, será retornada a opção 380 – VEROBANRISUL na lista. Se selecionada (380), os próximos passos de coleta serão: TC_COLETA_OPERADORA_ONLINE (0xFCEE = 64750) para obter a lista de Operadoras Online e TC_COLETA_VALOR_RECARGA_ONLINE (0xFCEF = 64751) para obter a lista de Valores Online.
4. O módulo TstColeca do SCOPE foi atualizado para trabalhar com a Recarga de Celular da rede VEROBANRISUL R17. Use-o como exemplo de codificação.

Exemplo de Recarga de Celular da rede VEROBANRISUL R17 através do TstColeta:

Passo	Tela do módulo de exemplo TstColeta
-------	-------------------------------------

1	<pre>+----- TELA DO CLIENTE ----- Recarga Celular Aguarde... +----- TECLADO DO OPERADOR ----- Recarga Celular Operadora ? +-----</pre> <p>LISTA DE OPERADORAS:</p> <p>Numero de operadoras de celular = [1] Codigo da operadora = [380] Nome = [VERO BANRISUL]</p> <p>:Codigo da operadora ? 380_</p>																				
2	<pre>+----- TELA DO CLIENTE ----- Recarga Celular Aguarde... +----- TECLADO DO OPERADOR ----- Recarga Celular Pag. com Cartao? 1-Sim 0-Nao +-----</pre> <p>:Coleta ? 1</p>																				
3	<pre>+----- TELA DO CLIENTE ----- Recarga Celular Aguarde... +----- TECLADO DO OPERADOR ----- Recarga Celular Insira ou Passe o Cartao +-----</pre>																				
4	<pre>+----- TELA DO CLIENTE ----- Recarga Celular Aguarde... +----- TECLADO DO OPERADOR ----- Recarga Celular Solicite DDD + Numero do Telefone +-----</pre>																				
5	<pre>+----- TELA DO CLIENTE ----- Recarga Celular Aguarde... +----- TECLADO DO OPERADOR ----- Recarga Celular Operadora VEROBANRISUL? +-----</pre> <p>LISTA DE OPERADORAS:</p> <table border="0"> <tr> <td>Idx</td> <td>Codigo</td> <td>Nome</td> <td></td> </tr> <tr> <td>[1]</td> <td>[00601000000]</td> <td>[CLARO DIGITAL - RS]</td> <td>]</td> </tr> <tr> <td>[2]</td> <td>[01004000000]</td> <td>[TIM-CENTRO SUL]</td> <td>]</td> </tr> <tr> <td>[3]</td> <td>[01201000000]</td> <td>[VIVO - RS]</td> <td>]</td> </tr> <tr> <td>[4]</td> <td>[01325000000]</td> <td>[OI RS]</td> <td>]</td> </tr> </table> <p>:Idx da operadora online ? 3_</p>	Idx	Codigo	Nome		[1]	[00601000000]	[CLARO DIGITAL - RS]]	[2]	[01004000000]	[TIM-CENTRO SUL]]	[3]	[01201000000]	[VIVO - RS]]	[4]	[01325000000]	[OI RS]]
Idx	Codigo	Nome																			
[1]	[00601000000]	[CLARO DIGITAL - RS]]																		
[2]	[01004000000]	[TIM-CENTRO SUL]]																		
[3]	[01201000000]	[VIVO - RS]]																		
[4]	[01325000000]	[OI RS]]																		

6	<pre>+----- TELA DO CLIENTE ----- Recarga Celular Aguarde... +----- TECLADO DO OPERADOR ----- Recarga Celular Valor da Recarga ? +-----</pre> <p>LISTA DE VALORES (Qty=6):</p> <table border="0"> <tr><td>ValorRecarga</td><td>Bonus</td><td>Informacao</td></tr> <tr><td>[0000001000]</td><td>[0000000000]</td><td>[0000000000]</td></tr> <tr><td>[0000002000]</td><td>[0000000000]</td><td>[0000000000]</td></tr> <tr><td>[0000003000]</td><td>[0000000000]</td><td>[0000000000]</td></tr> <tr><td>[0000004000]</td><td>[0000000000]</td><td>[0000000000]</td></tr> <tr><td>[0000005000]</td><td>[0000000000]</td><td>[0000000000]</td></tr> <tr><td>[0000006000]</td><td>[0000000000]</td><td>[0000000000]</td></tr> </table> <p>Valor Variavel Min = [0000000100] Valor Variavel Max = [0000000900]</p> <p>:Valor de recarga ? 1000</p>	ValorRecarga	Bonus	Informacao	[0000001000]	[0000000000]	[0000000000]	[0000002000]	[0000000000]	[0000000000]	[0000003000]	[0000000000]	[0000000000]	[0000004000]	[0000000000]	[0000000000]	[0000005000]	[0000000000]	[0000000000]	[0000006000]	[0000000000]	[0000000000]
ValorRecarga	Bonus	Informacao																				
[0000001000]	[0000000000]	[0000000000]																				
[0000002000]	[0000000000]	[0000000000]																				
[0000003000]	[0000000000]	[0000000000]																				
[0000004000]	[0000000000]	[0000000000]																				
[0000005000]	[0000000000]	[0000000000]																				
[0000006000]	[0000000000]	[0000000000]																				
7	<p>VERO - BANRICOMPRA RECARGA DE CELULAR - via cliente</p> <p>EMPRESA 0001 FILIAL 0001 CNPJ: 19.973.770/0004-91 SÃO PAULO 041003500000100 IT003</p> <p>06/02/2019 - 13:03:35 NSU: 00003044 EM DINHEIRO VALOR: 10,00 (51) 99999-9999 BONUS: 0,00 Mensagem de Rodape linha 1 Mensagem de Rodape linha 2 Mensagem de Rodape linha 3</p> <p>ESTE DEMONSTRATIVO E VALIDO COMO COMPROVANTE DE PAGAMENTO, SENDO OS DADOS INFORMADOS DE RESPONSABILIDADE DO CLIENTE.</p> <p>[Pressione uma tecla para continuar...]</p>																					
8	<pre>+----- TELA DO CLIENTE ----- Recarga Celular AUTORIZADO +----- TECLADO DO OPERADOR ----- Recarga Celular AUTORIZADO Controle 02600443008 +-----</pre> <p>[<P>roximo]</p>																					

Estados e funções a serem usadas em cada passo:

Passo	Estado	Codificação
1	OxFC70 (64752)	Função: ScopeObtemOperadorasRecCel Tipo de tabela: REC_CEL_OPERADORAS_MODELO_3 Estrutura: stREC_CEL_ID_OPERADORA_M3

		Resposta esperada pelo SCOPE: Operadora: 380 = VERO BANRISUL
5	0xFCEE (64750)	<p>Função: ScopeRecuperaOperadorasRecCel</p> <p>Tipo de tabela: REC_CEL_OPERADORAS_MODELO_4</p> <p>Estrutura: stListaOperadorasCelularBanrisul</p> <p>Resposta esperada pelo SCOPE: Código da operadora</p>
6	0xFCEF (64751)	<p>Função: ScopeRecuperaValoresRecCel</p> <p>Tipo de tabela: REC_CEL_VALORES_MODELO_5</p> <p>Estrutura: stListaValoresCelularBanrisul</p> <p>Resposta esperada pelo SCOPE: Valor da recarga</p>

Definições da estruturas de dados específicas da rede VEROBANRISUL R17:

Estrutura de Operadoras Online:

```
#define MAX_OPERADORAS_CELULAR_BANRISUL 12
#define TIPO_OPERADORAS_CELULAR_BANRISUL 'A'

typedef struct
{
    char Tipo;
    char CodRede[3];
    char CodBandeira[3];
} stOperadoraCelularOnlineHeader, *LPOperadoraCelularOnlineHeader;

typedef struct
{
    char Codigo[11];
    char Nome[38];
} stOperadoraCelularBanrisul, *LPOperadoraCelularBanrisul;

typedef struct
{

```

```

stOperadoraCelularOnlineHeader sHeader;
char QtdOperadoras[2];
stOperadoraCelularBanrisul sOperadora[MAX_OPERADORAS_CELULAR_BANRISUL];
}stListaOperadorasCelularBanrisul, *LPListaOperadorasCelularBanrisul;

```

Onde:

- sHeader.Tipo: 'A' (para permitir eventuais evoluções futuras)
- sHeader.CodRede: '099' (VEROBANRISUL)
- sHeader.CodBandeira: '380' (VERO BANRISUL)
- QtdOperadoras: Quantidade de operadoras presentes na lista a seguir (Máximo 12)
- sOperadora.Codigo: Código da operadora (TIM, CLARO, VIVO, etc) retornado de forma online pela rede VEROBANRISUL
- sOperadora.Nome: Nome da operadora (TIM, CLARO, VIVO, etc)

Estrutura de Valores Online:

```

#define MAX_VALORES_CELULAR_BANRISUL 6
#define TIPO_VALORES_CELULAR_BANRISUL 'B'

typedef struct
{
    char Tipo;
    char CodRede[3];
    char CodBandeira[3];
} stOperadoraCelularOnlineHeader, *LPOperadoraCelularOnlineHeader;

typedef struct
{
    charValorRecarga[10];
    charBonus[10];
    charInformacao[10];
} stValorCelularBanrisul, *LPValorCelularBanrisul;

typedef struct
{
    stOperadoraCelularOnlineHeader sHeader;
    charPermiteValorVariavel;
    charValorVariavelMin[10];
    charValorVariavelMax[10];
    charQtdValores[2];
    stValorCelularBanrisul sValor[MAX_VALORES_CELULAR_BANRISUL];
}stListaValoresCelularBanrisul, *LPListaValoresCelularBanrisul;

```

Onde:

- sHeader.Tipo: 'B' (para permitir eventuais evoluções futuras)
- sHeader.CodRede: '099' (VEROBANRISUL)
- sHeader.CodBandeira: '380' (VERO BANRISUL)
- PermiteValorVariavel: Indicador se permite ou não valor variável
- ValorVariavelMin: Valor variável mínimo de recarga
- ValorVariavelMax: Valor variável máximo de recarga
- QtdValores: Quantidade de valores presentes na lista a seguir (Máximo 6)
- sValor.ValorRecarga: Valor de recarga
- sValor.Bonus: Valor do bônus
- sValor.Informacao: Informação para o valor

Recarga de Celular para a rede BRADESCO

A Recarga de Celular da Rede BRADESCO (Release 5.58B) possui algumas particularidades que a diferem das outras redes:

1. As listas de "Operadoras de Recarga" e "Valores Disponíveis" são transações "online" e possuem formatos próprios, conforme estruturas descritas mais abaixo;
2. Para manter a compatibilidade do fluxo operacional com outras redes de Recarga de Celular, a própria rede BRADESCO é considerada pelo SCOPE como uma Operadora/Bandeira = 527 – BRADESCO CELULAR. Assim, o fluxo de coleta da Recarga de Celular foi mantido o mesmo até o ponto em que o PDV solicita a lista de Operadoras disponíveis. Caso disponível, será retornada a opção 527 – BRADESCO CELULAR na lista. Se selecionada (527), os próximos passos de coleta serão: TC_COLETA_OPERADORA_ONLINE (0xFCEE = 64750) para obter a lista de Operadoras Online e TC_COLETA_VALOR_RECARGA_ONLINE (0xFCEF = 64751) para obter a lista de Valores Online.
3. O módulo TstColeca do SCOPE foi atualizado para trabalhar com a Recarga de Celular da rede BRADESCO. Use-o como exemplo de codificação.

Exemplo de Recarga de Celular da rede BRADESCO especificação v5.58B, através do TstColeta:

Passo	Tela do módulo de exemplo TstColeta
-------	-------------------------------------

1	<pre>+----- TELA DO CLIENTE -----+ Recarga Celular Aguarde... +----- TECLADO DO OPERADOR -----+ Recarga Celular Operadora ? +-----+</pre> <p>LISTA DE OPERADORAS:</p> <p>Numero de operadoras de celular = [2] Codigo da operadora = [380] Nome = [VERO BANRISUL] Codigo da operadora = [527] Nome = [BRADESCO CELULAR]</p> <p>:Codigo da operadora ? 527</p>
2	<pre>+----- TELA DO CLIENTE -----+ Recarga Celular Aguarde... +----- TECLADO DO OPERADOR -----+ Recarga Celular Codigo Localidade Telefone ? +-----+</pre> <p>:Coleta ? 11</p>
3	<pre>+----- TELA DO CLIENTE -----+ Recarga Celular Aguarde... +----- TECLADO DO OPERADOR -----+ Recarga Celular Solicite o Numero do Telefone +-----+</pre> <p>:Coleta ? 11987643322</p>

4

```
+----- TELA DO CLIENTE -----+
| Recarga Celular |
| Aguarde... |
+----- TECLADO DO OPERADOR -----+
| Recarga Celular |
| Operadora BRADESCO? |
+-----+
```

LISTA DE OPERADORAS:

Idx	Codigo	UF	Nome
[1]	[11001]	[SP]	[VIVO SAO PAULO]
[2]	[11002]	[SP]	[TIM SAO PAULO]
[3]	[11003]	[SP]	[CLARO SAO PAULO]
[4]	[11004]	[SP]	[OI SAO PAULO]

:Idx da operadora online ? 2

5

```
+----- TELA DO CLIENTE -----+
| Recarga Celular |
| Aguarde... |
+----- TECLADO DO OPERADOR -----+
| Recarga Celular |
| Valor da Recarga ? |
+-----+
```

LISTA DE VALORES (Qtd=7):

ValorRecarga	Informacao
[000000001500]	[VAL 30 DIAS]
[000000002000]	[VAL 30 DIAS + 1GB]
[000000002500]	[VAL 30 DIAS + 1GB]
[000000003000]	[VAL 30 DIAS + 1GB]
[000000003500]	[VAL 90 DIAS + 1GB]
[000000004000]	[VAL 90 DIAS + 2GB]
[000000005000]	[VAL 120 DIAS + 3GB]

Valor Variavel Min = [000000001000]
 Valor Variavel Max = [000000005000]

Valor Tarifa = [000000000000]

:Valor de recarga ? 2500

6	<p>Correspondente do Banco Bradesco S.A. Comprovante de Recarga de Celular Data: 30/09/2022 Hora de Brasilia: 11:54</p> <p>TIM SAO PAULO Codigo: 11002</p> <p>Valor da Recarga: 25,00 Valor da Tarifa: 0,00 Telefone: (11) 987643322</p> <p>VAL 30 DIAS + 1GB</p> <p>D E M O N S T R A C A O Transacao sem validade. Autorizacao gerada por simulador.</p> <p>CONTROLE 09000661030 NCR BRASIL SCOPE</p>	
7	<pre>+----- TELA DO CLIENTE -----+ Recarga Celular AUTORIZADO +----- TECLADO DO OPERADOR -----+ Operador AUTORIZADO Controle 02600443009 +-----+ [<P>proxima]</pre>	

Estados e funções a serem usadas em cada passo:

Passo	Estado	Codificação
1	0xFC70 (64752)	<p>Função: ScopeObtemOperadorasRecCel</p> <p>Tipo de tabela: REC_CEL_OPERADORAS_MODELO_3</p> <p>Estrutura: stREC_CEL_ID_OPERADORA_M3</p> <p>Resposta esperada pelo SCOPE: Operadora: 527 = BRADESCO CELULAR</p>
5	0xFCEE (64750)	<p>Função: ScopeRecuperaOperadorasRecCel</p> <p>Tipo de tabela: REC_CEL_OPERADORAS_MODELO_5</p> <p>Estrutura: stListaOperadorasCelularBradesco</p> <p>Resposta esperada pelo SCOPE: Código da operadora</p>
6	0xFCEF (64751)	<p>Função: ScopeRecuperaValoresRecCel</p>

		<p>Tipo de tabela: REC_CEL_VALORES_MODELO_6</p> <p>Estrutura: stListaValoresCelularBradesco</p> <p>Resposta esperada pelo SCOPE: Valor da recarga</p>
--	--	---

Definições da estruturas de dados específicas da rede BRADESCO especificação v5.58B:

Estrutura de Operadoras Online:

```
#define MAX_OPERADORAS_CELULAR_BRADESCO 15
#define TIPO_OPERADORAS_CELULAR_BRADESCO 'C'

typedef struct
{
    char Tipo;
    char CodRede[3];
    char CodBandeira[3];
} stOperadoraCelularOnlineHeader, *LPOperadoraCelularOnlineHeader;

typedef struct
{
    char Codigo[5];
    char Nome[50];
    char UF[2];
} stOperadoraCelularBradesco, *LPOperadoraCelularBradesco;

typedef struct
{
    stOperadoraCelularOnlineHeader sHeader;
    char QtdOperadoras[2];
    stOperadoraCelularBradesco sOperadora[MAX_OPERADORAS_CELULAR_BRADESCO];
} stListaOperadorasCelularBradesco, *LPListaOperadorasCelularBradesco;
```

Onde:

- sHeader.Tipo: 'C (para permitir eventuais evoluções futuras)
- sHeader.CodRede: '004' (BRADESCO)
- sHeader.CodBandeira: '527' (BRADESCO CELULAR)
- QtdOperadoras: Quantidade de operadoras presentes na lista a seguir (Máximo 15)
- sOperadora.Codigo: Código da operadora (TIM, CLARO, VIVO, etc) retornado de forma online pela rede BRADESCO
- sOperadora.Nome: Nome da operadora (TIM, CLARO, VIVO, etc)

Estrutura de Valores Online:

```
#define MAX_VALORES_CELULAR_BRADESCO 20
#define TIPO_VALORES_CELULAR_BRADESCO 'D'

typedef struct
{
    char Tipo;
    char CodRede[3];
    char CodBandeira[3];
} stOperadoraCelularOnlineHeader, *LPOperadoraCelularOnlineHeader;

typedef struct
{
    Char ValorRecarga[12];
    Char Informacao[20];
} stValorCelularBradesco, *LPValorCelularBradesco;

typedef struct
{
    stOperadoraCelularOnlineHeader sHeader;
    char PermiteValorVariavel;
    char ValorVariavelMin[12];
    char ValorVariavelMax[12];
    char ValorTarifa[12];
    char QtdValores[2];
    stValorCelularBradesco sValor[MAX_VALORES_CELULAR_BRADESCO];
} stListaValoresCelularBradesco, *LPListaValoresCelularBradesco;
```

Onde:

- sHeader.Tipo: 'D' (para permitir eventuais evoluções futuras)
- sHeader.CodRede: '004' (BRADESCO)
- sHeader.CodBandeira: '527' (BRADESCO CELULAR)
- PermiteValorVariavel: Indicador se permite ou não valor variável
- ValorVariavelMin: Valor variável mínimo de recarga
- ValorVariavelMax: Valor variável máximo de recarga
- ValorTarifa: Valor da tarifa, a ser somado no valor da transação. Pode ser zero.
- QtdValores: Quantidade de valores presentes na lista a seguir (Máximo 20)
- sValor.ValorRecarga: Valor de recarga
- sValor.Informacao: Informação para o valor

Recarga de Celular para as redes VERO BANRISUL e BRADESCO em “modo compatibilidade”

O SCOPE também pode operar o fluxo de Recarga de Celular da VERO BANRISUL e BRADESCO de forma a manter a compatibilidade com o SIAC (e eventualmente outras Automações Comerciais), no sentido de manter-se os mesmos estados de coleta (TCs) e formados já utilizados pelas Recargas de Celular mais antigas.

Premissas

1. Se o lojista habilitou no SCOPE uma única Operadora de Recarga de Celular, dentre as que trabalham com Consulta de Operadoras e Valores de forma online, a saber:
 - 380 = VERO BANRISUL
 - 527 = BRADESCO CELULAR

O SCOPE irá fazer a seleção automática e seguir com a Consulta de Operadoras/Valores de forma online com a rede VERO BANRISUL ou BRADESCO.

Opcionalmente, se configurada ambas as Recargas (VERO BANRISUL e BRADESCO), pode ser usada uma configuração no SCOPE.INI do PDV, conforme abaixo:

```
[ScopeAPI]
FixaOperRecarga=NNN
```

Sendo: NNN = 380 (VERO BANRISUL) ou 527 (BRADESCO) ou N (não deve fixar operadora)

2. Os TCs usados para coleta da Operadora/Valor online serão traduzidos para os TCs usados numa Recarga convencional, conforme tabela abaixo.

DE	PARA
TC_COLETA_OPERADORA_ONLINE 0xFCEE	TC_COLETA_OPERADORA 0xFC70

TC_COLETA_VALOR_RECARGA_ONLINE 0xFCEF	TC_COLETA_VALOR_RECARGA 0xFC2E

3. Os formatos solicitados pelo SIAC ou outra AC serão convertidos dos formatos utilizados pela VERO BANRISUL e BRADESCO para o formato solicitado. Devido à limitação dos formatos antigos, podem ocorrer tratamentos de conversão, com possibilidade de perda de informação, conforme mapeamento abaixo:

REC_CEL_OPERADORAS_MODELO_2			
Código da Operadora			
Formato	VERO BANRISUL	BRADESCO	Tradução
unsigned char (limite = 255)	char[11]	char[5]	Traduz para 'n' (1, 2, 3...) Após seleção do 'n', SCOPE obtém o código original VERO BANRISUL ou BRADESCO
Nome da Operadora			
Formato	VERO BANRISUL	BRADESCO	Tradução
char[20]	char[38]	char[50]	Trunca em 20
Estado (Unidade Federativa)			
Formato	VERO BANRISUL	BRADESCO	Tradução
Não existe	Não existe	UF[2]	Informação perdida

REC_CEL_OPERADORAS_MODELO_3			
Código da Operadora			
Formato	VERO BANRISUL	BRADESCO	Tradução
unsigned short (limite = 65535)	char[11]	char[5]	Traduz para 'n' (1, 2, 3...) Após seleção do 'n', SCOPE obtém o código original VERO BANRISUL ou BRADESCO
Nome da Operadora			
Formato	VERO BANRISUL	BRADESCO	Tradução
char[20]	char[38]	char[50]	Trunca em 20
Estado (Unidade Federativa)			
Formato	VERO BANRISUL	BRADESCO	Tradução
Não existe	Não existe	UF[2]	Informação perdida

REC_CEL_VALORES_MODELO_2			
Quantidade máxima de valores			
Formato	VERO BANRISUL	BRADESCO	Tradução
10	6	20	Limita em 10
Campos			
Formato	VERO BANRISUL	BRADESCO	Tradução
ValorMinimo[12]	ValorMinimo[10]	ValorMinimo[12]	Zeros à esquerda (VERO BANRISUL)
ValorMaximo[12]	ValorMaximo[10]	ValorMaximo[12]	Zeros à esquerda (VERO BANRISUL)
MsgPromocional[40]	Não existe	Não existe	String vazia
Valor[12]	Valor[10]	Valor[12]	Zeros à esquerda (VERO BANRISUL)
Bonus[12]	Bonus[10]	Não existe	Zeros à esquerda (VERO BANRISUL) Zeros (BRADESCO)
Custo[12]	Não existe	Custo[12]	Zeros (VERO BANRISUL)
Não existe	Informação[10] (por valor)	Informação[20] (por valor)	Informação perdida

Os formatos **REC_CEL_VALORES_MODELO_3** e **REC_CEL_VALORES_MODELO_4** são idênticos ao formato **REC_CEL_VALORES_MODELO_2** acima, mas com acréscimo dos campos abaixo:

REC_CEL_VALORES_MODELO_3		
Formato	VERO BANRISUL	BRADESCO
stREC_CEL_FAIXA_VALORES TabFaixaValores[10]	Não existe	Não existe
REC_CEL_VALORES_MODELO_4		
Formato	VERO BANRISUL	BRADESCO
stREC_CEL_FAIXA_VALORES_2 TabFaixaValores[10]	Não existe	Não existe

Estorno de transações

O estorno é uma transação de anulação de outra transação que já está confirmada no lado da autorizadora e pode ser realizada quando ocorrerem erros na digitação (valor, data de agendamento das parcelas, numero de parcelas, etc.) ou desistência da compra por parte do cliente. O SCOPE disponibiliza uma única função para o estorno das diversas transações: `ScopeCancelamento()`. No entanto, nem todas as transações são estornáveis, como por exemplo, transações de consultas. Na tentativa de estorno destas transações, o SCOPE retornará através da função `ScopeStatus()` o código de erro 65286, informando que a transação não é cancelável.

O estorno difere do desfazimento no sentido em que o primeiro é uma nova transação isolada, enquanto o segundo é a finalização de uma transação. Tanto, que existe o desfazimento do estorno, que não permite que ocorra o cancelamento da transação original, mantendo-a com o status de confirmada.

CUIDADO: não é possível o cancelamento de uma transação sem que esta já tenha sido confirmada. Em outras palavras, no SCOPE não se pode cancelar uma transação que ainda está numa sessão de TEFem aberto. Para este cancelamento a transação deve ser desfeita (ver [Encerrando a sessão](#)).

Estornando a transação

Para a realização do estorno, é necessário informar ao SCOPE os dados da transação original, e, em muitas vezes, o número do controle gerado pelo próprio SCOPE na transação.

Protótipo

```
LONG EXPORT ScopeCancelamento (char *Valor, char *TxServico)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```
...
char valor[12 + 1];
char taxa[12 + 1];
...
```

```
// obtém o valor da compra e armazena em 'valor'  
// obtém a taxa de serviço e armazena em 'taxa'  
// abre sessão  
...  
retorno = ScopeCancelamento(valor, taxa);  
...  
// processa o cancelamento  
...  
// fecha a sessão  
...
```

Comprovantes

Comprovantes de transações

Qualquer transação realizada com sucesso e com algum tipo de comprovante para a impressão deve ser obtido pela função ScopeGetCupomEx().

Obtendo os cupons de TEF

Durante o processamento de uma transação, a aplicação deverá esperar pelos códigos de coleta:

- 64582 (0xFC46): imprimir cupom parcial;
- 64514 (0xFC02): imprimir cupom;
- 64539 (0xFC1B): imprimir consulta;
- 64546 (0xFC22): imprime nota promissória;
- 64593 (0xFC51): imprime cupom promocional.

Neste momento, a aplicação chamará a função para obter os cupons da transação.

CUIDADO: quando a aplicação realizar uma consulta CDC no meio de uma TEF de débito, a aplicação não deverá enviar o cupom da consulta para a impressora, mas apenas exibi-lo na tela. No entanto, quando for apenas uma transação de consulta, isto é, foi chamada apenas uma função do tipo ScopeConsulta<transação>(), o comprovante recebido deve ser enviado para a impressora.

Protótipo

```
LONG EXPORT ScopeGetCupomEx(WORD CabecLen, char *Cabec,
    WORD CupomClienteLen, char *CupomCliente,
    WORD CupomLojaLen, char *CupomLoja,
    WORD CupomReduzLen, char *CupomReduz,
    BYTE *NroLinhasReduz)
```

Parâmetros

[in]	WORD	CabecLen	Tamanho reservado pela aplicação de PDV para receber o cabeçalho do cupom
[out]	String	Cabec	Ponteiro para área onde será recebido o cabeçalho do cupom
[in]	WORD	CupomClienteLen	Tamanho reservado pela aplicação de PDV para receber a via do cupom Cliente
[out]	String	CupomCliente	Ponteiro para área onde será recebida a via do cliente que sempre estará disponível para a aplicação de PDV
[in]	WORD	CupomLojaLen	Tamanho reservado pela aplicação para receber a via da loja
[out]	String	CupomLoja	Ponteiro para área onde será recebida a via da loja que sempre estará disponível para a aplicação

[in]	WORD	CupomReduzLen	Tamanho reservado pelo aplicativo para receber o cupom reduzido
[out]	String	CupomReduz	Ponteiro para área onde será recebida a via cupom reduzido que poderá substituir a via do cliente e que em alguns casos, esta via pode não estar disponível para a aplicação
[out]	WORD	NroLinhasReduz	Número de linhas que tem o cupom reduzido. Se o valor for zero, não há cupom reduzido disponível

IMPORTANTE: Os finalizadores de linha dos cupons retornados pela função ScopeGetCupomEx estão de acordo com a parametrização do ScopeForneceCampo(SCOPE_DADO_SEPARADOR_LINHA).

Retorno

Ver [tabela de código de retorno.](#)

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE04	65028	Não existe transação suspensa
0xFF12	65298	Área reservada para o buffer é insuficiente para o SCOPE Client x os dados solicitados

Exemplo

```

...
BYTE NroLnhReduzido = 0;
char Cabec[1024],
    CpCliente[2048],
    CpLoja[2048],
    CpReduzido[2048];
LONG retorno;
...
// obtém status do SCOPE
switch(status_scope)
{
...
case 64582:
case 64514:
case 64539:
case 64546:
case 64593:
    memset(Cabec, 0, sizeof(Cabec));
    memset(CpCliente, 0, sizeof(CpCliente));
    memset(CpLoja, 0, sizeof(CpLoja));
    memset(CpReduzido, 0, sizeof(CpReduzido));
    retorno = ScopeGetCupomEx(sizeof(Cabec), Cabec,
                               sizeof(CpCliente), CpCliente,
                               sizeof(CpLoja), CpLoja,
                               sizeof(CpReduzido), CpReduzido,
                               &NroLnhReduzido);
    if (RC == RCS_SUCESSO)
    {

```

```

    ImprimeCupomTEF("CABECALHO", Cabec);
    ImprimeCupomTEF("CUPOM DO CLIENTE", CpCliente);
    ImprimeCupomTEF("CUPOM DA LOJA", CpLoja);
    if (NroLnhReduzido > 0)
        ImprimeCupomTEF("CUPOM REDUZIDO", CpReduzido);
    }
    break;
...
}
...

```

Reimpressão de comprovante

Conforme necessidade, o operador pode solicitar uma nova cópia do comprovante da última transação realizada ou alguma específica anterior a última, desde que tenha sido aprovada e se encontre no SCOPE. Essa necessidade pode ter sido ocasionada por algum problema com a impressão original (cupom ilegível, papel enroscado na impressora, etc.).

A reimpressão do comprovante é uma transação e como tal, deverá ser tratada numa sessão de TEF, preferencialmente, sendo a única da sessão.

Solicitando o comprovante

Para toda e qualquer rede, o SCOPE permite reimprimir apenas o último comprovante realizado num PDV. É possível reimprimir cupons antigos, exceto o reduzido, desde que estes estejam na base de dados do SCOPE. Esta reimpressão atua apenas entre o SCOPE Client e o SCOPE Server e não vai para a autorizadora.

Protótipo

```
LONG EXPORT ScopeReimpressaoComprovante (void)
```

Parâmetros

Não há parâmetros.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada

OxFF0A	65290	Banco de dados off-line
*OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
// abre sessão
...
...
retorno = ScopeReimpressaoComprovante ();
...
...
// processa a transação
...
...
// fecha a sessão
...
...

```

IMPORTANTE:

- na reimpressão não há cupom reduzido.
- a função ScopeReimpressaoOffLine() possui o mesmo comportamento da ScopeReimpressaoComprovante() e foi mantida por questões de compatibilidade.

Solicitando o comprovante on-line (obsoleto)

Utilizada nos casos em que a rede autorizadora ou o correspondente bancário oferece a transação (on-line) de reimpressão do comprovante do pagamento de conta.

Protótipo

```
LONG EXPORT ScopeReimpressaoOnLine (WORD CodBandeira)
```

Parâmetros

[in]	WORD	CodBandeira	Código da bandeira do cartão (ver Código das bandeiras)
------	------	-------------	--

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado

OxFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
WORD CodBandeira;
// abre sessão
...
// Obtém a bandeira do cartão
...
retorno = ScopeReimpressaoOnLine(CodBandeira);
...
// processa a transação
...
// fecha a sessão
...

```

IMPORTANTE:

- atualmente esta função está obsoleta, pois as transações que permitiam obter o comprovante de forma online foram descontinuadas pelas redes adquirentes
- embora obsoleta, esta função é mantida por questões de compatibilidade.

Imprimindo o comprovante correto

Como acontece em qualquer outra transação, na reimpressão de comprovante, os cupons são recebidos pela função de cupons (ver [Comprovantes de transações](#)), a qual retorna todos os cupons. Entretanto, devido à exigência da Visanet, a aplicação deverá imprimir apenas o cupom solicitado pelo operador. Para isso, a aplicação deverá consultar no SCOPE qual é o cupom que deverá ser impresso, com a ajuda da função ScopeObtemTipoViaReimpressao().

Protótipo

```
LONG EXPORT ScopeObtemTipoViaReimpressao (BYTE *EhReimpres, BYTE *Via)
```

Parâmetros

[out]	BYTE	EhReimpres	Informa se é uma transação de reimpressão (valor = 1) ou qualquer outra transação (valor = 0)
[out]	BYTE	Via	Caso seja uma transação de reimpressão, informa qual a via a ser impressa: 0: todas as vias

			1: apenas a via da loja 2: apenas a via do cliente
--	--	--	---

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE04	65028	Não existe transação suspensa
0xFF11	65297	Não existe cupom válido

Exemplo

```

...
BYTE NroLnhReduzido = 0,
EhReimpressao,
ViaReimpressao;
char Cabec[1024],
CpCliente[2048],
CpLoja[2048],
CpReduzido[2048];
LONG retorno;
...
// obtém status do SCOPE
switch(status_scope)
{
...
case 64582:
case 64514:
case 64539:
case 64546:
case 64593:
    memset(Cabec, 0, sizeof(Cabec));
    memset(CpCliente, 0, sizeof(CpCliente));
    memset(CpLoja, 0, sizeof(CpLoja));
    memset(CpReduzido, 0, sizeof(CpReduzido));
    retorno = ScopeGetCupomEx(sizeof(Cabec), Cabec,
                               sizeof(CpCliente), CpCliente,
                               sizeof(CpLoja), CpLoja,
                               sizeof(CpReduzido), CpReduzido,
                               &NroLnhReduzido);
    if (retorno == RCS_SUCESSO)
    {
        retorno = ScopeObtemTipoViaReimpressao( &EhReimpressao,
                                                &ViaReimpressao );
        if(EhReimpressao == 0)
        {
            ImprimeCupomTEF("CABECALHO", Cabec);
            ImprimeCupomTEF("CUPOM DO CLIENTE", CpCliente);
            ImprimeCupomTEF("CUPOM DA LOJA", CpLoja);
            if (NroLnhReduzido > 0)
                ImprimeCupomTEF("CUPOM REDUZIDO", CpReduzido);
        }
    }
}
else

```

```

{
    switch(ViaReimpressao)
{
    case 0:
        ImprimeCupomTEF("CUPOM DO CLIENTE", CpCliente);
        ImprimeCupomTEF("CUPOM DA LOJA", CpLoja);
        break;
    case 1:
        ImprimeCupomTEF("CUPOM DO CLIENTE", CpCliente);
        break;
    case 2:
        ImprimeCupomTEF("CUPOM DA LOJA", CpLoja);
        break;
    }
}
break;
...
}
...

```

PBM - Medicamentos

Aqui trataremos do grupo de funções que atendem a funcionalidade de PBM.

Consultando medicamento

A função ScopeConsultaMedicamento() aciona uma transação no SCOPE para obter uma lista de medicamentos relacionados a uma autorização, assim como a quantidade autorizada, o PMC (Preço Máximo para o Consumidor) e o preço e-Pharma para cada um destes.

O processamento segue a linha de uma transação qualquer de TEF, em que a aplicação deve processar através das chamadas à função de consulta status (ver [Status de transação](#)) e coleta de dados. O PDV deve esperar pelo estado 64580 para chamar a função específica e obter a lista de medicamentos (ver [Lista de medicamentos](#), [Lista de medicamentos com CRM](#) e [Lista de medicamentos Ex](#)).

Protótipo

LONG EXPORT ScopeConsultaMedicamento(BYTE TipoConvenio, BYTE CódigoRede)
--

Parâmetros

[in]	BYTE	TipoConvenio	Tipo do convênio. (0=PBM, 1=Empresa)
[in]	BYTE	CódigoRede	Código da rede (ver Convênios)

RetornoVer [tabela de código de retorno](#).**Possíveis Retornos de Erros**

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
BYTE TipoConvenio, CódigoRede;
...
// abre sessão
...
// coleta o tipo do convênio, o código da rede
...
retorno = ScopeConsultaMedicamento(TipoConvenio, CódigoRede);
...
// processa a transação
...
// fecha a sessão
...

```

Nesse tipo de transação, o SCOPE não possui controle do valor de venda a ser registrado. Portanto, ao receber o código 64579 da função de status, a aplicação deve atualizar o valor (ver [ScopeAtualizaValor](#)), informando assim o valor da venda.

Exemplo

```

char VlOperac[12+1];
...

```

```

switch(status_scope)
{
...
Case 64579:
    RecebeValorDoOperador                               (&VIOperac);
    iRet = ScopeAtualizaValor (VIOperac);
    if (iRet != RCS_SUCESSO)
        // Tratar erro
        break;
...

```

Compra de medicamento

Para iniciar uma compra de medicamentos PBM, deve-se chamar a função ScopeCompraMedicamento(). O PDV deve esperar pelo estado de coleta 64579 (ver [Status de transação](#)) para poder fornecer ao Scope a Lista de medicamentos selecionados para a compra, que será enviado posteriormente no bit 61 da transação. Quando o estado de coleta 64579 é acionado, o PDV chama a função ScopeForneceCampo para fornecer a lista de medicamentos vide em ([Fornecendo Lista de Medicamentos](#)).

Protótipo

LONG EXPORT ScopeCompraMedicamento(BYTE TipoConvenio, BYTE CódigoRede, char *NumCpFiscal)

Parâmetros

[in]	BYTE	TipoConvenio	Tipo do convênio. (0=PBM, 1=Empresa)
[in]	BYTE	CódigoRede	Código da rede (ver Convênios)
[out]	String	NumCpFiscal	Número do cupom fiscal para o PDV

A Automação Comercial deverá enviar o parâmetro NumCpFiscal formatado da seguinte maneira:

- Se NumCpFiscal for referente ao cupom fiscal, o mesmo deverá ser preenchido de 1 a 9 dígitos.
- Se NumCpFiscal for referente ao regime tributário SAT, o mesmo deverá ser preenchido com 44 dígitos no seguinte formato:

4.7. Chave de Acesso do CF-e-SAT

A chave de acesso do CF-e-SAT será representada por 44 caracteres numéricos, sendo composta pelos seguintes campos que se encontram no CF-e-SAT:

- **cUF** - Código da UF do emitente do Cupom Fiscal;
- **AAMM** – Ano e Mês de emissão do CF-e-SAT;
- **CNPJ** – CNPJ do emitente;
- **mod** – Modelo do Documento Fiscal;
- **nserieSAT** – Número de série do Equipamento SAT;
- **nCF** – Número do Cupom Fiscal;
- **cNF** – Código Numérico Aleatório;
- **cDV** – Dígito Verificador da Chave de Acesso.

Os campos estão dispostos da seguinte forma:

	Código da UF	AAMM da emissão	CNPJ do emitente	mod	Nº de Série do SAT	Número do CF-e-SAT	Código Número Aleatório	Dígito Verificador
Quantidade de caracteres	02	04	14	02	09	06	06	01

Tabela 9 – Disposição dos Campos da Chave de acesso

O Dígito Verificador (DV) irá garantir a integridade da chave de acesso, protegendo principalmente contra digitações erradas.

No campo modelo virá a informação que o Scope deverá preencher no bit 19, para esse caso do SAT o conteúdo será 59.

Se NumCpFiscal for referente ao regime tributário NFCe ou NFE, o mesmo deverá ser preenchido com 44 dígitos no seguinte formato:

A partir da versão 2.00 do leiaute da NF-e, o campo *tpEmis* (forma de emissão da NF-e) passou a compor a chave de acesso da seguinte forma:

	Código da UF	AAMM da emissão	CNPJ do Emitente	Modelo	Série	Número da NF-e	forma de emissão da NF-e	Código Numérico	DV
Quantidade de caracteres	02	04	14	02	03	09	01	08	01

O tamanho do campo *cNF* - código numérico da NF-e foi reduzido para oito posições para não alterar o tamanho da chave de acesso da NF-e de 44 posições, que passa a ser composta pelos seguintes campos que se encontram dispersos na NF-e :

- *cUF* - Código da UF do emitente do Documento Fiscal
- *AAMM* - Ano e Mês de emissão da NF-e
- *CNPJ* - CNPJ do emitente
- *mod* - Modelo do Documento Fiscal
- *serie* - Série do Documento Fiscal
- *nNF* - Número do Documento Fiscal
- *tpEmis* – forma de emissão da NF-e
- *cNF* - Código Numérico que compõe a Chave de Acesso
- *cDV* - Dígito Verificador da Chave de Acesso

O Dígito Verificador (DV) irá garantir a integridade da chave de acesso, protegendo-a principalmente contra digitações erradas.

No campo modelo virá a informação que o Scope deverá preencher no bit 19, para esse se for NFCe o conteúdo virá 65, se for NFE, o conteúdo virá 55.

OBS – Nas transações de compra e cancelamento de medicamento, o bit 19 é composto das seguintes informações:

Bit	Form.	Atrib.	Descrição	Farm.■Host	Host■Farm.
19		n3	Tipo de documento fiscal usado na venda: 1xx=Cupom Fiscal. 2xx=NFC-e. 3xx=SAT.	M	

Atualmente somente a ePharma com versão de especificação V5.1.0 faz o envio dessas informações no bit 19 e 62 e também somente a informação do cupom fiscal, pois a automação comercial não está preparada para enviar a chave de acesso de 44 bytes no momento. Mesmo assim o Scope está preparado para futuramente receber e tratar essas informações.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado

OxFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
BYTE TipoConvenio, CódigoRede;
char *NumCpFiscal;
...
// abre sessão
...
// coleta o tipo do convênio, o código da rede e o número do cupom fiscal
...
retorno = ScopeCompraMedicamento (TipoConvenio, CódigoRede, NumCpFiscal);
...
// processa a transação
...
// fecha a sessão
...

```

Funcionamento a partir do layout 5

A mudança para o layout 5 (conforme descrito no documento "Scope PBM - Versão 1.10.doc") trouxe mudanças significativas no funcionamento da compra de medicamentos. Para realizar uma compra de medicamentos, é preciso obrigatoriamente ter feito uma consulta de medicamentos antes e estar na mesma sessão TEF.

O motivo para tal funcionamento é que a partir do layout 5, pode ser requisitado coleta de senha através do bit 22 do retorno da consulta (mensagem 0110). Como essa informação referente à necessidade de coleta de senha deve ser coletada em uma transação diferente (0200), ela é armazenada em um arquivo de contexto e recuperado na compra de medicamentos. Atualmente, as redes Vidalink e PBM Padrão trabalham com o layout na nova versão.

Fornecendo Lista de Medicamentos

Num fluxo de compra de medicamento, recebendo o código de coleta 64579 (ver [Status de transação](#)), a automação deverá fornecer a lista de medicamentos selecionada pelo operador para ser enviada na transação de compra. O fornecimento da lista é feito através da função ScopeForneceCampo() e o formato das listas fornecidas por esta função encontra se nas tabelas abaixo.

- Identificado como formato 02

Tam.	Formato	Descrição
6	String	Header fixo em "FMT02#"
13	String	Código EAN do medicamento.
2	String	Quantidade autorizada do produto (podendo ser zero)
7	String	Valor da venda (em centavos)
7	String	Preço bruto do Produto
1	Char	Trailer OxoO

- Identificado como formato 03 específico para a PharmaSystem

Tam.	Formato	Descrição
6	String	Header fixo em "FMT03#"
13	String	Código EAN do medicamento.
4	String	Quantidade autorizada do produto (podendo ser zero)
7	String	Preço bruto do Produto
7	String	Preço líquido
10	String	Código da categoria do produto
40	String	Descrição do Produto se categoria convênio
1	Char	Trailer OxoO

- Identificado como formato 04, esse registro é específico da nova versão da ePharma V5.1.0

Tam.	Formato	Descrição
6	String	Header fixo em "FMT04#"
13	String	Código EAN do medicamento.
2	String	Quantidade autorizada do produto (podendo ser zero)
7	String	Valor da venda (em centavos)
1	Char	Tipo de Pagamento (1 para PBM / 2 para Convênio)
8	String	Código da categoria do produto
40	String	Descrição do Produto se categoria convênio
1	Char	Trailer OxoO

Protótipo

```
LONG EXPORT ScopeForneceCampo (char TypeField, void *StructField)
```

Parâmetros

[in]	char	TypeField	Tipo de dado
[in]	void *	StructField	Ponteiro para os dados

Para transações de compra PBM o parametro TypeField deverá ir com o valor 16 (Fornecendo informações extras para a transação) e a estrutura StructField com uma das Listas acima.

Lista de medicamentos

Num fluxo de compra ou consulta de medicamento, recebendo o código de coleta 64580 (ver [Status de transação](#)), a aplicação deverá obter a lista de medicamentos disponíveis, através das funções ScopeObtemMedicamentos(), ScopeObtemMedicamentosComCRM() ou ScopeObtemMedicamentosEx(), que serão tratadas logo abaixo. O formato da lista obtido pela função ScopeObtemMedicamentos() encontra-se na tabela abaixo.

Posição	Formato	Descrição
01 a 13	String	Código EAN do medicamento.
14 a 15	String	Quantidade autorizada do produto.
16 a 22	String	Preço máximo ao consumidor.
23 a 29	String	Preço de venda.
30 a 36	String	Preço de fábrica.
37 a 43	String	Preço de aquisição.
44 a 50	String	Preço de repasse.
51	Byte	Reservado para uso futuro.
52 a 53	String	Motivo da rejeição.

Protótipo

```
LONG EXPORT ScopeObtemMedicamentos(BYTE *Qtd,
char *ListaMedicamentos,
WORD TamLista)
```

Parâmetros

[out]	BYTE	Qtd	Retorna a quantidade de medicamentos consultados ou comprados
[out]	String	ListaMedicamentos	Retorna os medicamentos consultados ou comprados
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaMedicamentos"

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE04	65028	Não existe transação suspensa
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada

OxFFOA	65290	Banco de dados off-line
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

BYTE bQtdVias = 0;
LONG IRet;
stREGISTRO_MEDICAMENTO lstMedsCRM [38] = {0};

IRet = ScopeObtemMedicamentos
    ( &bQtdVias, (char *) &lstMedsCRM, sizeof (lstMedsCRM) );

if (IRet == RCS_SUCESSO)
{
    /* Sucesso no recebimento dos medicamentos,
       os membros das estruturas estarão preenchidos */
}
else
    // Erro ...

```

Lista de medicamentos com CRM

A função ScopeObtemMedicamentosComCRM() recupera a lista dos medicamentos consultados ou comprados com o CRM do médico. Esta função não suporta as bandeiras Novartis e FlexMed e seu formato obedece à definição da tabela abaixo.

Posição	Formato	Descrição
01 a 53	String	Dados do medicamento, obedecendo a definição da tabela retornada para a função ScopeObtemMedicamentos().
54 a 62	String	Número do CRM.

Protótipo

```

LONG EXPORT ScopeObtemMedicamentosComCRM(BYTE *Qtd,
                                         BYTE *TipoConv,
                                         char *ListaMedicamentos,
                                         WORD TamLista)

```

Parâmetros

[out]	BYTE	Qtd	Retorna a quantidade de medicamentos consultados ou comprados
[out]	BYTE	TipoConv	Retorna o tipo de convênio: 0: Pagamento à vista; 1: Pagamento a prazo; 2: Empresa Fechada; 3: Empresa Aberta.
[out]	String	ListaMedicamentos	Retorna os medicamentos consultados ou comprados
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaMedicamentos"

Retorno

Ver [tabela de código de retorno](#).

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE04	65028	Não existe transação suspensa
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere “#” como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF11	65297	Não existe cupom válido
0xFF12	65298	Área reservada para o buffer é insuficiente para o SCOPE Client x os dados solicitados
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Descrição dos tipos de Convênio:

0 – Pagamento à vista = indica que o consumidor final irá pagar integralmente o valor do medicamento no ato da compra.

Para o caso da FuncionalCard, esse tipo representa PBM indústria, ou seja, ocorre quando há descontos de laboratórios.

1 – Pagamento à prazo = indica que o consumidor final não paga nada no ato da compra e que o valor integral dos medicamentos contidos na lista de medicamento (BIT 61) será descontado futuramente do convenio do consumidor final.

Para o caso da FuncionalCard, esse tipo de PBM indica convênio corporativo.

2 – Pagamento Empresa Fechada = indica que o consumidor final não irá pagar nada no ato da compra e que o valor integral poderá ou não ser descontado em folha de pagamento. Além de que no cupom de venda deve conter somente todos os itens da autorização de PBM.

Para o caso da FuncionalCard, esse tipo indica pagamento parcial a vista e convênio coporativo.

3 - Pagamento Empresa Aberta = indica que o consumidor final não irá pagar nada no ato da compra e que o valor integral poderá ou não ser descontado em folha de pagamento. Além de que no cupom de venda pode conter outros itens além daqueles constantes na autorização de PBM.

Observação:

Para a FuncionalCard, foi definido que o retorno dos seguintes tipos de convênios:

Lista de medicamentos Estendida

A função ScopeObtemMedicamentosEx() recupera a lista dos medicamentos consultados ou comprados em qualquer formato disponível atualmente. A função deve receber qual a versão que se deseja da lista de medicamentos como será descrito a seguir. Os formatos de listas antigas permanecem os mesmos, porém, há o formato do layout 5 (conforme descrito no documento “Scope PBM - Versão 1.10.doc”) que obedece à tabela abaixo:

Posição	Formato	Descrição
01 a 13	String	Código EAN do medicamento.
14 a 15	String	Quantidade autorizada do produto.
16 a 17	String	Motivo da rejeição.
18 a 26	String	Código do convênio.
27 a 34	String	Preço máximo ao consumidor.
35 a 42	String	Preço de venda.
43 a 50	String	Preço de fábrica.
51 a 58	String	Preço de aquisição.
59 a 66	String	Preço de repasse.
67 a 79	String	Número do CRM.

Protótipo

```
LONG EXPORT ScopeObtemMedicamentosEx(BYTE IdLayout,
BYTE *_QtdRegistros,
char *_ListaMedicamentos,
WORD _TamLista,
BYTE *_TipoConvenio)
```

Parâmetros

[in]	BYTE	IdLayout	Versão da lista de medicamentos que se deseja recuperar, devendo ser como definido na tabela a seguir.
[out]	BYTE	QtdRegistros	Retorna a quantidade de medicamentos consultados ou comprados.
[out]	String	ListaMedicamentos	Retorna os medicamentos consultados ou comprados.
[in]	WORD	TamLista	Tamanho, em bytes, do campo “ListaMedicamentos”

[out]	BYTE	TipoConvenio	Retorna o tipo de convênio: 0: Pagamento à vista; 1: Pagamento a prazo; 2: Empresa Fechada; 3: Empresa Aberta.
-------	------	--------------	--

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE04	65028	Não existe transação suspensa
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Layouts:

ID layout	Usar estrutura
1	stREGISTRO_MEDICAMENTO
2	stREGISTRO_MEDICAMENTO_CRM
3	stREGISTRO_MEDICAMENTO_L3
4	stREGISTRO_PHARMASYSTEM_RET

IMPORTANTE: Atualmente, somente a rede Vidalink e PBM Padrão é compatível com o layout 3 de estrutura.

Lista de Projetos

A função ScopeRecuperaBufTabela() fornecerá para a aplicação de automação comercial uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada. Ela servirá para facilitar futuras implementações que atendam a mesma finalidade. O parâmetro _TipoTabela indicará o formato dos dados que serão fornecidos.

```
LONG EXPORT ScopeRecuperaBufTabela (BYTE TipoTabela,
char *_QtdRegistros,
char *_Buffer,
WORD _TamBuffer)
```

Parâmetros

[in]	BYTE	TipoTabela	Tipo de lista de elementos que se deseja recuperar, devendo ser como definido na tabela a seguir.
[out]	String	QtdRegistros	Retorna a quantidade de elementos da tabela
[out]	String	Buffer	Retorna os elementos da tabela.
[in]	WORD	TamBuffer	Tamanho, em bytes, do campo "Buffer"

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFA02	64002	Parâmetro 2 inválido
OxFA03	64003	Parâmetro 3 inválido

Layouts:

ID layout	Usar estrutura	Descrição
BUF_TAB_PRJ_PHARMASYSTEM (ID 0)	stREGISTRO_PROJETO_PHMS	Projeto PharmaSystem

O formato obedece à tabela, conforme a especificação da PharmaSystem

Posição	Formato	Descrição
01 a 06	String	Código do Projeto
07 a 36	String	Descrição do Projeto
37 a 56	String	Operadora do Projeto

Elegibilidade do Cartão PBM

Para iniciar uma compra de medicamentos PDM, sem a existência de uma autorização previamente fornecida por uma aplicação de balcão de farmácia, deve-se chamar a função ScopeElegibilidadeCartao(), que fará a validação dos dados do cliente que deseja efetuar a compra. Essa função suporta a bandeira PharmaSystem.

Protótipo

```
LONG EXPORT ScopeElegibilidadeCartao (BYTE _TipoConvenio,
                                     BYTE _CodRede,
                                     char *NumCpFiscal)
```

Parâmetros

[in]	BYTE	_TipoConvenio	Tipo do convênio. (0=PBM, 1=Empresa)
[in]	BYTE	_CodRede	Código da rede (ver Convênios)

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFA02	64002	Parâmetro 2 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
BYTE _TipoConvenio, _CodigoRede;
...
// abre sessão
...
// coleta o tipo do convênio, o código da rede
...
retorno = ScopeElegibilidadeCartao (_TipoConvenio, _CodigoRede);
...
// processa a transação
...
// fecha a sessão
...

```

Pré-Autorização de Medicamentos PBM

Para informar cada produto que será incluído na compra, obtendo informações do mesmo, deve ser chama a função ScopePreAutorizacaoMedicamento(). A função pode retornar no fluxo de coleta o código de coleta 64579 para fornecer a lista de medicamentos, e o código de coleta 64580 para obter a lista de medicamentos. Essa função suporta a bandeira PharmaSystem.

Protótipo

LONG EXPORT ScopePreAutorizacaoMedicamento (BYTE _TipoConvenio, BYTE _CodRede, char *NumCpFiscal)

Parâmetros

[in]	BYTE	_TipoConvenio	Tipo do convênio. (0=PBM, 1=Empresa)
------	------	---------------	--------------------------------------

[in]	BYTE	_CodRede	Código da rede (ver Convênios)
------	------	----------	---

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```

...
BYTE TipoConvenio, CodigoRede;
...
// abre sessão
...
// coleta o tipo do convênio, o código da rede
...
retorno = ScopePreAutorizacaoMedicamento (TipoConvenio, CodigoRede);
...
// processa a transação
...
// fecha a sessão
...

```

Cancelamento de Pré-Autorização de Medicamentos PBM

Para cancelar um produto da lista dos produtos autorizados, deve-se chamar a função ScopeCancelaPreAutMedicamento(), que excluirá o produto e a quantidade da lista de produtos. A função

pode retornar no fluxo de coleta o código de coleta 64579 para fornecer a lista de medicamentos, e o código de coleta 64580 para obter a lista de medicamentos. Essa função suporta a bandeira PharmaSystem.

Protótipo

```
LONG EXPORT ScopeCancelaPreAutMedicamento()
```

Parâmetros

[in]	BYTE	_TipoConvenio	Tipo do convênio. (0=PBM, 1=Empresa)
[in]	BYTE	_CodRede	Código da rede (ver Convênios)

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFFFF	65535	Erro genérico
0xFF5E	65374	Erro ao desmontar a estrutura ISO

Exemplo

```
...
BYTE TipoConvenio, CodigoRede;
...
// abre sessão
...
// coleta o tipo do convênio, o código da rede
...
retorno = ScopeCancelaPreAutMedicamento (TipoConvenio, CodigoRede);
...
// processa a transação
...
// fecha a sessão
...
```

Operações Fidelidade

Aqui trataremos do grupo de funções que atendem a funcionalidade de Fidelidade.

Tipos de Operação Fidelidade

As operações (transações) que podem ser efetuadas com Fornecedores de Produtos Fidelidade são:

- Consulta Pontos
- Acúmulo por Valor
- Acúmulo por Pontos
- Resgate de Produtos
- Resgate Monetizado (por valor)

Operação Fidelidade

Para realizar a operação Fidelidade deve-se utilizar a função ScopeFidelidade(), como descrito abaixo.

Essa função suporta atualmente a bandeira Multiplus.

Como as outras transações, o fluxo de processamento é similar.

Protótipo

```
LONG EXPORT ScopeFidelidade (WORD Servico, WORD Bandeira)
```

Parâmetros

[in]	WORD	Servico	Para efetuar a operação Fidelidade, usar um dos seguintes serviços <ul style="list-style-type: none"> • 63 S_CONSULTA_PONTOS • 145 S_ACUMULO_POR_VALOR • 146 S_ACUMULO_POR_PONTOS • 147 S_RESGATE_DE_PRODUTO • 148 S_RESGATE_MONETIZADO
[in]	WORD	Bandeira	Código da bandeira (ver Código das bandeiras) O código da bandeira representa o Fornecedor Fidelidade da transação

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF22	65314	Serviço não configurado
0xFF34	65332	Código de barra inválido
0xFF41	65345	Erro: dados indisponíveis

OxFF5F	65375	Bandeira não está configurada
OxFF6B	65387	Não Encontrado (para quando não veio menu dinâmico na carga de tabelas).

Exemplo

```

...
Long servico = S_CONSULTA_PONTOS;
Long bandeira = B_MULTIPLUS;
...
// abre sessão
...
retorno = ScopeFidelidade(servico, bandeira);
...
// processa a transação
...
// fecha a sessão
...

```

Cancelamentos das Operações

Tanto para realizar a operação de cancelamento de Acúmulo como cancelamento de Resgate deve ser usada a função [ScopeCancelamento\(\)](#). A documentação desta função esta descrita no capítulo Estorno de transações.

A operação de Consulta de Pontos não requer cancelamento.

Pinpad Compartilhado e ABECS

Esta seção documenta as funções de acesso ao pinpad ABECS e ao pinpad compartilhado. Estas funções seguem o formato ScopePPFuncao().

Utilize o apêndice [B](#), [C](#) e a [tabela de código de retorno](#) como complemento.

IMPORTANTE: A utilização destas funções, com exceção das usadas para a exibição de mensagens no visor, está condicionada à configuração no ScopeCNF, onde a opção “Uso exclusivo do Scope” deve estar desmarcada.

O pinpad ABECS é um dispositivo que segue a especificação criada pela ABECS para a interoperação com software de pagamento eletrônico. O pinpad compartilhado segue a especificação da Biblioteca Compartilhada, criada pelas adquirentes CIELO, REDE e AMEX. Esse pinpad será substituído pelo pinpad ABECS.

A especificação ABECS garante a compatibilidade com a especificação da Biblioteca Compartilhada, isto é, se configurado um pinpad compartilhado e conectar um pinpad ABECS, ele irá funcionar como um compartilhado. O SCOPE, por sua vez, irá manter a compatibilidade se estiver configurado um pinpad ABECS e conectar um pinpad compartilhado no PDV.

Funcionamento

O funcionamento dos pinpads ABECS é semelhante ao dos pinpads Compartilhados. Existem comandos para iniciar e encerrar a comunicação, para escrever na tela do pinpad, para a leitura de cartão, para a coleta de um dado. No entanto, por motivos de segurança, haverá alteração na API de acesso ao pinpad. Algumas funções serão descontinuadas e substituída por outras. Esses comandos servirão tanto para o uso com pinpads ABECS quanto para o uso de pinpads compartilhado, exceto pelas funções de menu (ScopePPStartOptionMenu e ScopePPOptionMenu) que não é suportado pelo pinpad compartilhado.

Funções descontinuadas	Funções novas
ScopePPOpen	ScopePPOpenSecure
ScopePPGetInfo	ScopePPGetInfoEx
ScopePPStartGetPIN	
---	ScopePPStartGetData
---	ScopePPGetData
---	ScopePPStartOptionMenu
---	ScopePPOptionMenu
---	ScopePPGetOperationMode

Canais de comunicação

Com a diversidade de plataformas e meios de comunicação, houve a necessidade de adaptação e reestruturação do SCOPE para o suporte a tais canais, que são: serial, USB e bluetooth. Entretanto a disponibilidade do canal está sujeito à plataforma em que o SCOPE Client está rodando, conforme abaixo.

Plataforma	Canal de comunicação suportado	Endereço
MS-Windows	Serial	Número indicando a porta serial
	USB	Número indicando a porta serial equivalente
Linux	Serial	Número indicando a porta serial que será convertida para o correspondente device /dev/ttysX
	USB	Caminho para o device correspondente (exemplo: /dev/ttYACMO)
Android	Bluetooth	Endereço do dispositivo pareado

Comandos de controle

A seguir os comandos necessários para o uso do pinpad.

Abrir comunicação

Como em qualquer dispositivo, a comunicação com o pinpad precisa ser iniciada e, após o uso, finalizada. As funções aqui relacionadas devem sempre ser chamadas quando a opção “Uso exclusivo do Scope” não estiver marcada.

Antes do início de qualquer transação, a aplicação deve abrir o pinpad, ou seja, iniciar o canal de comunicação com o pinpad. A chamada bem sucedida desta função é pré-requisito para todas as outras da interface com o pinpad.

Essa função opera apenas no modo de pinpad compartilhado e será descontinuada. Abrir a comunicação com essa função, o SCOPE utilizará o pinpad no modo compartilhado. A aplicação deverá substituir pela função ScopePPOpenSecure.

Protótipo	LONG EXPORT ScopePPOpen(WORD PortaSerial)	
Parâmetros de entrada	PortaSerial	Porta serial que se encontra conectado o pinpad.
Parâmetros de saída	Não há.	
Retornos previstos	PC_OK PC_ALREADYOPEN PC_COMMERR PC_MEM_NAO_ALOCADA PC_ERRO_ALOCANDO_MEMORIA PC_JA_ABERTO_VIA_SCOPE	
Exemplo	<pre>... WORD porta = 1; LONG retorno = 0; ... retorno = ScopePPOpen(porta); ...</pre>	

Abrir comunicação segura

Com a nova função para abrir a comunicação com o pinpad, devido novos canais de comunicação, a automação comercial deverá indicar o canal de comunicação em que o pinpad está instalado, além do endereço ou porta.

Essa função é compatível com os pinpads ABECS e compartilhado. Quando o pinpad for ABECS, toda a comunicação entre o SCOPE e o pinpad será criptografada, conforme a especificação ABECS exige.

Protótipo	LONG ScopePPOpenSecure(WORD TipoCanal, char *Endereco)	
Parâmetros de entrada	TipoCanal	Tipo de canal de comunicação, que pode ser: PC_COMM_NONE – configurado no scope.ini PC_COMM_SERIAL – comunicação serial PC_COMM_USB – comunicação USB PC_COMM_BLUETOOTH – comunicação Bluetooth
	Endereco	String com o endereço do canal de comunicação.
Parâmetros de saída	Não há.	
Retornos previstos	PC_OK PC_ALREADYOPEN PC_COMMERR PC_MEM_NAO_ALOCADA PC_ERRO_ALOCANDO_MEMORIA PC_JA_ABERTO_VIA_SCOPE	
Exemplo	<pre>... char* endereco = "1"; LONG retorno = 0; ...</pre>	

	<pre>retorno = ScopePPOpenSecure(PC_COMM_SERIAL, endereco); ...</pre>
--	---

Encerrar comunicação

Uma vez que o pinpad não será mais utilizado, a aplicação deve encerrar a comunicação, deixando uma mensagem de formato livre no visor (display) do pinpad.

Protótipo	LONG ScopePPClose(char* IdleMsg)	
Parâmetros de entrada	IdleMsg	Mensagem a ser apresentada no visor do pinpad após a execução do comando. Se não fornecida, o conteúdo do visor é simplesmente apagado.
Parâmetros de saída	Não há.	
Retornos previstos	PC_OK PC_NAO_ABERTO_APP PC_JA_ABERTO_VIA_SCOPE	
Exemplo	... <pre>retorno = ScopePPClose("LOJAS TIRIRICA! PIOR QUE TÁ NÃO FICA.");</pre>	

Obter informações do pinpad (obsoleta)

Internamente, os pinpads com a biblioteca compartilhada ou ABECS, possuem informações armazenadas sobre o próprio dispositivo. Quando o parque de PDVs é muito grande, o levantamento via aplicativo dos pinpad é uma maneira mais segura e rápida que o método manual.

A função ScopePPGetInfo retorna as informações sobre o pinpad e suas aplicações, como descrito acima. Caso alguma informação não exista ou não se aplique para o modelo de pinpad, ela deverá ser fornecida em branco (espaços). Para decidir qual informação a aplicação quer obter, o primeiro parâmetro deve receber o valor correspondente:

- valor 0: dados do pinpad;
- valor maior que 0: dependendo do valor, serão obtidas as informações específicas das aplicações das autorizadoras.

Protótipo	LONG EXPORT ScopePPGetInfo(WORD IdSaída, WORD DadosLen, char* Dados)	
Parâmetros de entrada	IdSaída	Informa a opção do formato dos dados de saída. O único valor permitido é o PC_GETINFO_HW
	DadosLen	Tamanho da área reservada pelo aplicativo, indicado pela variável Dados, para receber os dados
Parâmetros de saída	Dados	Dados do pinpad, conforme o formato descrito abaixo.
Retornos previstos	PC_OK PC_NAO_ABERTO_APP PC_MEMORIA_INSUFICIENTE RCS_ERRO_PARM_1	
Exemplo	<pre>char aux[255]; ... if (ScopePPGetInfo(0, (WORD) sizeof(aux), aux)) { printf("Nome do fabricante.: [%20.20s]\n", &aux[00]); printf("Modelo hardware....: [%19.19s]\n", &aux[20]);</pre>	

	<pre> printf("Suporta Contactless: [%c]\n", &aux[39]); printf("Versao do firmware.: [%20.20s]\n", &aux[40]); printf("Versao da espec....: [%4.4s] \n", &aux[60]); printf("Versao da aplic....: [%16.16s]\n", &aux[64]); printf("Numero de serie....: [%20.20s]\n", &aux[80]); } ... </pre>
--	--

Dados das redes autorizadoras

Para algumas autorizadoras, o pinpad uma aplicação específica, cujos dados são retornadas pelo pinpad através da função ScopePPGetInfo, no formato abaixo.

Posição	Formato	Descrição
001-020	A20	Nome da rede adquirente (com espaços à direita)
021-033	A13	Versão da aplicação da rede adquirente, no formato "VVV.VV AAMMDD"
034-040	A7	Informações proprietárias da rede adquirente
041-042	A2	Tamanho em bytes dos dados a seguir ("00" a "yy")
043-???	Hxx(Byy)	Dados binários de identificação do SAM, caso existente, no layout exigido pela rede adquirente

Dados do dispositivo

O formato das informações do pinpad é padrão e segue o formato a seguir.

Posição	Formato	Descrição
001-020	A20	Nome do fabricante do pinpad
021-039	A19	Modelo / versão do hardware, no formato "xxx...xxx;mmm", onde "xxx" é o Nome do equipamento e "mmm" a capacidade de memória ("512KB", "1MB", "2MB", ...)
040	A1	Se o pinpad suporta cartão com chip sem contato, este campo deve conter a letra "C", caso contrário um espaço em branco.
041-060	A20	Versão do firmware (formato livre)
061-064	A4	Versão da especificação, no formato "V.VV" (neste caso, fixo em "1.05")
065-080	A16	Versão da aplicação básica, no formato "VVV.VV AAMMDD" (com 3 espaços à direita)
081-100	A20	Número de série do pinpad (com espaços à direita)

Obter informações do pinpad (ABECS)

A função ScopePPGetInfoEx substitui a função anterior.

Protótipo	LONG ScopePPGetInfoEx(WORD IdSaida, WORD DadosLen, char* Dados)	
Parâmetros de entrada	IdSaida	Informa a opção do formato dos dados de saída. O único valor permitido é o PC_GETINFO_HW
	DadosLen	Tamanho da área reservada pelo aplicativo, indicado pela variável Dados, para receber os dados
Parâmetros de saída	Dados	Dados do pinpad, conforme o formato descrito abaixo.
Retornos previstos	PC_OK	
	PC_NAO_ABERTO_APP	
	PC_MEMORIA_INSUFICIENTE	
	RCS_ERRO_PARM_1	

Exemplo	<pre> ... char info[255]; retorno = ScopePPGetInfoEx(PC_GETINFO_HW, sizeof(info), info); if (retorno == PC_OK) { // mostra as informações } ... </pre>
----------------	--

O parâmetro Dados segue o seguinte formato.

Posição	Formato	Descrição
001-020	A20	Nome do fabricante
021-040	A20	Modelo / versão do hardware, no formato "xx...xx;m...m", onde: "xx...xx" é o nome do equipamento, e; "m...m" a capacidade de memória ("512KB", "1MB", "2MB", ...)
041-060	A20	Versão do firmware (formato livre) Versão do software básico ou sistema operacional
061-064	A4	Versão da especificação, no formato "V.VV"
065-080	A16	Versão da aplicação básica, no formato "VVV.VV AAMMDD" (com 3 espaços à direita)
081-100	A20	Número de série (formato livre)
101-110	A10	(*)Capacidades do pinpad: "Oxxxxxxxxx" = Não suporta CTLS "1xxxxxxxxx" = Suporta CTLS "x0xxxxxxxx" = Não possui visor gráfico "x1xxxxxxxx" = Possui visor gráfico monocromático "x2xxxxxxxx" = Possui visor gráfico colorido "xx00000000" = RUF
111-114	N4	(*)Número máximo de linhas e colunas do visor do pinpad para apresentação de mensagens em modo texto (formato "LLCC").
115-122	N8	(*)Número máximo de linhas e colunas do visor gráfico do pinpad para apresentação de imagens (formato "LLLLCCCC", em pixels).
123-142	A20	(*)Tipos de arquivo multimídia suportados: "1xxx..." = Suporta o formato PNG; "x1xx..." = Suporta o formato JPG.

(*) Válidos apenas para o pinpad ABECS

Os últimos quatro campos são válidos para pinpads ABECS. Como os pinpads compartilhados não fornecem esse tipo de informação, o conteúdo deles terão os valores indicados abaixo.

Campo	Conteúdo do pinpad compartilhado
Capacidades do pinpad	"Oxxxxxxxxx" = Não suporta CTLS "1xxxxxxxxx" = Suporta CTLS "x0xxxxxxxx" = Não possui visor gráfico "xx00000000" = RUF
Número máximo de linhas e colunas do visor do pinpad para apresentação de mensagens em modo texto	Fixo com o valor "0216"
Número máximo de linhas e colunas do visor gráfico do pinpad para apresentação de imagens	Fixo com o valor "00000000"
Tipos de arquivo multimídia suportados	Fixo com o valor "00000000000000000000000000000000"

Obter o modo de operação do pinpad

Por motivo de compatibilidade, este conjunto de funções atende pinpads compartilhados e pinpads ABECS. Além disso, o próprio pinpad ABECS é compatível com o protocolo de pinpad compartilhado, isto é, ele também opera como se fosse um pinpad compartilhado. Embora este processo seja transparente para a automação comercial, se houver a necessidade de um tratamento específico, a automação comercial pode saber qual é o modo de operação que o SCOPE está se comunicando com o pinpad usando a função ScopePPGetOperationMode.

Protótipo	LONG ScopePPGetOperationMode(BYTE* Mode)	
Parâmetros de entrada	Não há	
Parâmetros de saída	Mode	Endereço para um espaço de um byte que receberá o modo de operação, que pode ser: PC_MODO_NONE – nenhum PC_MODO_COMPART – pinpad compartilhado PC_MODO_ABECS – pinpad ABECS
Retornos previstos	PC_OK PC_NAO_ABERTO_APP PC_COMMERR	
Exemplo	<pre>... BYTE ModoOperacao; retorno = ScopePPGetOperationMode(&ModoOperacao); if (retorno == PC_OK) { switch (ModoOperacao) { case PC_MODO_COMPART: // pinpad compartilhado ... break; case PC_MODO_ABECS: // pinpad ABECS ... break; } }</pre>	

Comandos

Nos tópicos a seguir, serão abordadas as funções para uso de recursos do pinpad.

Exibir de mensagens de tamanho fixo no visor

Para personalização da aplicação de frente de loja, a aplicação pode enviar mensagens para o pinpad, desde que o SCOPE não esteja processando alguma transação.

A exibição de mensagens é o único recurso que a automação comercial pode usar mesmo que o pinpad seja de uso exclusivo do SCOPE.

Protótipo	LONG ScopePPDisplay(char* Msg)
-----------	--------------------------------

Parâmetros de entrada	Msg	Mensagem de 32 caracteres (2 linhas x 16 colunas) a ser apresentada no display do pinpad. Não é permitido caracteres de quebra de linha.
Parâmetros de saída	Não há	
Retornos previstos	PC_OK	
	PC_NAO_ABERTO_APP	
	PC_PINPAD_NAO_CONFIGURADO	
	PC_DISPLAY_NAO_PERMITIDO	
Exemplo	<pre>... retorno = ScopePPDisplay("PROXIMO CLIENTE SEJA BEMVINDO "); ...</pre>	

Exibir de mensagens de formato livre no visor

Opcionalmente, a automação comercial pode enviar mensagens de tamanho livre usando a função ScopePPDisplayEx. Essa função segue a mesma regra da ScopePPDisplayEx em que a automação comercial pode usar mesmo que o pinpad seja de uso exclusivo do SCOPE e desde que o SCOPE não esteja processando alguma transação.

Protótipo	LONG ScopePPDisplayEx(BYTE LenMsg, char* Msg)	
Parâmetros de entrada	LenMsg	Tamanho da mensagem com o tamanho de até 999.
	Msg	Mensagem a ser apresentada, podendo conter caracteres de controle aceitos pelo display do pinpad, tal como o CR (0Dh) para quebra de linha.
Parâmetros de saída	Não há	
Retornos previstos	PC_OK	
	PC_NAO_ABERTO_APP	
	PC_PINPAD_NAO_CONFIGURADO	
	PC_DISPLAY_NAO_PERMITIDO	
Exemplo	<pre>... retorno = ScopePPDisplayEx(266, "ÁRIES: Lua Crescente estimula um" \ "envolvimento mais intenso com o " \ "trabalho e a saúde, ariano. É uma " \ "fase importante para reconhecer as " \ "suas responsabilidades e necessidade " \ "de aprimoramento. É preciso que aja " \ "discernimento e maturidade para " \ "alcançar os seus objetivos."); ...</pre>	

O pinpad tratará de exibir a mensagem, mas parte da mensagem poderá ser suprimida.

Aguardar o pressionando de uma tecla

O par de funções ScopePPStartGetKey / ScopePPGetKey captura uma tecla não-numérica pressionada no pinpad. Elas fazem parte do conjunto de funções não-blocantes.

A função ScopePPStartGetKey inicia o processo colocando o pinpad em modo de espera de uma tecla não-numérica pressionada. Uma vez que o pinpad está em modo de captura de tecla, a função ScopePPGetKey deverá ser chamada continuamente até que o retorno dela seja diferente de 1. Enquanto o retorno for igual a 1, o processo pode ser cancelado pela automação comercial chamando a função ScopePPAbort.

Protótipo	LONG ScopePPStartGetKey()
Parâmetros de entrada	Não há
Parâmetros de saída	Não há
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN PC_COMMERR PC_NAO_ABERTO_APP
Protótipo	LONG ScopePPStartGetKey()
Parâmetros de entrada	Não há
Parâmetros de saída	Não há
Retornos previstos	PC_OK PC_INVCALL PC_PROCESSING PC_F1 PC_F2 PC_F3 PC_F4 PC_BACKSP PC_CANCEL PC_TIMEOUT_USER
Exemplo	<pre> ... LONG retorno = 0; ... if (ScopePPStartGetKey() == PC_OK) // se OK { printf("\nAguardando tecla..."); do { retorno = ScopePPGetKey(); } while (retorno == PC_PROCESSING); switch (retorno) { case PC_OK: printf("\nPressionada a tecla <OK>\n"); break; case PC_F1: printf("\nPressionada a tecla de funcao 1\n"); break; case PC_F2: printf("\nPressionada a tecla de funcao 2\n"); break; case PC_F3: printf("\nPressionada a tecla de funcao 3\n"); break; case PC_F4: printf("\nPressionada a tecla de funcao 4\n"); break; case PC_BACKSP: printf("\nPressionada a tecla <Limpa>\n"); break; case PC_CANCEL: ... } } </pre>

	<pre> printf("\nPressionada a tecla <Cancela>\n"); break; default: printf("\nERRO: = %d\n", retorno); break; } } ... </pre>
--	---

Ler um cartão

A captura de cartão se dá com o uso das funções não-blocantes: ScopePPStartGetCard / ScopePPGetCard. A função ScopePPStartGetCard é a que inicia o processo de leitura do cartão, seja ele magnético ou com chip. A função ScopePPGetCard finaliza o processo iniciado por ScopePPStartGetCard e deve ser chamada diversas vezes enquanto retornar PC_PROCESSING ou PC_NOTIFY. Enquanto nessa situação, o processo pode ser cancelado pela automação comercial através da função ScopePPAbort.

Quando a função ScopePPGetCard retornar o código igual a PC_NOTIFY, o pinpad colocou uma mensagem no segundo parâmetro da função e a aplicação deve exibir-la na tela. Logo após a exibição da mensagem, a aplicação deve continuar chamando esta função até que esta não retorne mais um dos dois valores relacionados acima.

Protótipo	LONG EXPORT ScopePPStartGetCard(WORD TipoApl, char* ValorInicial)	
Parâmetros de entrada	TipoApl	Identifica o tipo de aplicação desejada (crédito = 1, débito = 2 e para qualquer aplicação = 99).
	ValorInicial	Valor inicial da transação. Podendo ser 0 (zero) caso este dado não esteja disponível. Este campo deverá ter tamanho máximo de 12 (doze) bytes, sendo 2 (duas) casas decimais.
Parâmetros de saída	Não há	
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN PC_COMMERR PC_NAO_ABERTO_APP	
	Protótipo LONG EXPORT ScopePPGetCard (WORD Id, char* MsgNotify, WORD Len, char* Dados)	
Parâmetros de entrada	Id	Informa a opção do formato dos dados de saída, podendo ser: 0 – formato padrão 1 – formato com a trilha 3
	Len	Tamanho do buffer alocado para os dados.
Parâmetros de saída	MsgNotify	Mensagem de 32 caracteres a ser apresentada no "checkout" caso a função retorne PP_NOTIFY.
	Dados	Dados do cartão conforme o que foi solicitado em Campos, sequencialmente. O formato de saída segue o padrão abaixo.
Retornos previstos	PC_OK PC_INVCALL PC_PROCESSING PC_CANCEL PC_TIMEOUT_USER	
	Exemplo	
	<pre> ... char crt[256]; </pre>	

	<pre> char msg [256]; LONG retorno = 0; ... if (ScopePPStartGetCard(99, "000") == 0) { printf("\nAguardando inserir ou passar o cartao..."); do { retorno = ScopePPGetCard(0, msg, sizeof(crt), crt); if (retorno == 2) { printf("\nMensagem\n[%s]\n", mensagem); } } while (retorno == 1); if (retorno == 0)// se OK { printf("\nOs dados do cartão [%s]\n", cartao); } } ...</pre>
--	--

Ao retornar o valor PC_OK, o último parâmetro da função estará preenchido com os dados do cartão que foram solicitados. Abaixo segue o formato para o Id igual à 0 (zero).

001-002	N2	Tipo de cartão lido: "00" – Magnético "01" – Moedeiro VisaCash sobre TIBC v1 "02" – Moedeiro VisaCash sobre TIBC v3 "03" – EMV "04" – Easy-Entry sobre TIBC v1
003-004	N2	Tamanho da trilha 1
005-080	A76	Trilha 1 (sem as sentinelas e com o byte de formato – primeiro caractere alfanumérico), alinhada à esquerda com espaços à direita.
081-082	N2	Tamanho da trilha 2
083-119	A37	Trilha 2 (sem as sentinelas), alinhada à esquerda com espaços à direita.
120-121	N2	Tamanho do PAN
122-139	A19	PAN, alinhado à esquerda com espaços à direita
140-166	A26	Nome do proprietário do cartão, com espaços à direita

O próximo formato é para o Id igual à 1 (um).

Posição	Formato	Descrição
001-002	N2	Tipo de cartão lido: "00" – Magnético "01" – Moedeiro VisaCash sobre TIBC v1 "02" – Moedeiro VisaCash sobre TIBC v3 "03" – EMV "04" – Easy-Entry sobre TIBC v1
003-004	N2	Tamanho da trilha 1
005-080	A76	Trilha 1 (sem as sentinelas e com o byte de formato – primeiro caractere alfanumérico), alinhada à esquerda com espaços à direita.
081-082	N2	Tamanho da trilha 2
083-119	A37	Trilha 2 (sem as sentinelas), alinhada à esquerda com espaços à direita.
120-122	N3	Tamanho da trilha 3
123-226	A104	Trilha 3 (sem as sentinelas), alinhada à esquerda com espaços à direita.

227-228	N2	Tamanho do PAN
229-247	A19	PAN, alinhado à esquerda com espaços à direita
248-273	A26	Nome do proprietário do cartão, com espaços à direita

Solicitar a digitação de um PIN aberto (obsoleto)

O conjunto ScopePPStartGetPIN / ScopePPGetPIN, também não-blocantes, é responsável pela captura da senha do usuário. A senha é criptografada pelo pinpad com a Master Key DES da NCR, e será retornada sem criptografia pela função ScopePPGetPIN, em caso de sucesso. A Master Key DES da NCR está inserida, por padrão, nos pinpad homologados pela Cielo.

OBSOLETO: Essas funções de captura de PIN serão descontinuadas devido a segurança. Elas funcionarão enquanto o pinpad for compartilhado (especificação 1.07^a ou 1.08^a) e este contiverem a Máster Key DES da NCR. Se conectado num pinpad ABECS, essas funções retornarão erro.

Como a informação digitada no pinpad é aberta e é possível passar a mensagem a ser exibida no display do pinpad, a aplicação pode utilizá-las para coletar apenas informações de domínio da aplicação. Entretanto, o uso dessas funções para tal finalidade é desaconselhada e cria um ambiente inseguro para os cliente.

Protótipo	LONG EXPORT ScopePPStartGetPIN (char *MsgDisplay)	
Parâmetros de entrada	MsgDisplay	Mensagem que será exibida no Pinpad. Esta mensagem deve ter no máximo 32 caracteres sendo duas linhas de 16 caracteres.
Parâmetros de saída	Não há	
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN PC_COMMERR PC_NAO_ABERTO_APP	
Protótipo	LONG ScopePPGetPIN(char* PIN)	
Parâmetros de entrada	Não há	
Parâmetros de saída	PIN	Senha capturada do usuário (já descriptografada), finalizada com caractere nulo ('\0').
Retornos previstos	PC_OK PC_INVCALL PC_PROCESSING PC_CANCEL PC_TIMEOUT_USER PC_ERRPIN	
Exemplo	<pre>... char pin[40]; LONG retorno = 0; ... if (ScopePPStartGetPIN("DIGITE A SENHA") == 0) { printf("\nAguardando digitação da senha..."); do { retorno = ScopePPGetPIN(pin); } while (retorno == 1); if (retorno == 0)// se OK printf("\nA senha digitada é [%s]\n", pin);</pre>	

	{ ...}
--	-----------

Solicitar a digitação de um PIN (seguro)

Coleta de um PIN é feita pelas funções não-blocantes ScopePPStartGetPINEx e ScopePPGetPINEx. A função ScopePPStartGetPINEx é a que inicia o processo de leitura do cartão, seja ele magnético ou com chip. A função ScopePPGetPINEx finaliza o processo iniciado por ScopePPStartGetPINEx e deve ser chamada diversas vezes enquanto retornar PC_PROCESSING. Enquanto nessa situação, o processo pode ser cancelado pela automação comercial através da função ScopePPAbort.

A mensagem de solicitação de PIN apresentada no pinpad é formatada pelo próprio dispositivo.

Protótipo	LONG ScopePPStartGetPIN(char* Msg, WORD Mode, WORD Mk, char* Wk, char* Pan)	
Parâmetros de entrada	Msg	Parâmetro não utilizado. A aplicação deve enviar uma string vazia.
	Mode	Modo de criptografia, que pode ser PP_GETPIN_MODO_DES (DES) ou PP_GETPIN_MODO_3DES (Triple DES)
	Mk	Índice da Master Key Triple DES. Mesmo que o parâmetro Mode seja PP_GETPIN_MODO_DES, o único valor previsto é PP_GETPIN_MK_3DES_OKI, pois o SCOPE sabe onde as chaves DES da NCR estão em cada fabricante/modelo de pinpad suportado.
	Wk	String hexadecimal representando a Working Key criptografada pela Master Key indicada em Mk. Se o parâmetro Mode for PP_GETPIN_MODO_DES, então este parâmetro teve ter tamanho 16 caracteres. Se o parâmetro Mode for PP_GETPIN_MODO_3DES, então este parâmetro teve ter tamanho 32 caracteres. A working key criptografada pode ser vazia ou nula se ela estiver configurada no arquivo scope.ini.
	Pan	String representando o PAN contendo no mínimo 2 e máximo 19 dígitos, alinhado à esquerda e espaços à direita. Se não for fornecido, será usado o PAN fixo "111111111111111111"
Parâmetros de saída	Não há	
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN PC_COMMERR PC_NAO_ABERTO_APP RCS_ERRO_PARM_2 RCS_ERRO_PARM_3 RCS_ERRO_PARM_4 RCS_ERRO_PARM_5	
Protótipo	LONG ScopePPGetPIN(char* PIN)	
Parâmetros de entrada	PIN	String hexadecimal de 16 caracteres representando o PIN criptografado
Parâmetros de saída	Não há	
Retornos previstos	PC_OK PC_INVCALL PC_PROCESSING PC_CANCEL	

	PC_TIMEOUT_USER PC_ERRPIN
Exemplo	<pre> ... char pin[16 + 1]; LONG retorno = 0; ... if (ScopePPStartGetPIN("", PP_GETPIN_MODO_3DES, PP_GETPIN_MK_3DES_OKI, "025D41065E30376F8A23B83556B1CAC1", "") == PC_OK) { do { // aguardando a digitação da senha retorno = ScopePPGetPIN(pin); } while (retorno == PC_PROCESSING); } ... </pre>

Solicitar a digitação de um dado

Com o pinpad ABECS, a automação comercial pode solicitar a digitação de certos dados no pinpad. A automação só precisa indicar qual dado deve ser solicitado. Os dados disponíveis para a digitação no pinpad são:

Valor (hexa)	Campo (scopeapi.h)	Mensagem	Versão ABECS Mínima
0001h	PC_GETDATA_DDD	DIGITE O DDD	2.03
0002h	PC_GETDATA_REDIGITE_DDD	REDIGITE O DDD	2.03
0003h	PC_GETDATA_TEL	DIGITE O TELEFONE	2.03
0004h	PC_GETDATA_REDIGITE_TEL	REDIGITE O TELEFONE	2.03
0005h	PC_GETDATA_DDD_TEL	DIGITE DDD+TELEFONE	2.03
0006h	PC_GETDATA_REDIGITE_DDD_TEL	REDIGITE DDD+TELEFONE	2.03
0007h	PC_GETDATA_CPF	DIGITE O CPF	2.03
0008h	PC_GETDATA_REDIGITE_CPF	REDIGITE O CPF	2.03
0009h	PC_GETDATA_RG	DIGITE O RG	2.03
000Ah	PC_GETDATA_REDIGITE_RG	REDIGITE O RG	2.03
000Bh	PC_GETDATA_4_ULTIMOS	DIGITE OS 4 ÚLTIMOS DÍGITOS	2.03
000Ch	PC_GETDATA_COD_SEGURANCA	DIGITE CÓDIGO DESEGURANÇA	2.03
000Dh	PC_GETDATA_CNPJ	DIGITE O CNPJ	2.12
000Eh	PC_GETDATA_REDIGITE_CNPJ	REDIGITE O CNPJ	2.12
000Fh	PC_GETDATA_DATA_DDMMAAAA	DIGITE A DATA (DDMMAAAA)	2.12
0010h	PC_GETDATA_DATA_DDMMAA	DIGITE A DATA (DDMMAA)	2.12
0011h	PC_GETDATA_DATA_DDMM	DIGITE A DATA (DDMM)	2.12
0012h	PC_GETDATA_DIA_DD	DIGITE O DIA (DD)	2.12
0013h	PC_GETDATA_MES_MM	DIGITE O MÊS (MM)	2.12
0014h	PC_GETDATA_ANO_AA	DIGITE O ANO (AA)	2.12
0015h	PC_GETDATA_ANO_AAAA	DIGITE O ANO (AAAA)	2.12
0016h	PC_GETDATA_DT_NASC_DDMMAAAA	DATA DE NASCIMENTO(DDMMAAAA)	2.12

0017h	PC_GETDATA_DT_NASC_DDMMAA	DATA DE NASCIMENTO(DDMMAA)	2.12
0018h	PC_GETDATA_DT_NASC_DDMM	DATA DE NASCIMENTO(DDMM)	2.12
0019h	PC_GETDATA_DIA_NASC_DD	DIA DO NASCIMENTO (DD)	2.12
001Ah	PC_GETDATA_MES_NASC_MM	MÊS DO NASCIMENTO (MM)	2.12
001Bh	PC_GETDATA_ANO_NASC_AA	ANO DO NASCIMENTO (AA)	2.12
001Ch	PC_GETDATA_ANO_NASC_AAAA	ANO DO NASCIMENTO (AAAA)	2.12
001Dh	PC_GETDATA_IDENTIFICACAO	DIGITE IDENTIFICAÇÃO	2.12
001Eh	PC_GETDATA_COD_FIDELIDADE	CÓDIGO DE FIDELIDADE	2.12
001Fh	PC_GETDATA_NUMERO_MESA	NÚMERO DA MESA	2.12
0020h	PC_GETDATA_QTDE_PESSOAS	QUANTIDADE DE PESSOAS	2.12
0021h	PC_GETDATA_QUANTIDADE	DIGITE QUANTIDADE	2.12
0022h	PC_GETDATA_NUMERO_BOMBA	NÚMERO DA BOMBA	2.12
0023h	PC_GETDATA_NUMERO_VAGA	NÚMERO DA VAGA	2.12
0024h	PC_GETDATA_NUMERO_CAIXA	NÚMERO DO GUICHÊ/CAIXA	2.12
0025h	PC_GETDATA_COD_VENDEDOR	CÓDIGO DO VENDEDOR	2.12
0026h	PC_GETDATA_COD_GARCOM	CÓDIGO DO GARÇOM	2.12
0027h	PC_GETDATA_NOTA_ATENDIMENTO	NOTA DO ATENDIMENTO	2.12
0028h	PC_GETDATA_NUMERO_NF	NÚMERO DA NOTA FISCAL	2.12
0029h	PC_GETDATA_NUMERO_COMANDA	NÚMERO DA COMANDA	2.12
002Ah	PC_GETDATA_PLACA_VEICULO	PLACA DO VEÍCULO	2.12
002Bh	PC_GETDATA QUILOMETRAGEM	DIGITE QUILOMETRAGEM	2.12
002Ch	PC_GETDATA QUILOMETRAGEM_INICIAL	QUILOMETRAGEM INICIAL	2.12
002Dh	PC_GETDATA QUILOMETRAGEM_FINAL	QUILOMETRAGEM FINAL	2.12
002Eh	PC_GETDATA PORCENTAGEM	DIGITE PORCENTAGEM	2.12
002Fh	PC_GETDATA_PESQUISA_SATISFACAO	PESQUISA DE SATISFAÇÃO (0 a10)	2.12
0030h	PC_GETDATA_AVALIE_ATENDIMENTO	AVALIE ATENDIMENTO (0 a 10)	2.12
0031h	PC_GETDATA_DIGITE_TOKEN	DIGITE O TOKEN	2.12
0032h	PC_GETDATA_DIGITE_NUMERO_CARTAO	DIGITE NÚMERO DO CARTÃO	2.12
0033h	PC_GETDATA_NUMERO_PARCELAS	NÚMERO DE PARCELAS	2.12
0034h	PC_GETDATA_CODIGO_DO_PLANO	CÓDIGO DO PLANO	2.12
0035h	PC_GETDATA_CODIGO_DO_PRODUTO	CÓDIGO DO PRODUTO	2.12

Protótipo

```
LONG ScopePPStartGetData(WORD Dado,
                         WORD TamMin,
                         WORD TamMax)
```

Parâmetros de entrada	Dado	Dado a ser solicitado no pinpad
	TamMin	Quantidade mínima de dígitos a ser capturada. No modo PC_MODO_ABECS, o valor for 0 (zero) permite uma entrada vazia. No modo PC_MODO_COMPART, se o valor for 0 (zero), assume-se o valor 4 (zero), não permitindo uma entrada vazia, pois é uma restrição dos pinpads nesse modo.

	TamMax	Quantidade máxima de dígitos a ser capturada. Se o valor for 0 (zero), será assumido o valor máximo permitido. Para valores diferentes de 0 (zero), o valor deve ser maior ou igual ao TamMin. No modo PC_MODO_ABECS, o valor máximo que é 32. No modo PC_MODO_COMPART, o valor máximo que é 12.
Parâmetros de saída	Não há	
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN PC_COMMERR PC_NAO_ABERTO_APP RCS_ERRO_PARM_1 RCS_ERRO_PARM_2 RCS_ERRO_PARM_3	
Protótipo	LONG ScopePPGetData(WORD LenDados, char* Dados)	
Parâmetros de entrada	LenDados	Tamanho do buffer alocado para os dados
Parâmetros de saída	Dados	String com o valor digitado no pinpad
Retornos previstos	PC_OK PC_INVCALL PC_PROCESSING PC_CANCEL PC_TIMEOUT_USER PC_ERRPIN	
Exemplo	<pre> ... char dado[32 + 1]; LONG retorno = 0; ... // solicita o CPF, permitindo entrada vazia if (ScopePPStartGetData(PC_GETDATA_CPF, 0, 11) == PC_OK) { do { // Aguardando digitar o CPF retorno = ScopePPGetData(sizeof(dado), dado); } while (retorno == PC_PROCESSING); if (retorno == PC_OK) // se OK { // usa o dado digitado } } ... </pre>	

IMPORTANTE: Por motivo de segurança, o uso das funções ScopePPStartGetPin e ScopePPGetPin serão descontinuadas no futuro.

Solicitar um item de um menu

Além de solicitar a digitação de um dado, outro recurso disponível é o uso de menu no pinpad para a seleção do usuário. Esse recurso está disponível apenas no modo PC_MODO_ABECS.

IMPORTANTE: Essa função só está disponível para o pinpad ABECS.

As funções ScopePPStartOptionMenu e ScopePPGetOptionMenu fazem com que o pinpad apresente em seu display um menu de até 20 opções para escolha do usuário, utilizando da melhor maneira os seus recursos de hardware. Para utilizar isso, deve-se seguir as seguintes regras:

- Cada opção pode ter no máximo 24 caracteres.
- O pinpad apresentará o menu de opções sempre respeitando a ordem em que foram fornecidas.
- Caso a opção seja iniciada por um caractere numérico ("0" a "9"), o pinpad poderá permitir a seleção através do teclado ("hot key"), mediante o pressionamento da tecla correspondente ao caractere. Caso a automação comercial opte por usar este recurso, cabe a ela garantir a integridade das opções para que não haja repetição.
- A automação deve fornecer ao menos uma opção.

Protótipo	LONG ScopePPStartOptionsMenu(char* Titulo, char* Lista)	
Parâmetros de entrada	Titulo	String com o título do menu. Caso não seja fornecido, o menu é apresentado sem título.
	Lista	Lista com as opções do menu.
Parâmetros de saída	Não há	
Retornos previstos	PC_OK	
	PC_INVCALL	
	PC_NOTOPEN	
	PC_COMMERR	
	PC_NAO_ABERTO_APP	
	RCS_ERRO_PARM_1	
	RCS_ERRO_PARM_2	
Protótipo	LONG ScopePPOptionMenu(char* Índice)	
Parâmetros de entrada	Não há.	
Parâmetros de saída	Índice	String com o índice de dois dígitos numéricos referente à opção de menu selecionada, considerando-se ordem em que foram fornecidas pela automação comercial (a partir de "01").
Retornos previstos	PC_OK	
	PC_INVCALL	
	PC_PROCESSING	
	PC_CANCEL	
	PC_TIMEOUT_USER	
Exemplo	<pre>// é solicitado ao pinpad que apresente um menu com o // título "Selecione, por favor:" e as opções: // "5.Chamado Técnico" // "1.Consultas" // "3.Ajuda" // "Voltar!!" ... char* titulo = "Selecione, por favor:"; char* opcoes = "15.Chamado Técnico111.Consultas073.Ajuda08Voltar!!"; char escolhido[2 + 1]; LONG retorno = 0; ... if (ScopePPStartOptionsMenu(titulo, opcoes) == PC_OK) {</pre>	

	<pre> do { // Aguardando a seleção de uma opção retorno = ScopePPGetOptionMenu(escolhido); } while (retorno == PC_PROCESSING); ... </pre>
--	---

As opções da lista devem seguir o seguinte formato:

Dado	Formato	Descrição
Primeira opção	N2	Tamanho do texto da primeira opção
	A..24	Texto para a primeira opção do menu (índice "01").
Segunda opção	N2	Tamanho do texto da segunda opção
	A..24	Texto para a primeira opção do menu (índice "02").
...
Última opção	N2	Tamanho do texto da última opção
	A..24	Texto para a última opção do menu (índice "xx", onde xx é o número total de opções fornecidas).

Ler uma comanda

Recurso para ler uma comanda de restaurante com chip.

Protótipo	LONG ScopePPStartGetComanda(WORD Tipo)	
Parâmetros de entrada	Tipo	Tipo de comanda. Atualmente existe apenas o tipo PP_COMANDA_TIPO_CAFE
Parâmetros de saída	Não há	
Retornos previstos	PC_OK	
	PC_INVCALL	
	PC_NOTOPEN	
	PC_COMMERR	
	PC_NAO_ABERTO_APP	
	RCS_ERRO_PARM_1	
Protótipo	LONG ScopePPGetComanda(WORD Tipo, WORD LenComanda, char *Comanda, WORD RemoveComanda)	
Parâmetros de entrada	Tipo	Tipo de comanda. Atualmente existe apenas o tipo PP_COMANDA_TIPO_CAFE
	LenComanda	Tamanho da área reservada para Comanda
	RemoveComanda	Indica se, após a leitura, deverá ser solicitada a retirada (PP_COMANDA_RETIRAR) da comanda ou se deverá ser mantida (PP_COMANDA_MANTER) para posterior limpeza.
Parâmetros de saída	Comanda	Ponteiro, reservado pela automação, da área onde será gravado o conteúdo da comanda
Retornos previstos	PC_OK	
	PC_INVCALL	
	PC_PROCESSING	
	PC_CANCEL	
	PC_TIMEOUT_USER	
	PC_MEMORIA_INSUFICIENTE	
	PC_COMANDA_VAZIA	
	PC_COMANDA_INVALIDA	
Exemplo	...	

	<pre> char comanda[4096+1]; LONG Retorno; ... if(ScopePPStartObtemComanda(PP_COMANDA_TIPO_CAFE)==PC_OK) { // aguardando a comanda memset(comanda, 0x00, sizeof(comanda)); do { // Aguardando a seleção de uma opção retorno = ScopePPObtémComanda(PP_COMANDA_TIPO_CAFE, sizeof(comanda), comanda, PP_COMANDA_RETIRAR); } while (retorno == PC_PROCESSING); } ...</pre>
--	--

Limpar uma comanda

Quando necessário, a automação deve usar as funções ScopePPStartLimpaComanda e ScopePPLimpaComanda para apagar o conteúdo da comanda.

Protótipo	LONG ScopePPStartLimpaComanda(WORD Tipo)	
Parâmetros de entrada	Tipo	Tipo de comanda. Atualmente existe apenas o tipo PP_COMANDA_TIPO_CAFE
Parâmetros de saída	Não há	
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN PC_COMMERR PC_NAO_ABERTO_APP RCS_ERRO_PARM_1	
Protótipo	LONG ScopePPLimpaComanda(WORD Tipo)	
Parâmetros de entrada	Tipo	Tipo de comanda. Atualmente existe apenas o tipo PP_COMANDA_TIPO_CAFE
Parâmetros de saída	Não há	
Retornos previstos	PC_OK PC_INVCALL PC_PROCESSING PC_CANCEL PC_TIMEOUT_USER PC_MEMORIA_INSUFICIENTE PC_COMANDA_VAZIA PC_COMANDA_INVALIDA	
Exemplo	<pre> ... char comanda[4096+1]; LONG Retorno; ... if(ScopePPStartLimpaComanda(PP_COMANDA_TIPO_CAFE)==PC_OK) { // aguardando a comanda </pre>	

	<pre> memset(comanda, 0x00, sizeof(comanda)); do { // Aguardando a seleção de uma opção retorno=ScopePPLimpaComanda(PP_COMANDA_TIPO_CAFE); } while (retorno == PC_PROCESSING); } ... </pre>
--	---

Interromper um processamento no pinpad

As funções não-blocantes podem ter seu processamento interrompido pela automação comercial a qualquer momento. A interrupção é feita pela função ScopePPAbort.

Protótipo	LONG ScopePPAbort()
Parâmetros de entrada	Não há
Parâmetros de saída	Não há
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN
Exemplo	<pre> ... if (ScopePPStartXXX(...) == PC_OK) { do { if /* deve interromper*/ ScopePPAbort(); retorno = ScopePPXXX(...); } while (retorno == PC_PROCESSING); } ... </pre>

Obter a mensagens de erro

Esta função retorna a descrição referente ao código de erro informado retornado pelas funções de acesso ao pinpad.

Protótipo	LONG EXPORT ScopePPMsgErro (LONG RC, char *MsgErro)	
Parâmetros de entrada	RC	Código do erro.
Parâmetros de saída	MsgErro	Mensagem referente ao código de erro informado.
Retornos previstos	PC_OK PC_INVCALL PC_NOTOPEN	
Exemplo	<pre> ... // alguma operação no pinpad if (retorno == 0) // se OK { printf("\nOs dados do cartão são [%s]\n", cartao); } else { ScopePPMsgErro(retorno, mensagem); printf("\n[%s]\n", mensagem); } </pre>	

	}
	...

Forçar a atualização de tabelas no PINPad

Esta função realiza a atualização de tabelas no pinpad sempre que for chamada.

Protótipo	LONG EXPORT ScopePPAtualizaTabelas ()
Retornos previstos	PC_OK
	PC_INVCALL
	PC_NOTOPEN
	PC_TABERR
Exemplo	<pre>... // abre comunicação com o pinpad if (ScopePPAtualizaTabelas == 0) // se OK { printf("Atualização de tabelas realizadas com sucesso"); } else { printf("Falha na atualização de tabelas"); } ScopePPDisplayEx("Oki Brasil"); ...</pre>

IMPORTANTE: Essa funcionalidade só deve ser utilizada em redes que possuem carga de tabelas.

IMPORTANTE: Logo a após a atualização de tabelas no pinpad, é necessário chamar a função ScopePPDisplayEx para atualizar o texto de exibição na tela do PINPad.

Funções Multimídia

A partir da versão de especificação **ABECS 2.12** e versão do Scope **3.01.17.005**, está disponível uma série de funções para uso em PINPads com recurso de multimídia (display gráfico colorido e/ou áudio). O suporte a estes comandos é opcional e depende dos recursos do equipamento.

Observe que esses comando somente funcionarão em pinpads ABECS. Além disso, o SCOPE deve estar configurado para trabalhar no modo ABECS para que a abertura do pinpad seja realizada no modo seguro.

Caso a aplicação esteja utilizando o comando ScopePPOpen para a abertura do pinpad, ele deve ser substituído pelo comando ScopePPOpenSecure, pois os comandos funcionam somente em modo seguro.

Estão previstos os seguintes formatos de arquivo, que podem ou não serem suportados pelo PINPad:

- Imagem PNG (Portable Network Graphics) de acordo com ISO/IEC 15948;
- Imagem JPG (ou JPEG) de acordo com ISO/IEC 10918;
- Imagem ou animação GIF (Graphics Interchange Format - CompuServe).

Observe sempre se a resolução da figura a ser utilizada está compatível com a resolução do equipamento para que a figura possa ser mostrada no visor corretamente.

As seguintes funções estão contempladas nessa versão:

Função	Descrição
ScopePPMMFileLoad	Envia um arquivo PNG, JPG ou GIF para o PINPad
ScopePPMMFileList	Lista os arquivos PNG, JPG ou GIF armazenados no PINPad
ScopePPMMFileDelete	Exclui um ou mais arquivos PNG, JPG ou GIF armazenados no PINPad
ScopePPMMDisplayImage	Mostra um arquivo PNG, JPG ou GIF no display do PINPad

Enviar um arquivo para o PINPad

Esta função realiza o envio de um arquivo PNG, JPG ou GIF para o PINPad. Ao recebê-lo, o PINPad confere os dados recebidos, aceitando ou não o arquivo.

Protótipo	LONG ScopePPMMFileLoad()
Parâmetros de entrada	LPABECS_PPFILE_INFO_ptInfoFile
Parâmetros de saída	Não há
Retornos previstos	PC_OK
	PC_NAO_ABERTO_APP
	RCS_ERRO_MODO_ABECS
	RCS_ERRO_TAMANHO_INVALIDO
	RCS_ERRO_NOME_ARQUIVO_PP_INVALIDO

	RCS_ERRO_TIPO_ARQUIVO_PP_INVALIDO RCS_ERRO_ARQUIVO_INVALIDO
Exemplo	<pre> ... stABECS_PPFILE_INFO info_file; info_file.Version = 1; memcpy(info_file.szFileNamePP, "ARQ00001", 8); memcpy(info_file.szFileName, "figura.jpg", 10); memcpy(info_file.szFilePath, "c:\\figuras", 10); info_file.cFileType = FILE_TYPE_JPG; if (ScopePPMMFileLoad (info_file) == PC_OK) { } ... </pre>

Estrutura utilizada no comando ScopePPMMFileLoad

Tipo	Nome do campo	Tamanho	Descrição
int	Version		Versão do layout. Fixo em 1
char	szFileNamePP	9	Nome do arquivo armazenado no PINpad (com terminação '\0')
char	cFileType	1	Tipo do arquivo: FILE_TYPE_PNG: '1' FILE_TYPE_JPG: '2' FILE_TYPE_GIF: '3'
char	szFileName	257	Nome do arquivo PNG, JPG ou GIF lido pelo SCOPE (com terminação '\0')
char	szFilePath	321	caminho de origem do arquivo PNG, JPG ou GIF (com terminação '\0')

Listar os arquivos armazenados no PINPad

Esta função retorna uma lista com os nomes dos arquivos carregados no PINPad pelo comando ScopePPMMFileLoad. Se não houver arquivos carregados, a função é bem-sucedida e a lista retornada é vazia. Não há ordem específica para a montagem da lista, dependendo exclusivamente das características de implementação PINPad.

Protótipo	LONG ScopePPMMFileList ()	
Parâmetros de entrada	Não há	
Parâmetros de saída	char *_ptFileList	Lista dos arquivos armazenados no PINpad
	int * _ptTamanho	Ponteiro para o tamanho da lista retornada
Retornos previstos	PC_OK PC_NAO_ABERTO_APP RCS_ERRO_MODO_ABECS	
Exemplo	<pre> ... int retorno; int nTamanho; int *ptTam = &nTamanho; if (_ptFileList != NULL) { retorno = ScopePPMMFileList(_ptFileList, ptTam); } else { retorno = ScopePPMMFileList(NULL, ptTam); } ... </pre>	

IMPORTANTE:

- O tamanho do arquivo armazenado no PINpad é fixo em 8 bytes.
- Se o primeiro parâmetro for NULL, a função retorna o tamanho da lista de arquivos no segundo parâmetro. O tamanho retornado é um múltiplo de 8, indicando que o nome do arquivo vem a cada 8 bytes,

Excluir um ou mais arquivos armazenados no PINPad

Esta função exclui um ou mais arquivos armazenados no pinpad.

Protótipo	LONG ScopePPMMFileDelete ()
Parâmetros de entrada	char *_ptFileNamePP
Parâmetros de saída	Não há
Retornos previstos	PC_OK PC_NAO_ABERTO_APP RCS_ERRO_MODO_ABEC5
Exemplo	<pre>... // Para apagar todos os arquivos Retorno = ScopePPMMFileDelete(ALL_FILES); // Para apagar um arquivo específico memcpy(szArquivo, "ARQ00001", 8); Retorno = ScopePPMMFileDelete(szArquivo); ...</pre>

IMPORTANTE: Essa função não retorna erro caso um ou mais arquivos estejam ausentes no PINpad..

Mostrar um arquivo armazenado no PINPad

Este comando apresenta no *display* um arquivo de imagem multimídia previamente carregado no PINpad. O conteúdo será centralizado no *display* caso suas dimensões sejam inferiores à capacidade do equipamento. O *display* do pinpad é previamente apagado, sendo que mensagens ou imagens anteriores não são mantidas. Este comando sempre retorna imediatamente (é não blocante), mesmo se o arquivo multimídia contiver animação (ou vídeo), que será apresentada enquanto o PINpad não recebe um novo comando.

Protótipo	LONG ScopePPMMDisplayImage ()
Parâmetros de entrada	char *_ptFileNamePP
Parâmetros de saída	Não há
Retornos previstos	PC_OK PC_NAO_ABERTO_APP

	RCS_ERRO_MODO_ABEC5
Exemplo	<pre>... memcpy(szArquivo, "ARQ00001", 8); Retorno = Retorno = ScopePPMMDDisplayImage(szArquivo); ...</pre>

Obter automaticamente a porta de conexão com o PINPad

Esta função retorna automaticamente a porta na qual o PINPad está conectado.

Protótipo	LONG ScopePPGetCOMPort()
Parâmetros de entrada	Não há
Parâmetros de saída	char *_ptComEndereco Porta ou endereço do computador no qual o PINpad se conectou
Retornos previstos	RCS_SUCESSO RCS_PP_NAO_ENCONTRADO
Exemplo	<pre>... char szComEndereco[48] = { 0 }; int Retorno = 0; RC = ScopePPGetCOMPort(szComEndereco); if (Retorno == RCS_PP_NAO_ENCONTRADO) {} if (Retorno == RCS_SUCESSO) {} ...</pre>

IMPORTANTE:

- O Windows retorna o número da COM associada ao PINPad. No Linux, o valor é uma string com a seguinte formatação `/dev/ttyXXXX` onde `XXXX` dependerá da distribuição (normalmente o valor atribuído é `/dev/ttyACM0`).
- Verifique se a opção AUTO foi escolhida na configuração realizada no Portal PSW.

Totalização de TEF

Esta sessão trata uma funcionalidade que o SCOPE disponibiliza para a aplicação de frente de loja: a totalização de TEF.

Relatório de TEF

Pensando na facilidade de fazer o balanço dos valores de venda realizados no checkout, o SCOPE Client provê à aplicação um relatório com informações do total de TEF confirmadas e canceladas por bandeira num certo período e que pode ser enviado para a impressora ou para outro meio de armazenamento. Estas informações podem ser levantadas por PDV ou por operador. No entanto, mesmo que “a aplicação se interesse” pelas informações relativas apenas por uma das opções (PDV ou operador), o SCOPE atualizará ambos. Portanto, não haverá otimização ao escolher apenas um dos dois.

No caso do PDV, o período escolhido para a totalização dos valores pode ser diário, semanal, mensal, etc., mas recomenda-se que abranja o período do movimento, isto é, ao iniciar o período num dia, seja zerado o totalizador e, no final do dia, o relatório seja obtido.

Quanto ao relatório por operador, aconselha-se que seja zerado o totalizador na entrada do operador ao caixa e obtenha-se o relatório, na saída do mesmo.

IMPORTANTE: o SCOPE não controla (identifica) qual operador está operando o PDV. É papel da aplicação o controle da entrada e saída de cada operador e consequentemente, iniciar o totalizador e obter o relatório no momento correto.

Iniciando os totais

No início do período de totalização, a aplicação deverá chamar a função ScopelniciaTotalTEF(), passando no parâmetro o código referente ao totalizador que se deseja zerar, conforme abaixo:

- valor 0: será zerados ambos os totalizadores, o do PDV e o do operador
- valor 1: será zerado apenas o do operador

Protótipo

```
LONG ScopelniciaTotalTEF (BYTE Nivel)
```

Parâmetros

[in]	BYTE	Nível	Indica a que nível deverá ser inicializado os totais de TEF (0: ambos, 1: apenas do operador)
------	------	-------	---

Retorno

Código	Descrição	Ação
0x0000	Sucesso na inicialização dos totalizadores	
0xFB07	Erro na totalização de TEF, causado por um dos motivos: parâmetro inválido; não foi possível criar o arquivo; não foi possível atualizar o arquivo; não foi possível ler o arquivo.	verifique o valor do parâmetro passado; verifique o parâmetro passado à função; verifique direitos de gravação, criação e leitura de arquivos para o usuário logado no PDV;

		verifique se existe espaço disponível no disco.
OxFEO0	Transação em andamento, o que impossibilita a execução desta função	Revisar a aplicação, pois, ela está chamando a função em momentos errado.

Ver [tabela de código de retorno](#).

Exemplo

```

...
// carrega configuração do PDV de um novo dia
...
// conecta ao SCOPE Server
...
// zera totalizador ao iniciar o período de totalização
// (por ex. ao iniciar o dia)
if (ScopelniciaTotalTEF (0) != 0) // zera ambos
{
    // erro ao zerar o arquivo
}
...
// Período transcorrido (por ex. até finalizar o dia)
...
// Ao final do período chamar uma das funções a seguir para obter o Total
...

```

Cupom dos totais

Não importa se é por PDV ou por operador, uma forma que o SCOPE entrega o relatório para a aplicação é um texto formatado, com 40 colunas, que pode ser enviado diretamente para a impressora, conforme exemplo abaixo:

CARTAO DE CREDITO		
BANDEIRA	EFETUADO	CANCELADO
VISA	167.00	0.00
MASTERCARD	564.80	0.00
CARTAO DE DEBITO		
BANDEIRA	EFETUADO	CANCELADO
CHEQUE ELETRONIC	1650.00	0.00
ELECTRON	18.00	4.00
CARTAO DE DEBITO (CDC)		
BANDEIRA	EFETUADO	CANCELADO
CONSULTA CHEQUE		
BANDEIRA	CONSULTADO	

CUIDADO: O valor efetuado contém o valor cancelado. Assim, no exemplo anterior, a bandeira Electron indica uma receita de R\$ 14,00 (R\$ 18,00 - R\$ 4,00) para a loja.

Protótipo

LONG ScopeObtemTotalTEF (BYTE Nivel, char *Cupom, WORD TamCupom, char SeparadorLinhasCupom)

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Parâmetros

[in]	BYTE	Nível	Indica a que nível deverá ser inicializado os totais de TEF (0: PDV, 1: operador)
[out]	String	Cupom	Cupom com o relatório de totais de TEF por bandeira
[in]	WORD	TamCupom	Tamanho da área que a aplicação alocou pela variável Cupom
[in]	char	SeparadorLinhasCupom	Caractere a ser usado na separação das linhas do cupom. Atualmente, suporta apenas o caractere '@'. Para qualquer outro caractere será utilizado '\n'.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFB07	0xF04263	64263 Erro na totalização de TEF

Exemplo

```

...
WORD TamCupom;
char Cupom[2048];
...
// encerrando o Dia
...
TamCupom = sizeof(Cupom);
if(ScopeObtemTotalTEF(0, Cupom, TamCupom, '\n') == 0)
{
    ...
    EnviaCupomParaImpressora(Cupom);
    ...
}
else
{
    // erro ao obter o cupom com o relatório
}
...

```

Dados dos totais

A outra maneira que o SCOPE fornece o relatório para a aplicação é por meio de um *buffer* com os campos de tamanho fixo. Há 3 tipos de *layouts* conforme as tabelas abaixo:

Layout Versão 1:

Tamanho	Descrição
3 bytes	Identificação do buffer ("R01")
2 bytes	Quantidade de bandeiras retornadas que indica a quantidade de repetições dos registros abaixo
1 byte	Identifica a função: 'C': Crédito 'D': Débito; 'H': Cheque 'A': CDC
2 bytes	Código da bandeira
17 bytes	Descrição da bandeira
14 bytes	Valor efetivado (2 casas decimais)
14 bytes	Valor cancelado (2 casas decimais)

Layout Versão 2:

Tamanho	Descrição
3 bytes	Identificação do buffer ("R02")
3 bytes	Quantidade de bandeiras retornadas que indica a quantidade de repetições dos registros abaixo
3 bytes	Código da bandeira
17 bytes	Descrição da bandeira
1 byte	Quantidade de Funções retornadas que indica a quantidade de repetições dos registros abaixo (29 bytes por função)
1 byte	Identifica a função: 'C': Crédito 'D': Débito; 'H': Cheque 'A': CDC 'S': Saque
14 bytes	Valor efetivado (2 casas decimais)
14 bytes	Valor cancelado (2 casas decimais)

Layout Versão 3:

Tamanho	Descrição
3 bytes	Identificação do buffer ("R03")
3 bytes	Quantidade de bandeiras retornadas que indica a quantidade de repetições dos registros abaixo
1 byte	Identifica a função: 'C': Crédito 'D': Débito; 'H': Cheque 'A': CDC
3 bytes	Código da bandeira
17 bytes	Descrição da bandeira
14 bytes	Valor efetivado (2 casas decimais)
14 bytes	Valor cancelado (2 casas decimais)

Protótipo

```
LONG ScopeObtemDadosTotalTEF (BYTE Nivel, char*Buffer, WORD TamBuffer)
```

Parâmetros

[in]	BYTE	Nível	Indica a que nível deverá ser inicializado os totais de TEF (0: PDV, 1: Operador)
[out]	char *	Buffer	Retorna os dados de totais TEF
[in]	WORD	TamBuffer	Tamanho da área que a aplicação alocou pela variável Buffer

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFB07	64263	Erro na totalização de TEF

Exemplo

```

...
WORD TamBuffer;
char Buffer[2048];
...
// encerrando o Dia
...
TamBuffer = sizeof(Buffer);
if(ScopeObtemDadosTotalTEF(0, Buffer, TamBuffer) == 0)
{
    ...
    GeraRelatorioEmArquivo(Buffer);
    ...
}
else
{
    // erro ao obter o cupom com o relatório
}
...

```

CUIDADO: Esta função somente aceita o *layout* Versão 1.

Protótipo

```
LONG ScopeObtemDadosTotalTEFEx (BYTE Versao, BYTE Nivel, char*Buffer, WORD TamBuffer)
```

Parâmetros

[in]	BYTE	Versao	Versão do <i>layout</i> a ser utilizado: 1, 2 ou 3
[in]	BYTE	Nivel	Indica a que nível deverá ser inicializado os totais de TEF (0: PDV, 1: Operador)
[out]	char *	Buffer	Retorna os dados de totais TEF
[in]	WORD	TamBuffer	Tamanho da área que a aplicação alocou pela variável Buffer

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFB07	64263	Erro na totalização de TEF

Exemplo

```

...
WORD TamBuffer;
char Buffer[2048];
...
// encerrando o Dia
...
TamBuffer = sizeof(Buffer);
if(ScopeObtemDadosTotalTEFEx(3, 0, Buffer, TamBuffer) == 0)
{
...
    GeraRelatorioEmArquivo(Buffer);
...
}
else
{
    // erro ao obter o cupom com o relatório
}
...

```

Transação de POS para conciliação

Aqui será tratada uma funcionalidade que guia a coleta de dados de um cupom de TEF.

Descrição da funcionalidade

Essa funcionalidade determina um fluxo de coleta que direcione o operador (a automação comercial) a entrar os dados do cupom de uma transação que foi realizada na máquina de POS. Este recurso foi criado para que a automação comercial obtivesse os dados do cupom para gravar no arquivo de conciliação do SCOPECON e que tais transações fossem conciliadas. Essa funcionalidade não se comunica com o SCOPE Server e a conexão pode estar offline. No entanto, é necessário que o SCOPE Client tenha feito o logon com o Server, isto é, que a função ScopeOpen tenha sido executado com sucesso.

A funcionalidade tinha como foco cartões que o SCOPE só passam em POS, como o CONSTRUCARD e o BNDES. No entanto, quaisquer cartões que passaram pelo POS podem usar esse recurso. Há situações em que a rede autorizadora ou até mesmo o próprio SCOPE Server está offline. Quando acontece isso, normalmente os estabelecimentos passam as transações em POS, que são utilizados como contingência do concentrador de TEF.

Em linhas gerais, o portador do cartão ao pagar sua compra na loja com um dos cartões, o funcionário da loja encaminha-o para o sistema que vai fazer o pagamento. Uma vez paga, o funcionário poderá registrar essa venda no PDV para a futura conciliação com a rede autorizadora.

Os dados da transação serão retornados para a automação comercial pela função ScopeObtemCampoExt2/ScopeObtemCampoExt3. Assim, esses dados poderão ser gravados em arquivo e exportados posteriormente para o módulo SCOPECon. Em outras palavras, os dados da transação para a conciliação não é feita pelo SCOPE. É de responsabilidade da automação comercial a geração do arquivo com os dados da transação. A participação do SCOPE nesse processo está em dois momentos:

1. A coleta dos dados do cupom pelo SCOPE Client para que a automação tenha os dados para o arquivo. Nessa situação o SCOPE serve apenas como um guia ao operador para solicitar as informações do cupom. No final da coleta, a automação solicitará ao SCOPE Client tais dados para o arquivo.
2. A conciliação da transação pelo SCOPECon. O arquivo gerado pela automação comercial será importado por esse módulo.

Transação POS

Essa função inicia um fluxo padrão de coleta do SCOPE para coletar as informações necessárias para que a automação comercial gere um registro do arquivo de conciliação que este seja futuramente importado pelo SCOPECon. No final desse fluxo a automação comercial poderá obter quase todos os campos necessário utilizando as funções ScopeObtemCampoExt2/ ScopeObtemCampoExt3.

Na loja, o operador de caixa irá com o portador do cartão a um POS passar o cartão para pagar a venda. O operador retornará ao caixa com o cupom do estabelecimento impresso pelo POS e executará essa transação. Os dados contidos na via do estabelecimento são importantes para alimentar os dados da transação. A não utilização do SCOPE poderá ser por diversas situações. A principal é quando o cartão é CONSTRUCARD ou BNDES, pois eles não são tratados pelo SCOPE. Outra situação é quando a rede autorizadora ou o próprio

SCOPE Server estiver offline. Nesse caso, a transação será paga no POS, mas o estabelecimento poderá fazer a conciliação da transação.

Os parâmetros da função não são obrigatórios. Passar valores diferentes do padrão simplificará o fluxo de coleta, não sendo solicitado ao operador. Se esta função vai ser chamada para finalizar a venda que está aberta, a automação poderá passar o valor da venda automaticamente.

IMPORTANTE: Essa transação não exige abertura de sessão. Também ela não tem garantia de queda de energia, ou seja, no caso de interrupção da transação, os dados coletados serão perdidos. Se aberta a sessão de TEF, ela não é considerada no limite de 9 TEFs permitidos na sessão.

Protótipo

```
LONG EXPORT ScopeTransacaoPOS(char *Valor,
WORD Rede,
WORD Bandeira,
WORD Servico)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300"). Passar string vazia para o SCOPE solicitar. Não pode ser nulo nem string apenas com zeros.
[in]	WORD	Rede	Número positivo que representa código da rede autorizadora segundo o SCOPE. Passar zero para o SCOPE solicitar.
[in]	WORD	Bandeira	Número positivo que representa código da bandeira segundo o SCOPE. Passar zero para o SCOPE solicitar.
[in]	WORD	Servico	Número positivo que representa código do serviço segundo o SCOPE. Passar zero para o SCOPE solicitar.

Retorno

Ver tabela de código de retorno no SCOPE.

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
0xFE00	65024	A transação em andamento – a aplicação deve aguardar

Exemplo

```
...
retorno = ScopeTransacaoPOS("", 0, 0, 0);
...
// processa a transação
...
```

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta. A maioria dos estados não necessitam tratamentos específicos pela automação. Basta que mostre a mensagem na tela do

operador e aguarde a digitação. Quando necessário, está descrito abaixo o tratamento específico que a automação poderá fazer.

ESTADO DE COLETA		DESCRIÇÃO	TRATAMENTO
HEX	DECIMAL		
0xFC87	64647	Exibe menu. Para essa transação, esse estado poderá ser utilizado para coletar dois dados diferentes. Um para a coleta da rede. Outro para a coleta da bandeira.	Obter a lista de opções do menu para a exibição de todas numa única vez. Ver tópico específico.
0xFC09	64521	Coletar se a transação é a vista ou não	Padrão
0xFC0A	64522	Coletar se a transação é parcelada pela administradora ou pelo estabelecimento	Padrão
0xFC0E	64526	Coleta a quantidade de parcelas	Padrão
0xFC34	64564	Coleta o valor da transação	Padrão
0xFCC7	64711	Coleta os 6 primeiros dígitos do cartão	Padrão
0xFC18	64536	Coleta os 4 últimos dígitos do cartão	Padrão
0xFC01	64513	Data de validade [MMAA]	Padrão
0xFC32	64562	Coleta a data da transação no formato DDMMAA	Padrão
0xFCC8	64712	Coleta o campo AUT (código de autorização)	Padrão
0xFC3A	64570	Coleta o campo CV (NSU do host)	Padrão
0xFCC9	64713	Coleta o campo DOC (NSU)	Padrão
0xFCE6	64742	Coleta o nome do portador do cartão	Padrão
0xFCFD	64765	Coleta em andamento	Aguarda a troca de estado. Não solicita dados ao operador.
0xFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE	Mostrar a mensagem para o operador e devolver o controle para o SCOPE. Não solicita dados ao operador.

Constantes em C dos estados de coleta

Esta tabela mostra as constantes definidas no arquivo de cabeçalho ScopeApi.h. Esse arquivo é disponibilizado junto com o SCOPE Client. Muito útil para automações escritas em linguagem C.

CONSTANTE	CÓDIGO EM HEXA
TC_VALIDADE_CARTAO	0xFC01
TC_DECIDE_AVISTA	0xFC09
TC_DECIDE_P_ADM_EST	0xFC0A
TC_QTDE_PARCELAS	0xFC0E
TC_ULTIMOS_DIGITOS	0xFC18
TC_COLETA_DDMMAA	0xFC32
TC_COLETA_VALOR	0xFC34
TC_COLETA_NSU_HOST	0xFC3A
TC_EXIBE_MENU	0xFC87
TC_PRIMEIROS_DIGITOS	0xFCC7
TC_COLETA_CAMPO_AUT	0xFCC8
TC_COLETA_CAMPO_DOC	0xFCC9
TC_COLETA_NOME_PORTADOR	0xFCE6
TC_COLETA_EM_ANDAMENTO	0xFCFD
TC_INFO_RET_FLUXO	0xFCFE

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Redes e bandeiras

O SCOPE Client tem um conjunto mínimo padrão implementado internamente para permitir a transação POS. São garantidas as redes CIELO e REDE-L0500, cujos respectivos códigos no SCOPE são 102 e 103. As bandeiras garantidas são VISA, MASTERCARD, ELECTRON, ELO DEBITO, ELO CREDITO, BNDES, CONSTRUCARD. Os serviços são: Débito à Vista, Crédito à Vista, Crédito Parc. Adm, Crédito Parc. Loja, Compra CDC e Crediário. A tabela abaixo mostra o relacionamento que o SCOPE Client mantém internamente.

REDE		BANDEIRA		SERVIÇO	
CÓD	NOME	CÓD	NOME	CÓD	NOME
102	CIELO	1	VISA	9	Crédito à Vista
102	CIELO	1	VISA	27	Crédito Parc. Adm
102	CIELO	1	VISA	28	Crédito Parc. Loja
102	CIELO	2	MASTERCARD	9	Crédito à Vista
102	CIELO	2	MASTERCARD	27	Crédito Parc. Adm
102	CIELO	2	MASTERCARD	28	Crédito Parc. Loja
102	CIELO	8	MAESTRO	6	Débito à Vista
102	CIELO	21	ELECTRON	6	Débito à Vista
102	CIELO	247	BNDES	6	Débito à Vista
102	CIELO	246	CONSTRUCARD	6	Débito à Vista
103	REDE-L0500	1	VISA	9	Crédito à Vista
103	REDE-L0500	1	VISA	27	Crédito Parc. Adm
103	REDE-L0500	1	VISA	28	Crédito Parc. Loja
103	REDE-L0500	2	MASTERCARD	9	Crédito à Vista
103	REDE-L0500	2	MASTERCARD	27	Crédito Parc. Adm
103	REDE-L0500	2	MASTERCARD	28	Crédito Parc. Loja
103	REDE-L0500	8	MAESTRO	6	Débito à Vista
103	REDE-L0500	8	MAESTRO	45	Compra CDC
103	REDE-L0500	21	ELECTRON	6	Débito à Vista
103	REDE-L0500	21	ELECTRON	45	Compra CDC
103	REDE-L0500	246	CONSTRUCARD	6	Débito à Vista
103	REDE-L0500	246	CONSTRUCARD	45	Compra CDC

Este relacionamento interno poderá ser desconsiderado se configurado no scope.ini um novo relacionamento. Ver tópico

Configuração do arquivo scope.ini.

Exemplos de configuração 1

Esse exemplo mostra uma configuração em que a transação teria um comportamento similar como à situação em que o arquivo não está configurado. O comportamento é similar, mas não igual, pois não temos a associação de serviço para cada configuração de rede ou bandeira. Logo, no fluxo as bandeiras não serão separadas conforme a opção escolhida de crédito, débito ou CDC e todas elas aparecerão para que o operador escolha uma.

```
[SCOPEAPIPOS]
DigitacaoCodigo=n
Complementar=n
Redes=102,103
Bandeiras=1,2,8,21,246,247
NomeBandeira001=VISA
NomeBandeira002=MASTERCARD
NomeBandeira008=MAESTRO
```

```
NomeBandeira021=ELECTRON  
NomeBandeira246=BNDES  
NomeBandeira247=CONSTRUCARD
```

```
[SCOPEAPIPOS-R102]  
Nome=CIELO  
Bandeiras=1,2,8,21,247,246
```

```
[SCOPEAPIPOS-R103]  
Nome= REDECARD  
Bandeiras=1,2,8,21, 246
```

Exemplos de configuração 2

Aqui temos uma configuração para um estabelecimento que tem apenas POS da CIELO como contingência. Não será perguntada pela rede REDE/REDECARD.

```
[SCOPEAPIPOS]  
DigitacaoCodigo=n  
Complementar=n  
Redes=102  
Bandeiras=1,2,8,21,246,247  
NomeBandeira001=VISA  
NomeBandeira002=MASTERCARD  
NomeBandeira008=MAESTRO  
NomeBandeira021=ELECTRON  
NomeBandeira246=BNDES  
NomeBandeira247=CONSTRUCARD
```

```
[SCOPEAPIPOS-R102]  
Nome=CIELO  
Bandeiras=1,2,8,21,247,246
```

Exemplos de configuração 3

A configuração desse exemplo agrupa à rede GETNET e as bandeiras SIMCRED e VISA para um estabelecimento que tem um POS da GETNET como contingência além do POS da CIELO e da REDE/REDECARD.

```
[SCOPEAPIPOS]  
DigitacaoCodigo=n  
Complementar=s  
Redes=73  
Bandeiras=136  
NomeBandeira136=SIMCRED
```

```
[SCOPEAPIPOS-R073]  
Nome=GETNET  
Bandeiras=136,1
```

Exemplos de configuração 4

Com essa configuração será dada a opção OUTRA e coletado o código de bandeira se essa opção for selecionada.

```
[SCOPEAPIPOS]  
DigitacaoCodigo=s
```

Exemplos de configuração 5

A configuração desse exemplo agrupa à rede GETNET e as bandeiras SIMCRED e VISA e ELECTRON para um estabelecimento que tem um POS da GETNET como contingência além do POS da CIELO e da REDE. Além disso, associa os serviços de crédito para SIMCRED e VISA e o serviço de débito para ELECTRON.

```
[SCOPEAPIPOS]
DigitacaoCodigo=n
Complementar=s
Redes=73
Bandeiras=136
NomeBandeira136=SIMCRED
```

```
[SCOPEAPIPOS-R073]
Nome=GETNET
Bandeiras=136,1,21
ServicoB001=9,027,28
ServicoB136=009,27,028
ServicoB021=6
```

Exemplos de configuração 6

Esse exemplo mostra uma configuração em que a transação teria um comportamento igual à situação em que o arquivo não está configurado. Diferente do exemplo 1, aqui o comportamento é igual, pois temos a associação de serviço para cada configuração de rede ou bandeira.

```
[SCOPEAPIPOS]
DigitacaoCodigo=n
Complementar=n
Redes=102,103
Bandeiras=1,2,8,21,246,247
NomeBandeira001=VISA
NomeBandeira002=MASTERCARD
NomeBandeira008=MAESTRO
NomeBandeira021=ELECTRON
NomeBandeira246=BNDES
NomeBandeira247=CONSTRUCARD
```

```
[SCOPEAPIPOS-R102]
Nome=CIELO
Bandeiras=1,2,8,21,247,246
ServicoB001=009,27,28
ServicoB002=9,27,28
ServicoB008=6,134
ServicoB021=6,134
ServicoB247=6,134
ServicoB246=6,134
```

```
[SCOPEAPIPOS-R103]
Nome=REDECARD
Bandeiras=1,2,8,21, 246
ServicoB001=9,27,028
ServicoB002=9,027,28
ServicoB008=6,45
ServicoB021=6,45
ServicoB246=6,45
```

Códigos de serviços

No SCOPE existem vários códigos de serviços disponíveis, mas nem todos são aplicáveis nesta funcionalidade. Cada serviço é agrupado em um grupo de serviço. Abaixo está a relação de grupos de serviços e os serviços previstos como é no SCOPE.

Grupo de serviço		Serviço	
Código	Descrição	Código	Descrição
01	Cartão de Débito	6	Débito à Vista
		45	Compra CDC
02	Cartão de Crédito	9	Crédito à Vista
		27	Crédito Parc. Adm
		28	Crédito Parc. Loja

Percebe-se que o CDC está identificado como débito, mas para fins da transação POS, os serviços estarão agrupados conforme abaixo:

Grupo de serviço		Serviço	
Descrição	Código	Descrição	Código
Cartão de Débito	6	Débito à Vista	
	45	Compra CDC	
CDC	45	Compra CDC	
	9	Crédito à Vista	
Cartão de Crédito	27	Crédito Parc. Adm	
	28	Crédito Parc. Loja	

MENSAGEM TBL 9X
VISA ELECTRON-I
Visanet
444054-2871-07/15
2a VIA - ESTABELECIMENTO AUT=095023
0010000244470001/POS=25000002
DOC=096044 04/12/12 09:46 ONL-D
VENDA A DEBITO EM 02 PARCELAS
VALOR: 10,00
1a PARC: 04/12/12
MENSAGEM TBL 7C
RECONHECO COMO LIQUIDA E CERTA A DIA-
VIDA POR MIM ASSUMIDA EM DECORRENTE
DO CONTRATO CELEBRADO COM A INSTITUI-
CAO FINANCEIRA EMISSORA DO MEU CARTAO
TRANSACAO AUTORIZADA COM SENHA


Figura 6: cupom da CIELO

Cupom CIELO X ScopeObtemCampoExt2 X arquivo

Nesse tópico são mapeados os campos de cupons da rede CIELO com as constantes da função ScopeObtemCampoExt2 do SCOPE Client e os campos do registro do arquivo de conciliação. Na transação POS o cupom utilizado deverá ser a via do estabelecimento.

Seguindo o exemplo do cupom de uma transação da CIELO da **Figura 6**, os campos serão mapeados da forma conforme abaixo.

ARQUIVO CONCILIAÇÃO					FUNÇÃO ScopeObtemCampoExt2			CUPOM	
Descrição	POS	TAM	TIPO	COMENTÁRIOS	MÁSCARA1	MÁSCARA2	MÁSCARA3	Campo	

Tipo de registro	1	1	N	Constante '1'	NA*	NA*	NA*	NA*
Numero do PV	2	15	N	Código do Estabelecimento	0x00000800	NA*	NA*	NA*
Data da venda	17	8	D	Data da venda (YYYYMMDD)	NA*	0x00000008	NA*	Data
NSU	25	6	A	Num. sequencial único	0x00000004	NA*	NA*	DOC
NSU do Host	31	15	A	NSU complementar	0x00004000	NA*	NA*	NA*
Código de autorização	41	14	A	Código de autorização	0x00000100	NA*	NA*	AUT
Valor da compra	51	15(2)	N	Valor da operação	0x00000002	NA*	NA*	VALOR
Número do cartão	66	22	A	Número do cartão	0x00000001	NA*	NA*	6 primeiros e 4 últimos dígitos do cartão
Status da transação	85	1	A	Transações OK, preencher com a letra "O"	NA*	NA*	NA*	NA*
Código Empresa	86	4	N	Formato "0000"	NA*	0x08000000	NA*	NA*
Código Filial	90	4	N	Formato "0000"	NA*	NA*	NA*	NA*
Qtd Parcelas	94	2	N	Quantidade de parcelas. Se venda for à vista, preencher '00'	0x00001000	NA*	NA*	Número de parcelas
Forma Captura	96	1	A	"P"-POS, "I"- Internet, "M"- Manual, "T"-TEF ou "O" Outros.	NA*	NA*	0x00000080	NA*
Código Bandeira	97	3	N	Código da Bandeira	0x00040000	NA*	NA*	Bandeira
Código Rede	100	3	N	Código da Rede	0x00400000	NA*	NA*	Rede
Código Serviço	103	5	N	Código do Serviço	0x00080000	NA*	NA*	Venda
RESERVADO	108	89	A	RESERVADO	NA*	NA*	NA*	NA*

* NA - não se aplica

Para ilustrar o mapeamento, considera-se o código da empresa e o código da filial do PDV sendo "0001" e "0001", respectivamente, e o cupom de exemplo. Após a transação POS do SCOPE Client e a formatação da automação comercial, o cupom deverá resultar no registro com exibido na próxima tabela.

CUPOM CAMPO	FUNÇÃO ScopeObtemCampoExt2			ARQUIVO REGISTRO	
	MÁSCARA CAMPO	MÁSCARA 1	MÁSCARA 2	MÁSCARA 3	RETORNO
-	-	-	-	-	"1"

-	0x00000800	0x00000000	0x00000000	" "	" "
041212	0x00000000	0x00000008	0x00000000	"041212"	"20121204"
096044	0x00000004	0x00000000	0x00000000	"096044"	"096044"
-	0x00004000	0x00000000	0x00000000	" "	" "
095023	0x000000100	0x00000000	0x00000000	"095023"	"095023 "
1000	0x00000002	0x00000000	0x00000000	"1000"	"0000000000001000"
444054-2871	0x00000001	0x00000000	0x00000000	"444054-2871"	"444054-2871 "
-	-	-	-	-	"O"
-	0x0000000000	0x08000000	0x0000000000	"0001"	"0001"
-	-	-	-	-	"0001"
2	0x00001000	0x00000000	0x00000000	"2"	"02"
-	0x0000000000	0x00000000	0x0000000080	"P"	"P"
ELECTRON	0x00040000	0x00000000	0x00000000	"21"	"021"
CIELO	0x00400000	0x00000000	0x00000000	"102"	"102"
Débito parcelado	0x00080000	0x00000000	0x00000000	"134"	"134"
-	-	-	-	-	" ... "

Cupom REDE X ScopeObtemCampoExt2 X arquivo

Como no tópico anterior, nesse são mapeados os campos de cupons da rede REDE com as constantes da função ScopeObtemCampoExt2 do SCOPE Client e os campos do registro do arquivo de conciliação. Na transação POS o cupom utilizado deverá ser a via do estabelecimento.

REDECARD

**Figura 7:** cupom da rede REDE

Seguindo o exemplo do cupom de uma transação da REDE da Figura 7, os campos serão mapeados da forma conforme tabela.

ARQUIVO CONCILIAÇÃO				FUNÇÃO ScopeObtemCampoExt2			CUPOM	
Descrição	Pos	Tam	Tipo	Comentários	Máscara 1	Máscara 2	Máscara 3	Campo
Tipo de registro	1	1	N	Constante '1'	NA*	NA*	NA*	NA*
Numero do PV	2	15	N	Código do Estabelecimento	0x00000800	NA*	NA*	NA*
Data da venda	17	8	D	Data da venda (YYYYMMDD)	NA*	0x00000008	NA*	Data
NSU	25	6	A	Num. sequencial único	0x00000004	NA*	NA*	NA*
NSU do Host	31	15	A	NSU complementar	0x00004000	NA*	NA*	CV
Código de autorização	41	14	A	Código de autorização	0x00000100	NA*	NA*	AUTO
Valor da compra	51	15(2)	N	Valor da operação	0x00000002	NA*	NA*	TOT. APROVADO
Número do cartão	66	22	A	Número do cartão	0x00000001	NA*	NA*	6 primeiros opcionais e 4 últimos obrigatórios dígitos do cartão
Status da transação	85	1	A	Transações OK, preencher com a letra "O"	NA*	NA*	NA*	NA*
Código Empresa	86	4	N	Formato "0000"	NA*	0x08000000	NA*	NA*

Código Filial	90	4	N	Formato "0000"	NA*	NA*	NA*	NA*
Qtd Parcelas	94	2	N	Quantidade de parcelas. Se venda for à vista, preencher '00'	0x00001000	NA*	NA*	Número de parcelas
Forma Captura	96	1	A	"P"-POS, "I"-Internet, "M"-Manual, "T"-TEF ou "O" Outros.	NA*	NA*	0x00000080	NA*
Código Bandeira	97	3	N	Código da Bandeira	0x00040000	NA*	NA*	Bandeira
Código Rede	100	3	N	Código da Rede	0x00400000	NA*	NA*	Rede
Código Serviço	103	5	N	Código do Serviço	0x00080000	NA*	NA*	Venda
RESERVADO	108	89	A	RESERVADO	NA*	NA*	NA*	NA*

* NA – não se aplica

Para ilustrar o mapeamento, considera-se o código da empresa e o código da filial do PDV sendo "0001" e "0001", respectivamente, e o cupom de exemplo. Após a transação POS do SCOPE Client e a formatação da automação comercial, o cupom deverá resultar no registro com exibido nessa tabela.

CUPOM	FUNÇÃO ScopeObtemCampoExt2					ARQUIVO
	CAMPO	MÁSCARA 1	MÁSCARA 2	MÁSCARA 3	RETORNO	
-	-	-	-	-	"1"	
-	0x00000800	0x00000000	0x00000000	0x00000000	"00000000000000000"	"00000000000000000"
121212	0x0000000000	0x0000000008	0x0000000000	0x0000000000	"121212"	"20121212"
-	0x0000000004	0x0000000000	0x0000000000	0x0000000000	"096044"	"096044"
499931944	0x00004000	0x0000000000	0x0000000000	0x0000000000	"499931944"	"499931944 "
018833	0x000000100	0x0000000000	0x0000000000	0x0000000000	"018833"	"018833 "
1330	0x000000020	0x0000000000	0x0000000000	0x0000000000	"1330"	"0000000000001330"
8096	0x0000000101	0x0000000000	0x0000000000	0x0000000000	"000000-8096"	"000000-8096 "
-	-	-	-	-	"O"	
-	0x0000000000	0x0800000000	0x0000000000	0x0000000000	"0001"	"0001"
-	-	-	-	-	"0001"	
-	0x000001000	0x0000000000	0x0000000000	0x0000000000	"1"	"01"
-	0x0000000000	0x0000000000	0x0000000080	0x0000000080	"P"	"P"
MAESTRO	0x000400000	0x0000000000	0x0000000000	0x0000000000	"8"	"008"
REDECARD	0x004000000	0x0000000000	0x0000000000	0x0000000000	"103"	"103"

Débito à vista	0x000800 00	0x000000 00	0x000000 00	"6"	"006"
-	-	-	-	-	"0000...000"

Outros cupons X ScopeObtemCampoExt2 X arquivo

Pela configuração do scope.ini, o SCOPE Client poderá coletar dados de cupom de outras redes. No entanto, os nomes dos campos coletados serão genéricos.

ARQUIVO CONCILIAÇÃO					FUNÇÃO ScopeObtemCampoExt2			CUPOM	
Descrição	Pos	Tam	Tipo	Comentários	Máscara A1	Máscara A2	Máscara A3	Campo	Obrig.
Tipo de registro	1	1	N	Constante '1'	NA*	NA*	NA*	NA*	NA*
Numero do PV	2	15	N	Código do Estabelecimento	0x00000 800	NA*	NA*	NA*	NA*
Data da venda	17	8	D	Data da venda (YYYYMMDD)	NA*	0x00000 008	NA*	Data	SIM
NSU	25	6	A	Num. sequencial único	0x00000 004	NA*	NA*	NA*	NÃO
NSU do Host	31	15	A	NSU complementar	0x00004 000	NA*	NA*	CV	NÃO
Código de autorização	41	14	A	Código de autorização	0x000001 00	NA*	NA*	AUTO	NÃO
Valor da compra	51	15(2)	N	Valor da operação	0x00000 002	NA*	NA*	TOT. APROVADO	SIM
Número do cartão	66	22	A	Número do cartão	0x00000 001	NA*	NA*	6 primeiros dígitos do cartão	NÃO
								4 últimos dígitos do cartão	SIM
Status da transação	85	1	A	Transações OK, preencher com a letra "O"	NA*	NA*	NA*	NA*	NA*
Código Empresa	86	4	N	Formato "0000"	NA*	0x08000 000	NA*	NA*	NA*
Código Filial	90	4	N	Formato "0000"	NA*	NA*	NA*	NA*	NA*
Qtd Parcelas	94	2	N	Quantidade de parcelas. Se venda for à vista, preencher '00'	0x000010 00	NA*	NA*	Número de parcelas	SIM
Forma Captura	96	1	A	"P"-POS, "I"-Internet, "M"-Manual, "T"-TEF ou "O" Outros.	NA*	NA*	0x00000 080	NA*	NA*

Código Bandeira	97	3	N	Código da Bandeira	0x00040000	NA*	NA*	Bandeira	SIM
Código Rede	100	3	N	Código da Rede	0x00400000	NA*	NA*	Rede	SIM
Código Serviço	103	5	N	Código do Serviço	0x00080000	NA*	NA*	Venda	SIM
RESERVADO	108	89	A	RESERVADO	NA*	NA*	NA*	NA*	NA*

Formatação para o arquivo de conciliação

Observa-se nos exemplos a automação comercial precisará tratar alguns campos, formatando-os para gravar no arquivo de conciliação. Será necessária a formatação nos campos:

- Tipo de registro: preencherá sempre com o dígito '1'.
- Data da transação: o SCOPE Cliente fornece a data no formato DDMMAA e a automação alterará para o formato AAAAMMDD.
- NSU do host: completará com espaços à direita.
- Código de autorização: completará com espaços à direita.
- Valor da transação: completará com zeros à esquerda.
- Número do cartão: completará com espaços à direita.
- Status da transação: preencherá sempre com a letra "O".
- Código da filial: preencherá com o código da filial usado na função ScopeOpen.
- Número de parcelas: completará com zeros à esquerda.
- Código da rede: completará com zeros à esquerda.
- Código da bandeira: completará com zeros à esquerda.
- Reservado: preencherá sempre com espaços à direita.

Funções de Pagamento Recorrente

Este capítulo descreve as funções relacionadas ao Pagamento Recorrente as quais a aplicação de Automação Comercial tem acesso.

Cadastro de Pagamento Recorrente

A função ScopeCadastroPagamentoRecorrente() aciona o SCOPE Client para efetuar uma transação de cadastro de pagamento recorrente na rede VINDI.

Protótipo em C/C++

```
LONG EXPORT ScopeCadastroPagamentoRecorrente (WORD CodBandeira, char *Dados, WORD TamDados)
```

Parâmetros

[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.
[in]	char	Dados	Dados do pagamento recorrente.
[in]	WORD	TamDados	Tamanho da área de dados.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Hexadecim al	Decimal	Significado
0xFA01	64001	Parâmetro 1 inválido	
0xFA02	64002	Parâmetro 2 inválido	
0xFA03	64003	Parâmetro 3 inválido	
0xFE00	65024	A transação em andamento – a aplicação deve aguardar	
0xFE01	65025	SCOPE API não foi inicializada corretamente	
0xFFOC	65292	Transação não implementada	
0xFE06	65030	Logon duplicado	
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server	
0xFE08	65032	POS não cadastrado	
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis	
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).	
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida	
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada	
0xFF0A	65290	Banco de dados off-line	
0xFF5E	65374	Erro ao desmontar a estrutura ISO	
0xFF7D	65405	Chamada da função não permitida	
0xFFFF	65535	Erro genérico	

Exemplo em C/C++

| ...

```

// abre sessão
...
retorno = ScopeCadastraPagamentoRecorrente( (WORD) CódigoBandeira, Dados, Tamanho);
...
// processa a transação
...
// fecha a sessão
...

```

Estruturas de apoio

Aplicações escritas em linguagem C podem usufruir de estruturas definidas no arquivo de cabeçalho *ScopeApi.h*. O buffer contendo os dados de cadastro do pagamento recorrente resume-se na estrutura abaixo:

```

typedef struct
{
    char      Formato;
    stPAGREC_CARD   Card;
    stPAGREC_CUSTOMER Customer;
    stPAGREC_SUBSCRIPTION Subscription;
}stPAGREC_F1, *ptPAGREC_F1;

```

onde:

- Formato: fixo em '1' para viabilizar eventuais evoluções futuras;
- Card: contém os dados do cartão;
- Customer: contém os dados do usuário;
- Subscription: contém os dados da assinatura.

O membro *Card* utiliza a estrutura:

```

typedef struct
{
    char ModoOper;
    char Token[2];
    char PaymentProfileId[36];
    char PortadorCPFCNPJ[14];
    char NomeBandeira[40];
    char GrupoServiço[2];
} tPAGREC_CARD, *ptPAGREC_CARD;

```

onde:

- ModoOper: Modo de operação. Pode assumir os seguintes valores:
 - '0'=Coletar – indica que haverá coleta de cartão no fluxo
 - '1'=Token – indica que é informado o identificador da transação
 - '2'=PaymentProfileId – indica que é informado o identificador do perfil de pagamento.
- Token: identificador da transação; mandatório, se ModoOper igual a '1';
- PaymentProfileId: identificador do perfil de pagamento; mandatório, se ModoOper igual a '2';
- PortadorCPFCNPJ: CPF ou CNPJ do portador do cartão;
- NomeBandeira: nome da bandeira;
- GrupoServiço: grupo de serviço ao qual pertence a transação; mandatório se ModoOper igual a '0'. Pode assumir os valores "01"=Débito, "02"=Crédito.

O membro *Customer* utiliza a estrutura:

```

typedef struct
{
    char Id[36];
}

```

```

char Nome[40];
char CPFCNPJ[14];
char Email[99];
char TelPais[2];
char TelArea[2];
char TelNum[9];
char TelExt[4];
char TelTipo;
char APICode[15];
} stPAGREC_CUSTOMER, *ptPAGREC_CUSTOMER;

```

onde:

- Id: se fornecido, não cadasra cliente; caso contrário, fornecer os campos abaixo;
- Nome: nome do cliente;
- CPFCNPJ: CPF ou CNPJ do cliente;
- Email: e-mail do cliente;
- TelPais: código do país (55=Brasil);
- TelArea: código de área;
- TelNum: número do telefone;
- TelExt: ramal;
- TelTipo: tipo do telefone: '0'=Desconhecido, '1'=Fixo, '2'=Celular;
- APICode: código do cliente para referência via API.

O membro *Subscription* utiliza a estrutura:

```

typedef struct
{
    char PlanId[36];
    char APICode[15];
} stPAGREC_SUBSCRIPTION, *ptPAGREC_SUBSCRIPTION;

```

onde:

- PlanId: Identificador do plano para assinatura;
- APICode: código da assinatura para referência via API.

Protótipo em Java

```
public int cadastroPagamentoRecorrente(int codBandeira, PagamentoRecorrente dadosPagamento)
```

Parâmetros

[in] int	codBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.
[in] PagamentoRecorrente	dadosPagamento	Dados do pagamento recorrente.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido

OxFA02	64002	Parâmetro 2 inválido
OxFA03	64003	Parâmetro 3 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFFOC	65292	Transação não implementada
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFF7D	65405	Chamada da função não permitida
OxFFFF	65535	Erro genérico

Exemplo em Java

```

...
// abre sessão
...
Scope scope = new Scope();
retorno = scope.cadastraPagamentoRecorrente(codBandeira, dadosPagamento);
...
// processa a transação
...
// fecha a sessão
...

```

Classes de apoio

Aplicações escritas em linguagem Java podem usufruir de classes definidas para esta estrutura de negócio. O conteúdo dos dados de cadastro do pagamento recorrente resume-se nas classes abaixo:

PagamentoRecorrente
- cartao : PRCartao - cliente : PRCliente - assinatura : PRAssinatura

onde:

- cartao: contém os dados do cartão;
- cliente: contém os dados do usuário;
- assinatura: contém os dados da assinatura;

O membro *Card* utiliza a estrutura:

PRCartao
- modoOperacao : int
- token : String
- paymentProfileId : String
- cpfCnpjPortador : String
- nomeBandeira : String
- grupoServico : String

onde:

- modoOperacao: Modo de operação. Pode assumir os seguintes valores:
0 = Coletar – indica que haverá coleta de cartão no fluxo
1 = Token – indica que é informado o identificador da transação
2 = PaymentProfileId – indica que é informado o identificador do perfil de pagamento.
- token: identificador da transação; mandatório, se ModoOper igual a '1';
- paymentProfileId: identificador do perfil de pagamento; mandatório, se modoOperacao igual a '2';
- cpfCnpjPortador: CPF ou CNPJ do portador do cartão;
- nomeBandeira: nome da bandeira;
- grupoServico: grupo de serviço ao qual pertence a transação; mandatório se ModoOper igual a '0'. Pode assumir os valores "01"=Débito, "02"=Crédito.

O membro *Customer* utiliza a estrutura:

PRCliente
- id : String
- nome : String
- cpfCnpj : String
- email : String
- telCodPais : String
- telArea : String
- telNumero : String
- telRamal : String
- telTipo : String
- codCliente : String

onde:

- id: se fornecido, não cadastra cliente; caso contrário, fornecer os campos abaixo;
- nome: nome do cliente;
- cpfCnpj: CPF ou CNPJ do cliente;
- email: e-mail do cliente;
- telCodPais: código do país (55=Brasil);
- telArea: código de área;
- telNumero: número do telefone;
- telRamal: ramal;
- telTipo: tipo do telefone: '0'=Desconhecido, '1'=Fixo, '2'=Celular;
- codClient: código do cliente para referência via API.

O membro *Subscription* utiliza a estrutura:

PRAssinatura
- id : String - codAssinatura : String

onde:

- id: Identificador do plano para assinatura;
- codAssinatura: código da assinatura para referência via API.

Estados de coleta

A tabela abaixo mostra os principais estados de coleta que poderão retornar no modo coleta:

Códigos Retorno			
Estado	Hexa	Decim al	Significado
TC_COLETA_CARTAO_EM_ANDAMENTO	0xFCFC	64764	<p>Coleta de cartão no pinpad está em andamento.</p> <p>Será retornado nas seguintes situações:</p> <ul style="list-style-type: none"> ○ A automação não informou um token referente aos dados do cartão de uma transação aprovada anteriormente na mesma sessão de TEF; ○ O parâmetro CFG_CANCELAR_OPERACAO_PINPAD está habilitado; ○ Há um pinpad compartilhado ou ABECS instalado e funcionando.
TC_OBTEM_SERVICOS	0xFC84	64644	<p>Serviços disponíveis para a transação.</p> <p>Será retornado somente se o parâmetro CFG_OBTER_SERVICOS estiver habilitado.</p>
TC_CARTAO_DIGITADO	0xFC85	64645	<p>Número do cartão (PAN) no modo digitado.</p> <p>Será retornado nas seguintes situações:</p> <ul style="list-style-type: none"> ○ A automação não informou um token referente aos dados de uma transação aprovada anteriormente na mesma sessão TEF; ○ Não há pinpad instalado ou a coleta do cartão no pinpad foi cancelada.
TC_COLETA_EM_ANDAMENTO	0xFCFD	64765	<p>Coleta de informação no pinpad, exceto coleta de cartão.</p> <p>Será retornado nas seguintes situações:</p> <ul style="list-style-type: none"> ○ A automação não informou um token referente aos dados de uma transação aprovada anteriormente na mesma sessão TEF;

			<ul style="list-style-type: none"> ○ O pinpad é ABECS: a coleta do código de segurança deverá ser no dispositivo.
TC_CVV_CVC_2	0xFC29	64553	<p>Coleta do código de segurança. Será retornado nas seguintes situações:</p> <ul style="list-style-type: none"> ○ A automação não informou um token referente aos dados de uma transação aprovada anteriormente na mesma sessão TEF; ○ O pinpad está executando a biblioteca compartilhada.
TC_VALIDADE_CARTAO	0xFC01	64513	<p>Data de vencimento do cartão. Será retornado na seguinte situação:</p> <ul style="list-style-type: none"> ○ A automação não informou um token referente aos dados de uma transação aprovada anteriormente na mesma sessão TEF;
TC_IMPRIME_CUPOM	0xFC02	64514	Impressão do cupom. Será retornado quando todas as mensagens da transação enviadas para a VINDI forem aprovadas.

Inclusão de Cartão

A função ScopelIncluiCartaoPagamentoRecorrente() aciona o SCOPE Client para efetuar uma transação de inclusão de um cartão para pagamento recorrente.

Protótipo em C/C++

```
LONG EXPORT ScopelIncluiCartaoPagamentoRecorrente (WORD CodBandeira, char *Dados, WORD TamDados)
```

Parâmetros

[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.
[in]	char	Dados	Dados do cartão para o pagamento recorrente.
[in]	WORD	TamDados	Tamanho da área de dados.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		
Hexadecim	Decimal	Significado
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF0C	65292	Transação não implementada
0xFE06	65030	Logon duplicado

OxFEO7	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFF7D	65405	Chamada da função não permitida
OxFFFF	65535	Erro genérico

Exemplo em C/C++

```

...
// abre sessão
...
retorno = ScopeIncluiCartaoPagamentoRecorrente( (WORD) CódigoBandeira, Dados, Tamanho);
...
// processa a transação
...
// fecha a sessão
...

```

Estruturas de apoio

Aplicações escritas em linguagem C podem usufruir de estruturas definidas no arquivo de cabeçalho ScopeApi.h. O buffer contendo os dados do cartão para pagamento recorrente resume-se na estrutura abaixo:

```

typedef struct
{
    char      Formato;
    stPAGREC_CARD      Card;
    stPAGREC_CUSTOMER  Customer;
    stPAGREC_SUBSCRIPTION Subscription;
}stPAGREC_F1, *ptPAGREC_F1;

```

onde:

- Formato: fixo em '1' para viabilizar eventuais evoluções futuras;
- Card: contém os dados do cartão;
- Customer: não é utilizado por essa função (preencher com brancos);
- Subscription: não é utilizado por essa função (preencher com brancos).

O membro *Card* utiliza a estrutura:

```

typedef struct
{
    char ModoOper;
    char Token[2];
    char PaymentProfileId[36];
    char PortadorCPFCNPJ[14];
    char NomeBandeira[40];
    char GrupoServiço[2];
}tPAGREC_CARD, *ptPAGREC_CARD;

```

onde:

- ModoOper: Modo de operação. Pode assumir os seguintes valores:
 - 0=Coletar :indica que haverá coleta de cartão no fluxo
 - 1=Token: indica que é informado o identificador da transação
 - 2=PaymentProfileId: não é permitido para essa função.
- Token: identificador da transação; mandatório, se ModoOper igual a '1';
- PaymentProfileId: preencher com brancos;
- PortadorCPFCNPJ: CPF ou CNPJ do portador do cartão;
- NomeBandeira: nome da bandeira;
- GrupoServiço: grupo de serviço ao qual pertence a transação; mandatório, se ModoOper igual a '0'. Pode assumir os valores "01"=Débito, "02"=Crédito.

IMPORTANTE: Apesar dessa função não utilizar as informações dos membros Customer e Subscription, tenha certeza de que o tamanho do buffer passado seja no mínimo o tamanho total da estrutura `stPAGREC_F1` a fim de evitar problemas de invasão de memória.

Protótipo em Java

```
public int incluiCartaoPagamentoRecorrente (int codBandeira, PagamentoRecorrente pagamentoRecorrente)
```

Parâmetros

[in] int	codBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.
[in] PagamentoRecorrente	pagamentoRecorrente	Dados do cartão para o pagamento recorrente.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF0C	65292	Transação não implementada
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida

OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFF7D	65405	Chamada da função não permitida
OxFFFF	65535	Erro genérico

Exemplo em Java

```

...
// abre sessão
...
Scope scope = new Scope();
retorno = scope.incluiCartaoPagamentoRecorrente(codBandeira, pagamentoRecorrente);
...
// processa a transação
...
// fecha a sessão
...

```

Classes de apoio

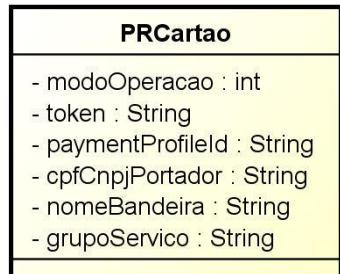
Aplicações escritas em linguagem Java podem usufruir de classes definidas para esta estrutura de negócio. O conteúdo dos dados de cadastro do pagamento recorrente resume-se nas classes abaixo:



onde:

- cartao: contém os dados do cartão;
- cliente: não é utilizado por essa função (preencher com brancos);
- assinatura: não é utilizado por essa função (preencher com brancos).

O membro *Card* utiliza a estrutura:



onde:

- modoOperacao: Modo de operação. Pode assumir os seguintes valores:

- 0 = Coletar : indica que haverá coleta de cartão no fluxo;
- 1 = Token: indica que é informado o identificador da transação;
- 2 = PaymentProfileId: não é permitido para essa função.
- token: identificador da transação; mandatório, se ModoOper igual a '1';
- paymentProfileId: preencher com brancos;
- cpfCnpjPortador: CPF ou CNPJ do portador do cartão;
- nomeBandeira: nome da bandeira;
- grupoServico: grupo de serviço ao qual pertence a transação; mandatório, se modoOperacao igual a '0'. Pode assumir os valores:
 - 01 = Débito
 - 02 = Credito

Estados de coleta

A tabela [acima](#) mostra os principais estados de coleta que poderão retornar no modo coleta.

Identificação da Transação

A função ScopeObtemTransacaold() permite ao SCOPE Client obter uma identificação de uma transação (ou *token*) de pagamento recorrente. Ela poderá ser chamada durante o fluxo de crédito ou débito na mesma sessão de TEF.

Protótipo em C/C++

```
LONG EXPORT ScopeObtemTransacaold (char *Transacaold, WORD TamTransacaold)
```

Parâmetros

[out]	string	Trancacaold	Identificador da transação na sessão corrente composto por dois dígitos ASCII e o caracter terminador de string.
[in]	WORD	TamTransacaold	Tamanho da área indicada por Transacaold que deve ser pelo menos 3 bytes.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno Hexadecim al	Decimal	Significado
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF0C	65292	Transação não implementada
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado

OxFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF41	65345	Chamada da função em momento incorreto
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFF7D	65405	Chamada da função não permitida
OxFFFF	65535	Erro genérico

Exemplo em C/C++

```

...
// abre sessão
...
ScopeGetParam((WORD) RC, &PColeta);

// Tratamento dos estados de coleta
switch(RC)
{
...
case TC_IMPRIME_CUPOM:
// Realiza tratamento do cupom
    Retorno = ScopeObtemTransacaold( (char *) Transacaold,
                                    TamTransacaold);
...
}

// fecha a sessão
...

```

IMPORTANTE: A automação poderá chamar essa função apenas após a aprovação da TEF pela VINDI. Consequentemente, a chamada deverá ser realizada a partir do estado de coleta **TC_IMPRIME_CUPOM**.

IMPORTANTE: A chamada da função em momento inoportuno retornará o código de erro **RCS_ERRO_DADOS_INDISPONIVEIS** (FF41_h).

Protótipo em Java

```
public int obtemTransacaold (SessaoTransacao transacao)
```

Parâmetros

[in]	SessaoTransacao	transacao	Identificador da transação na sessão corrente composto por dois dígitos
------	-----------------	-----------	---

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFA02	64002	Parâmetro 2 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFFOC	65292	Transação não implementada
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF41	65345	Chamada da função em momento incorreto
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFF7D	65405	Chamada da função não permitida
OxFFFF	65535	Erro genérico

Exemplo em Java

```

...
// abre sessão
...
Scope scope = new Scope();
scope.getParam(tipoparam, parametros);

// Tratamento dos estados de coleta
switch(rc)
{
    ...
    case TC_IMPRIME_CUPOM:
        // Realiza tratamento do cupom
        ...
        retorno = scope.obtemTransacaold(transacao);
        ...
    }
}

// fecha a sessão
...

```

Classes de apoio

Aplicações escritas em linguagem Java podem usufruir de classes definidas para esta estrutura de negócio. O conteúdo dos dados de cadastro do pagamento recorrente resume-se nas classes abaixo:

SessaoTransacao
- id : String

onde:

- id: Identificador da transação ou token do pagamento recorrente.

PIN Impresso

Este capítulo descreve as funções relacionadas à operação de PIN Impresso (*PIN Printing*) as quais a aplicação de Automação Comercial tem acesso.

Um **Cartão Digital** é um produto eletrônico que permite ao consumidor acessar um serviço ou realizar pagamentos utilizando um código exclusivo, geralmente chamado de PIN (*Personal Identification Number*). No contexto do **PIN Printing**, o **Cartão Digital** é gerado e entregue ao cliente em formato eletrônico ou físico (como um comprovante impresso). Ele pode ser utilizado para ativar créditos pré-pagos, como assinaturas de serviços, acesso a plataformas de conteúdo digital ou recargas de contas.

- **Exemplo de uso:** Um consumidor compra um **Cartão Digital** em um ponto de venda, recebe um comprovante impresso com o PIN gerado pelo sistema, e insere esse código no site ou aplicativo correspondente para ativar o serviço.

Consulta Conteúdo Digital

A função `ScopeConsultaConteudoDigital()` aciona o SCOPE Client para consultar os provedores e produtos de conteúdo digital disponíveis, tendo como objetivo posterior a aquisição de um Cartão Digital.

Protótipo em C/C++

```
LONG EXPORT ScopeConsultaConteudoDigital (WORD _Servico, WORD _CodBandeira)
```

Parâmetros

[in]	WORD	_Servico	Para efetuar a operação de Consulta Conteúdo Digital usar o serviço 192 S_SOLICITACAO_PIN_IMPRESSO
			O Código da Bandeira indicado poderá ser B_EPAY [221], ou outra que venha a suportar esta operação no futuro. Se o Código da Bandeira for informado como zero `0`, o SCOPE verificará quais redes estão habilitadas para o serviço. - Caso apenas uma rede esteja habilitada, ela será selecionada automaticamente, juntamente com a bandeira associada. - Se mais de uma rede estiver habilitada, o SCOPE disponibilizará uma lista das redes disponíveis através da função <code>ScopeObtemListaRedesConteudoDigital()</code> , permitindo que o sistema de automação escolha qual utilizar para prosseguir com a operação.

Retorno

Ver [tabela de código de retorno](#).

O objetivo desta função é chegar ao Código do Produto Digital e Valor desejado para a solicitação de um PIN de ativação de créditos a ser impresso pelo sistema de frente de caixa.

O sistema de frente de caixa, após a conclusão do devido pagamento do valor desejado, deverá acionar a função ScopeEfetivaConteudoDigital() para gerar e imprimir um cupom contendo o PIN necessário para uso posterior do produto digital adquirido.

NOTA: O Código do Produto Digital, Valor e Código da Bandeira relacionada devem servir como parâmetros de entrada para a função ScopeEfetivaConteudoDigital().

Exemplo

```

...
unsigned short servico = S_SOLICITACAO_PIN_IMPRESSO; // valor 192 (servico.h)
unsigned short bandeira = B_EPAY; // valor 221 (bandeira.h)
...
// abre sessão
...
retorno = ScopeConsultaConteudoDigital (servico, bandeira);
...
// processa a transação
...
// obtem e guarda dados para uso posterior em ScopeEfetivaConteudoDigital()
char szValor[TAM_PADRAO_VALOR + 1];
char szCodProdDigital[TAM_COD_PROD_DIGITAL + 1];
ScopeObtemCampoExt3(scopeHandle, Valor_transacao, 0x00, 0x00, 0x00, '\0', szValor);
ScopeObtemCampoExt3(scopeHandle, 0x00, 0x00, 0x00, Cod_Produto_Digital, '\0',
szCodProdDigital);
...
// fecha a sessão
...

```

Estados de coleta

A tabela abaixo mostra alguns estados de coleta que o fluxo poderá retornar, no modo coleta. A maioria dos estados não necessitam tratamentos específicos pela automação. Basta que mostre a mensagem na tela do operador e aguarde a digitação. Quando necessário, está descrito abaixo o tratamento específico que a automação poderá fazer.

ESTADO DE COLETA		DESCRÍÇÃO	TRATAMENTO
HEX	DECIMAL		
0xFC61	64609	Coleta Bandeira Pode ocorrer caso nenhuma bandeira tenha sido informada (0) e mais de uma rede esteja habilitada para	Pode ser utilizada a função ScopeObtemListaRedesConceudoDigital() para obter uma lista de redes, bandeiras e serviços habilitados.

		operações de conteúdo digital.	
OxFCFB 0x0004	64763 4	Seleciona Provedor Digital Pode ocorrer caso a rede e bandeira selecionada possua lista de provedores disponível no TEF.	É necessário obter uma lista com dados dos provedores disponíveis, apresentá-la ao usuário e informar o código do provedor selecionado. Pode ser utilizada a função ScopeObtemProvedoresConteudoDigital() para obter uma lista de provedores habilitados.
OxFCFB 0x0005	64763 5	Seleciona Produto Digital Pode ocorrer após o estado OxFCFB 0x0005 para seleção do produto digital	É necessário obter uma lista dos produtos digitais disponíveis para o provedor selecionado, apresentá-la ao usuário e informar o código do produto digital escolhido. Pode ser utilizada a função ScopeObtemProdutosConteudoDigital() para obter uma lista de produtos digitais disponíveis para o provedor selecionado.
OxFCFB 0x0006	64763 6	Coleta Produto Digital Pode ocorrer caso a rede e bandeira selecionada não possua lista de produtos disponível no TEF. Neste caso o código será coletado diretamente do usuário por meio deste estado.	Deve ser informado um Código do produto digital válido para a operação.
OxFC34	64564	Coleta valor Pode ocorrer após o estado OxFCFB 0x0005 ou 0x0006.	Padrão

Efetiva Conteúdo Digital

A função ScopeEfetivaConteudoDigital () aciona o SCOPE Client para a efetivação da solicitação de um Cartão Digital, no contexto do TEF, um cupom com PIN impresso (*PoR*) pronto para ativação de créditos em plataformas de provedores de conteúdo digital, como jogos, filmes e música.

PIN-On-Receipt (PoR) indica que o código PIN será impresso diretamente no comprovante da transação. Esse método é comum em vendas de créditos pré-pagos, onde o cliente recebe o PIN no recibo emitido pelo sistema de automação comercial.

Protótipo em C/C++

```
LONG EXPORT ScopeEfetivaConteudoDigital (WORD _Servico, char *_Valor, WORD _CodBandeira, char *_CodProdDigital)
```

Parâmetros

[in]	WORD	_Servico	Para efetuar a operação de Efetivação de Conteúdo Digital usar o serviço 192 S_SOLICITACAO_PIN_IMPRESSO
[in]	String com o máximo de 12 dígitos	_Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 125,00 = "12500"). Caso não informado haverá solicitação no fluxo de coleta.
[in]	WORD	_CodBandeira	O Código da Bandeira indicado poderá ser B_EPAY [221], ou outra que venha a suportar esta operação no futuro. Este código é necessário para possibilitar o roteamento da transação para a rede adquirente alvo do serviço PIN Impresso (S_SOLICITACAO_PIN_IMPRESSO).
[in]	String com o máximo de 13 dígitos	_CodProdDigital	O Código do Produto Digital desejado deverá ser informado. O código poderá ser EAN13, UPC12 ou outro formato compatível com a operação. O TEF reconhece o formato esperado (pela quantidade de dígitos) e valida o Dígito Verificador.

Retorno

Ver [tabela de código de retorno](#).

NOTA: Os valores dos parâmetros `_Valor`, `_CodBandeira` e `_CodProdDigital` podem ser obtidos através de `ScopeConsultaConteudoDigital`.

Exemplo

```

...
unsigned short servico = S_SOLICITACAO_PIN_IMPRESSO; // valor 192 (servico.h)
unsigned short bandeira = B_EPAY; // valor 221 (bandeira.h)

...
// abre sessão
...
retorno = ScopeEfetivaConteudoDigital(servico, "10000", bandeira, "7894900011517");
...
// processa a transação
...
// obtem e imprime o cupom com o PIN fornecido pela rede autorizadora
...
// fecha a sessão
...

```

Estados de coleta

A tabela abaixo mostra alguns estados de coleta que o fluxo poderá retornar, no modo coleta. A maioria dos estados não necessitam tratamentos específicos pela automação. Basta que mostre a mensagem na tela do operador e aguarde a digitação. Quando necessário, está descrito abaixo o tratamento específico que a automação poderá fazer.

ESTADO DE COLETA		DESCRIÇÃO	TRATAMENTO
HEX	DECIMAL		
0xFC61	64609	Coleta Bandeira Pode ocorrer caso não tenha sido informada por parâmetro.	Padrão
0xFC34	64564	Coleta valor Pode <u>não</u> ocorrer se valor recebido como parâmetro, neste caso apenas confirma através do estado 0xFCFF.	Padrão
0xFC13	64531	Coleta Forma de Pagamento Opcionalmente pode ser chamado ScopeForneceCampo(SCOPE_DADOS_PAGAMENTO_EX) para informar dados de pagamento.	Padrão
0xFC02	64514	Imprime Cupom Imprime cupom contendo PIN	Imprimir via cliente. Opcionalmente, pode ser impressa a via do lojista, porém o PIN permanecerá mascarado nessa via. Não há cupom reduzido.

IMPORTANTE: A confirmação para o Multi-TEF terá efetividade apenas na transação de pagamento (se eletrônica), pois a transação de PIN impresso já é automaticamente confirmada na segunda perna (se aprovada).

Recomendações de integração para o Multi-TEF com PIN Impresso (*PIN Pintring*)

1. Como a transação de PIN Impresso possui somente duas pernas, o sistema de automação deverá restringir o processo de venda em somente um produto por vez, isto para garantir segurança ao processo.
2. O desfazimento de transação de pagamento deve ser inibido caso a transação de PIN Impresso tenha sido aprovada pela administradora.
3. A transação de pagamento deve sempre anteceder a transação de PIN Impresso
4. Em caso de reimpressão de comprovante de PIN Impresso, é recomendado exigir autorização de um supervisor.
5. É sugerido que o comprovante de pagamento (se eletrônico) e de PIN impresso sejam impressos em conjunto.

ScopeObtemListaRedesConteudoDigital

Aciona o SCOPE Client para obter uma lista de redes, bandeiras e serviços habilitados para conteúdo digital.

Protótipo

```
LONG EXPORT ScopeObtemListaRedesConteudoDigital (WORD _Formato, char *_Versao, char *_Buffer, WORD
_TamBuffer);
```

Parâmetros

[in]	char	_Formato	Fixo REDES_PRODDIG_FORMATO_1 (scopeapi.h)
[in]	String	_Versao	Fixo "01"
[out]	String	_Buffer	Área reservada pela aplicação para receber a lista de redes, bandeiras e serviços disponíveis para operações de conteúdo digital. O layout dos dados é definido pela estrutura stPRODDIG_ListaProdutos em scopeapi.h
[in]	WORD	*_TamBuffer	A aplicação deve informar ao SCOPE o tamanho de _Buffer.

Retorno

Ver [tabela de código de retorno](#).

Exemplo

```
stPRODDIG_ListaProdutos sProdutos;
int iRetorno = RCS_SUCESSO;
iRetorno=
ScopeObtemListaRedesConteudoDigital(REDES_PRODDIG_FORMATO_1,"01",(char*)&sProdutos,
sizeof(sProdutos));
...
// Exibe o conteúdo de sProdutos
```

ScopeObtemProvedoresConteudoDigital

Aciona o SCOPE Client para obter lista de provedores de conteúdo digital.

Protótipo

```
LONG EXPORT ScopeObtemProvedoresConteudoDigital (WORD _Formato, char *_Buffer, int *_Size);
```

Parâmetros

[in]	WORD	_Formato	Fixo 0x00
[out]	String	_Buffer	Área reservada pela aplicação para receber a lista de provedores. O layout dos dados é definido pela estrutura stRecProvDigital em scopeapi.h
[in/out]	Int	*_Size	A aplicação deve informar ao SCOPE o tamanho da área reservada. Caso informe _Size com conteúdo zero (0), o tamanho exato necessário será retornado na própria variável.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFF16	65302	Memória insuficiente
0xFF41	65345	Dados indisponíveis

Nesta versão inicial, a função espera o valor `0x00` como valor o primeiro parâmetro `_Formato` .

Os dados solicitados serão retornados no buffer de saída indicado pelo segundo parâmetro `_Buffer` .

É necessário que a variável `_Buffer` já esteja previamente alocada, e o tamanho máximo do buffer deve ser especificado no terceiro parâmetro `_Size` .

NOTA: Recomenda-se que o sistema integrado invoque a função inicialmente com o parâmetro _Size definido como 0. Isso permitirá obter o tamanho necessário para a alocação de memória da variável _Buffer.

Exemplo

```

unsigned char* buffer = NULL;
int iSize = 0;
int iRetorno = RCS_SUCESSO;
...
// Primeira chamada para receber o tamanho do buffer a ser alocado
iRetorno = ScopeObtemProvedoresConteudoDigital(0x00, buffer, &iSize);
if (Size > 0)
{
    buffer = (unsigned char*)malloc((size_t) iSize + 1);
    memset(buffer, 0x00, Size+1);
    // Segunda chamada para receber os dados
    iRetorno = ScopeObtemProvedoresConteudoDigital(0x00, buffer, &iSize);
}

```

```

...
// Exibe o conteúdo de buffer

```

ScopeObtemProdutosConteudoDigital

Aciona o SCOPE Client para obter lista de produtos de conteúdo digital.

Protótipo

```
LONG EXPORT ScopeObtemProdutosConteudoDigital (WORD _Formato, char *_Buffer, int *_Size);
```

Parâmetros

[in]	WORD	_Formato	Fixo 0x00
[out]	String	_Buffer	Área reservada pela aplicação para receber a lista de produtos. O layout dos dados é definido pela estrutura stRecProdDigital em scopeapi.h
[in/out]	Int	*_Size	A aplicação deve informar ao SCOPE o tamanho da área reservada. Caso informe _Size com conteúdo zero (0), o tamanho exato necessário será retornado na própria variável.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFF16	65302	Memória insuficiente
0xFF41	65345	Dados indisponíveis

Exemplo

```

unsigned char* buffer = NULL;
int iSize = 0;
int iRetorno = RCS_SUCESSO;
...
// Primeira chamada para receber o tamanho do buffer a ser alocado
iRetorno = ScopeObtemProdutosConteudoDigital(0x00, buffer, &iSize);
if (Size > 0)
{
    buffer = (unsigned char*)malloc((size_t) iSize + 1);
    memset(Buffer, 0x00, Size+1);
    // Segunda chamada para receber os dados
}

```

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

```
iRetorno = ScopeObtemProdutosConteudoDigital(0x00, buffer, &iSize);
}

...
// Exibe o conteúdo de buffer
```



Funções diversas

Neste capítulo, abordamos diversas funções de uso frequente por diversas empresas, mas que não conseguimos categorizá-las.

Dados da transação

Após uma transação ou até mesmo durante o processamento desta, pode-se querer guardar dados da transação junto aos dados de venda, utilizados pela aplicação para controle ou relatório das vendas.

Na interface coleta quase todos os dados da transação a aplicação consegue obter, uma vez que é ela que provê a entrada de dados e repassa o que foi coletado para o SCOPE. A aplicação pode conhecer o dado coletado a partir do código do estado de coleta que o SCOPE passa para ela coletar.

Para obter estes dados, primeiramente, a aplicação solicita um *handle* para a transação. Uma vez obtido este *handle*, a aplicação solicita os dados da transação, um de cada vez ou todos de uma única vez.

Obtendo handle

A aplicação deverá solicitar o *handle* com a função `ScopeObtemHandle()`. Dependendo do momento em que coletará os dados, o parâmetro desta função será diferente:

- valor 0: este valor deverá ser passado para uma referência da última transação ou da transação em andamento após a solicitação de autorização;
- valor 8: este valor é utilizado para a obtenção de dados de uma transação após queda de energia.
- valor 9: este valor é usado quando o momento de obtenção de algum dado acontecer durante o processamento da transação, antes da solicitação de autorização.

Protótipo

LONG EXPORT ScopeObtemHandle (LONG Desloc)
--

Parâmetros

[in]	LONG	Desloc	Momento de obtenção do handle
------	------	--------	-------------------------------

Retorno

Valores maiores que 0xFFFF significam valores válidos de *handle*. Qualquer outro valor significa um código de erro (Ver [tabela de código de retorno](#)).

Possíveis Retornos de Erros

Códigos Retorno	Hexadecim al	Decimal	Significado
0xFA01	64001	Parâmetro 1 inválido	
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados	
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF	
0xFE01	65025	SCOPE API não foi inicializada corretamente	
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV	
0xFE0D	65037	Não há arquivo com dados da transação anterior salvo	

Exemplo

```

...
LONG handle;
// processa a transação
handle = ScopeObtemHandle(0);
if(handle > 0xFFFF)
{
    // obtém os dados desejados
}
else
{
    // erro! não conseguiu o handle
}
...
// fecha a sessão
...

```

Obtendo os campos

Após obter um *handle* válido, a aplicação está apta para o recebimento dos dados da transação. Estes dados podem ser obtidos de uma vez só ou campo a campo, ou até mesmo agrupados em partes. Isto é conseguido através de campos com combinação de bits.

Cada bit das variáveis de máscara passadas para o SCOPE possui um significado que podem ser combinados de qualquer maneira. A relação completa de campos e os bits que representam estes campos podem ser encontrados na seção [Dados disponíveis das transações](#). Para casos em que a aplicação solicitar vários dados, o SCOPE os retornará num único buffer separado por um caractere separador que a aplicação passará para o SCOPE usar. Mesmo que a aplicação solicitar um dado que não exista na transação, o SCOPE separará o campo em que estaria o dado com o caractere separador.

A sequência dos dados que serão devolvidos para a aplicação será a partir da máscara 1, depois para a máscara 2 e logo depois para a máscara 3 na sequência do bit menos significativo para o mais significativo. É importante observar que a máscara 3 somente pode ser obtida através da função ScopeObtemCampoExt2(), juntamente com as máscaras 1 e 2.

Protótipo

```
LONG EXPORT ScopeObtemCampoExt (LONG Handle,
    LONG Masc1,
    LONG Masc2,
    char FieldSeparator,
    char *Buffer)
```

Parâmetros

[in]	LONG	Handle	Handle da transação, uma referência ao registro contendo os dados da transação, que deve ser um valor retornado pela função ScopeObtemHandle()
[in]	LONG	Masc1	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 1)
[in]	LONG	Masc2	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 2)
[in]	char	FieldSeparator	Caractere de separação usado quando Masc1 e/ou Masc2 forem do tipo bitwise recuperando mais de um campo no mesmo buffer
[out]	string	Buffer	Buffer de destino que conterá os dados que o SCOPE retornará

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFA04	64004	Parâmetro 4 inválido
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
0xFE01	65025	SCOPE API não foi inicializada corretamente

Exemplo

```
...
long h;
char aux[128];
handle = ScopeObtemHandle(0);
if(handle > 0xFFFF)
{
    // obtendo o nsu da transação
    memset(aux, '\0', sizeof(aux));
```

```

ScopeObtemCampoExt(h, 0x00000004, 0x00 , ';', aux);
    ArmazenaCampo(aux); // exemplo de retorno "015236:"

    // obtendo o código da operadora de celular
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt(h, 0x00, 0x00004000, ':', aux);
    ArmazenaCampo(aux); // exemplo de retorno "02:"

    // código e nome da bandeira
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt(h, 0x00040000 | 0x00800000, 0x00 , '!', aux);
    ArmazenaCampo(aux); // exemplo de retorno "002:Mastercard:"
}

else
{
    // erro! não conseguiu o handle
}
...

```

Protótipo

```

LONG EXPORT ScopeObtemCampoExt2 (LONG Handle,
    LONG Masc1,
    LONG Masc2,
    LONG Masc3,
    char FieldSeparator,
    char *Buffer)

```

Parâmetros

[in]	LONG	Handle	Handle da transação, uma referência ao registro contendo os dados da transação, que deve ser um valor retornado pela função ScopeObtemHandle()
[in]	LONG	Masc1	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 1)
[in]	LONG	Masc2	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 2)
[in]	LONG	Masc3	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 3)
[in]	char	FieldSeparator	Caractere de separação usado quando Masc1 e/ou Masc2 forem do tipo bitwise recuperando mais de um campo no mesmo buffer
[out]	string	Buffer	Buffer de destino que conterá os dados que o SCOPE retornará

Retorno

Ver [tabela de código de retorno](#).

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Possíveis Retornos de Erros

Códigos Retorno Hexadecim al	Decimal	Significado
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFA04	64004	Parâmetro 4 inválido
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
0xFE01	65025	SCOPE API não foi inicializada corretamente

Exemplo

```

...
long h;
char aux[128];
handle = ScopeObtemHandle(0);
if(handle > 0xFFFF)
{
    // obtendo o nsu da transação
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt2(h, 0x00000004, 0x00 , 0x00 ,':', aux);
    ArmazenaCampo(aux); // exemplo de retorno "015236:"


    // obtendo o código da operadora de celular
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt2(h, 0x00, 0x00004000, 0x00 ':", aux);
    ArmazenaCampo(aux); // exemplo de retorno "02:"


    // código e nome da bandeira e dados da plataforma promocional
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt2(h, 0x00040000 | 0x00800000, 0x00 , 0x00000001':', aux);
    ArmazenaCampo(aux); // exemplo de retorno "002:Mastercard:"
}

else
{
    // erro! não conseguiu o handle
}
...

```

Protótipo

```
LONG EXPORT ScopeObtemCampoExt3 (LONG Handle,
    LONG Masc1,
    LONG Masc2,
    LONG Masc3,
    LONG Masc4,
    char FieldSeparator,
    char *Buffer)
```

Parâmetros

[in]	LONG	Handle	Handle da transação, uma referência ao registro contendo os dados da transação, que deve ser um valor retornado pela função ScopeObtemHandle()
[in]	LONG	Masc1	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 1)
[in]	LONG	Masc2	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 2)
[in]	LONG	Masc3	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 3)
[in]	LONG	Masc4	Máscara de bits indicando os campos de dados da transação a serem recuperados que pode ser um valor tipo bitwise (Ver bits relacionados na tabela Dados disponíveis das transações na parte da máscara 4)
[in]	char	FieldSeparator	Caractere de separação usado quando Masc1 e/ou Masc2 forem do tipo bitwise recuperando mais de um campo no mesmo buffer
[out]	string	Buffer	Buffer de destino que conterá os dados que o SCOPE retornará

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Hexadecim al	Decimal	Significado
0xFA01	64001		Parâmetro 1 inválido
0xFA02	64002		Parâmetro 2 inválido
0xFA03	64003		Parâmetro 3 inválido
0xFA04	64004		Parâmetro 4 inválido
0xFB09	64265		Estourou o número máximo de TEF numa sessão multi-TEF
0xFE01	65025		SCOPE API não foi inicializada corretamente

Exemplo

```

...
long h;
char aux[128];
handle = ScopeObtemHandle(0);
if(handle > 0xFFFF)
{
    // obtendo o nsu da transação
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt2(h, 0x00000004, 0x00 , 0x00 ':' , aux);
    ArmazenaCampo(aux); // exemplo de retorno "015236:";

    // obtendo o código da operadora de celular
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt2(h, 0x00, 0x00004000, 0x00 ':' , aux);
    ArmazenaCampo(aux); // exemplo de retorno "02:";

    // código e nome da bandeira e dados da plataforma promocional
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt2(h, 0x00040000 | 0x00800000, 0x00 , 0x00000001 ':' , aux);
    ArmazenaCampo(aux); // exemplo de retorno "002:Mastercard:";
}
else
{
    // erro! não conseguiu o handle
}
...

```

CUIDADO: tente alocar um buffer razoavelmente grande conforme os dados que se deseja recuperar, pois o SCOPE não alocará buffer.

Fornecendo informações extras para a transação

Para passar alguma informação que não é padrão do fluxo da transação existe a função [ScopeForneceCampoEx\(\)](#). São vários os dados que são recebidos por esta função.

Descrição dos tipos de dados

O SCOPE conhece cada tipo de dado pelo valor recebido no primeiro parâmetro, pois cada tipo possui um significado diferente, não tendo relação entre si. Os dados são esperados no segundo parâmetro (buffer).

No arquivo *ScopeApi.h* que segue com as bibliotecas do SCOPE Client, existe uma enumeração contendo todos os valores que indicam cada tipo de dado, cuja declaração é:

```

typedef enum {
    SCOPE_DADO_MIN,
    SCOPE_DADO_EMPRESA,
    SCOPE_DADO_PLANO_LOJA,
    SCOPE_DADO_ATRIBUTOS_APPLIC,
    SCOPE_DADO_TRILHA_01,
    SCOPE_DADO_REG_FORMA_PAGTO,
    SCOPE_DADO_AUT_SUPERVISOR,
    SCOPE_DADO_IMPRIME_CHEQUE,
    SCOPE_DADO_SEPARADOR_LINHA,
}

```

```

SCOPE_DADOS_APPLIC,
SCOPE_DADOS_PAGAMENTO,
SCOPE_DADOS_APPLIC_CBD,
SCOPE_DADO_DATA_MOVIMENTO,
SCOPE_DADO_NSU_CUPOM_FISCAL,
SCOPE_DADOS_AUTOMACAO,
SCOPE_DADOS_COD_AUT_MEDICAMENTOS,
SCOPE_DADOS_LISTA_MEDICAMENTOS,
SCOPE_AUTOMACAO_PERMITE_SALDO_VOUCHER,
SCOPE_COD_TABELA_PARCELE_MAIS,
SCOPE_DADOS_PAGAMENTO_EX,
SCOPE_AUTOMACAO_PARTICIPA_PPONLINE,
SCOPE_PERMITIR_SAQUE,
SCOPE_RESULTADO_VALIDACAO,
SCOPE_DADOS_ECF_BENEFICIO,
SCOPE_NOME_PRODUTO,
SCOPE_DADOS_CARTAO_PRESENTE,
SCOPE_DADOS_LISTA_PRECOS,
SCOPE_DADOS_LISTA_MERCADORIAS,
SCOPE_DADOS_LEADER,
SCOPE_DADOS_RAMO_ATIVIDADE,
SCOPE_AUTOMACAO_PERMITE_SOMENTE_TX_EMBARQUE,
SCOPE_DADO_ADICIONAL,
SCOPE_SOFT_DESCRIPTOR,
SCOPE_CNPJ_FILIAL,
SCOPE_SOFT_DESCRIPTOR_149,
SCOPE_DADOS_CARTEIRA_VIRTUAL,
SCOPE_SOFT_DESCRIPTOR_108,
SCOPE_SOFT_DESCRIPTOR_155,
SCOPE_AUTOMACAO_PERMITE_DESCONTO,
SCOPE_SOFT_DESCRIPTOR_135,
SCOPE_MCC,
SCOPE_LEITURA_SCANNER,
SCOPE_DADOS_LISTA_MERCADORIAS_CV,
SCOPE_DADO_MAX,
} tDadoForneceCampo;

```

Onde:

- SCOPE_DADO_MIN (ID 0): limite inferior do domínio de valores que o primeiro parâmetro pode receber
- SCOPE_DADO_EMPRESA (ID 1): buffer onde os 7 primeiros bytes representam uma string com o número do cupom/nota fiscal e outros 7 bytes com a série da nota fiscal (exemplo: "123456'\0'598654'\0'" = {'1', '2', '3', '4', '5', '6', '\0', '5', '9', '8', '6', '5', '4', '\0'})
- SCOPE_DADO_PLANO_LOJA (ID 2): string com o máximo 12 bytes de comprimento contendo o valor total do ticket, dado complementar utilizado pela função ScopeCompraCDC (exemplo: "10000" para R\$100,00)
- SCOPE_DADO_ATRIBUTOS_APPLIC (ID 3): ver Atributos da aplicação
- SCOPE_DADO_TRILHA_01 (ID 4): string com a trilha 01 do cartão lida no PIN-Pad
- SCOPE_DADO_REG_FORMA_PAGTO (ID 5): string de no máximo 30 bytes de comprimento com a descrição do registrador de forma de pagamento
- SCOPE_DADO_AUT_SUPERVISOR (ID 6): string de no máximo 15 bytes de comprimento com a autorização do supervisor
- SCOPE_DADO_IMPRIME_CHEQUE (ID 7): string com 1 byte de comprimento informando se imprime ("S") ou não ("N") o cheque

- SCOPE_DADO_SEPARADOR_LINHA (ID 8): string com 1 byte de comprimento parametrizando o separador de linha do cupom (exemplo: "@"). Atualmente, suporta apenas o caractere '@', sendo que para qualquer outro caractere parametrizado, ou caso esta parametrização não seja efetuada, será considerado o caractere '\n' como finalizador de linha do cupom.
- SCOPE_DADOS_APPLIC (ID 9): tipo de terminal ('A' para ATM ou 'P' para PDV) e usuário
- SCOPE_DADOS_PAGAMENTO (ID 10): dados do pagamento (confronte com SCOPE_DADOS_PAGAMENTO_EX)
- SCOPE_DADOS_APPLIC_CBD (ID 11): desenvolvido para a rede CBD e utilizado também pela rede SmartNet (VR), sendo um string com o primeiro char informando o tipo de terminal (numero de '1' a '5') e o restante da string representando a data de movimento no formato MMDD, como demonstrado a seguir:

Campo	Formato	Descrição
Tipo Terminal	n1	Tipos de Terminal que podem ser configurados: 1 = Frente de loja, (default) 2 = Magazine, 3 = Batimento/retaguarda 4 = Posto de gasolina 5 = WEB
Data de Movimento	n4	Data de Movimento no formato MMDD

- SCOPE_DADO_DATA_MOVIMENTO (ID 12): não mais suportado
- SCOPE_DADO_NSU_CUPOM_FISCAL (ID 13): não mais suportado
- SCOPE_DADOS_AUTOMACAO (ID 14): buffer onde os 21 primeiros bytes representam uma string com o fabricante do software de automação e outros 21 bytes com a versão do software da automação
- SCOPE_DADOS_COD_AUT_MEDICAMENTOS (ID 15): string com 13 bytes que contém o código de autorização da PBM
- SCOPE_DADOS_LISTA_MEDICAMENTOS (ID 16): buffer com a lista de medicamentos da PBM
- SCOPE_AUTOMACAO_PERMITE_SALDO_VOUCHER (ID 17): Através deste campo, a automação informa ao Scope se permite que o pagamento efetuado através de voucher utilize o valor disponível no saldo (neste caso deve informar 'S') ou aceita somente pagamento para os casos em que o saldo do voucher seja igual ou superior ao valor da transação (neste caso deve informar 'N'). Observe que este campo deve ser informado somente na abertura de conexão entre a aplicação e o Scope Cliente (ScopeOpen).
- SCOPE_COD_TABELA_PARCELE_MAIS (ID 18): Automação pode fornecer um código de tabela a ser usado na transação de Parcele Mais, indicando o custo financeiro.
- SCOPE_DADOS_PAGAMENTO_EX (ID 19): dados do pagamento para transações que necessitam do número do cartão (confronte com SCOPE_DADOS_PAGAMENTO)

- SCOPE_AUTOMACAO_PARTICIPA_PPONLINE (ID 20): responsável por informar se a automação comercial está apta a trabalhar com a plataforma promocional.
- SCOPE_PERMITIR_SAQUE (ID 21): em transações de débito à vista com a rede Cielo, ao fornecer o valor '1' indicará que o SCOPE pode pedir a digitação do valor do saque se configurado pela rede e ao fornecer '0' o SCOPE não perguntará pelo saque, mesmo configurado pela rede Cielo e no SCOPECNF.
- SCOPE_RESULTADO_VALIDACAO (ID 22): após a validação do saque pela automação, está passará o valor '1' aprovando o valor do saque digitado pelo portador do cartão no pinpad ou passará o valor '0' recusando o valor do saque solicitado.
- SCOPE_DADOS_ECF_BENEFICIO (ID 23): fornecer os dados da ECF e do comprovante para a transação.
- SCOPE_DADOS_LISTA_PRECOS (ID 26): fornecer os dados da Lista de Atualização de Preços das Mercadorias.
- SCOPE_DADOS_LISTA_MERCADORIAS (ID 27): fornecer os dados da Lista de Mercadorias Consumidas na Aplicação.
- SCOPE_AUTOMACAO_PERMITE_SOMENTE_TX_EMBARQUE (ID 30): string com 1 byte de comprimento informando se permite ("S") ou não ("N") - responsável por informar se a automação comercial está apta a trabalhar com transações de valor principal zerada e com Taxa de Embarque coletada – Esta opção é usada com Crédito – Vendas IATA.
- SCOPE_DADO_ADICIONAL (ID 31): Reservado para uso futuro.
- SCOPE_SOFT_DESCRIPTOR (ID 32): Soft Descriptor a ser usado para a transação. Válido até o próximo ScopeClose. Só tem efeito se a rede permitir tal configuração. Campo opcional, geralmente usado em plataformas de Market Place. Pode ser usado configuração por contrato, ou por PDV. Campo do tipo string com tamanho de 40. Para maiores detalhes verifique o Manual de Configuração.
- SCOPE_CNPJ_FILIAL (ID 33). CNPJ da filial a ser usado na transação. Válido até o próximo ScopeClose. Só tem efeito se a rede permitir tal configuração. Campo opcional, geralmente usado em plataformas de Market Place. Campo do tipo string com tamanho de 14. Deve ser fornecido apenas números, sem formatação. Se não fornecido e transação exigir essa informação, será usado o CNPJ configurado para a filial do SCOPE.
- SCOPE_AUTOMACAO_PERMITE_DESCONTO (ID 38): Utilizada para cartões Frota da rede Ticket Log. Os valores possíveis são:
 - "0" = Não permite desconto (default);
 - "1" = Permite desconto apenas no valor total;
 - "2" = Permite desconto "por item";

- SCOPE_LEITURA_SCANNER (ID 44): fornecer os dados capturados pela leitura do scanner
 - Tipo do conteúdo:
 - "1" = QRCode
 - "2" = Código Barras
 - "3" = Token
 - Modo de entrada:
 - "1" = Scanner
 - "2" = Digitado
- SCOPE_DADOS_LISTA_MERCADORIAS_CV (ID 45): fornecer os dados da Lista de Mercadorias consumidas nas transações de Carteira Virtual.
- SCOPE_DADO_MAX: limite superior do domínio de valores que o primeiro parâmetro pode receber;

Atributos da aplicação

Permite implementar atributos (campos) dinamicamente, que são vinculados às transações efetuadas pelo SCOPE. Existem três módulos envolvidos por esta funcionalidade:

- ScopeCNF: cadastrando os atributos (ver no documento de instalação, configuração e administração, informações sobre cadastramento)
- Aplicação do PDV: deverá usar a função ScopeForneceCampo()
- ScopeADM: irá exibir as transações vinculadas aos atributos cadastrados (ver o documento de instalação, configuração e administração para informações sobre relatório)

Exemplo: você poderá consultar as transações efetuadas pelo SCOPE, vinculadas a outras informações úteis a empresa, como: número da nota fiscal, do pedido, da compra, nome e telefone do cliente, etc.

O formato dos dados deve ser uma string que obedece ao seguinte padrão:

"nnc1tt1xx..x1c2tt2xx..x2cnttnxx..xn"

, onde:

- nn = quantidade de atributos (tamanho 2)
- c1 = código do atributo 1 (tamanho 2)
- tt1 = tamanho do atributo 1 (tamanho 3)
- xx..x1 = atributo 1 (tamanho tt1)
- c2 = código do atributo 2 (tamanho 2)
- tt2 = tamanho do atributo 2 (tamanho 3)
- xx..x2 = atributo 2 (tamanho tt2)
- ...
- cn = código do atributo n (tamanho 2)
- tt n = tamanho do atributo n (tamanho 3)

- xx..xn = atributo n (tamanho ttn)

IMPORTANTE: por razões de otimização de tráfego, o SCOPE Client envia estes dados para o servidor na solicitação da segunda pré-TEF. Portanto, a aplicação deve fornecer este buffer antes da realização da mesma.

Atrelando dados à transação

Passar algum dado ao SCOPE com a função ScopeForneceCampo() exige que ela seja chamada antes do envio da segunda pré-TEF.

Protótipo

LONG EXPORT ScopeForneceCampoEx (char TypeField, void *StructField, int _SizeField)

Parâmetros

[in]	char	TypeField	Tipo de dado
[in]	void *	StructField	Ponteiro para os dados
[in]	int	_SizeField	Tamanho dos dados

Protótipo antigo (mantido apenas por compatibilidade)

LONG EXPORT ScopeForneceCampo (char TypeField, void *StructField)

Parâmetros antigos

[in]	char	TypeField	Tipo de dado
[in]	void *	StructField	Ponteiro para os dados

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Hexadecim al	Decimal	Significado
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados	
0xFE01	65025	SCOPE API não foi inicializada corretamente	
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um handle sem ter feito nenhuma transação desde última conexão com o ScopeSRV	
0xFE0D	65037	Não há arquivo com dados da transação anterior salvo	
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF	

Exemplo

Digamos que a cada venda paga com o cartão de crédito, a empresa deseja que seja atrelada a transação o número de telefone do cliente e o número do pedido. Para isso, o administrador do SCOPE criou dois atributos no ScopeCNF (Figura 8).



Figura 8: atributos da aplicação cadastrados no ScopeCNF

A aplicação coletará estes dados sempre que houver uma venda com cartão de crédito. Exemplificando na linguagem C uma compra com cartão de crédito (interface Coleta):

```

...
// executa transação (crédito, débito, etc.)
BOOL bDadosAplicFornec = FALSE;
char szAtributosAplic[40];
...

// inicia iteração
cod_coleta = ScopeStatus();
if(cod_coleta == 0xFE00)
{
    ...
    if (!bDadosAplicFornec)
    {
        // coletou o telefone do cliente e número do pedido
        ...
        // montou o buffer do atributo da aplicação
        // (ex. "0201014(11) 6097-123402006123456" e guardou em szAtributosAplic
        ...
        ScopeForneceCampo(SCOPE_DADO_ATRIBUTOS_APPLIC, szAtributosAplic);
        bDadosAplicFornec = TRUE;
    }
    ...
    // entrega o que coletou ao SCOPE na função abaixo
    ScopeResumeParam(cod_coleta, dados, modo_entrada, acao);
    ...
}
// finaliza iteração

```

Separando o buffer do exemplo acima conforme o padrão descrito, obtemos:

- 02 – 2 campos no buffer
- 01 – este campo representa o que está cadastrado como código 1 no ScopeCNF

- 014 – o atributo que seguirá contém 14 bytes
- (11) 6097-1234 – valor do atributo de código 1 com 14 bytes
- 02 – este campo representa o que está cadastrado como código 2 no ScopeCNF
- 006 – o atributo que seguirá contém 6 bytes
- 123456 – valor do atributo de código 2 com 6 bytes

ScopeGetLastMsg

Obtém as últimas mensagens a serem mostradas para o operador e/ou cliente.

Protótipo

```
LONG EXPORT ScopeGetLastMsg (ptCOLETA_MSG ptParamColetaMsg)
```

Parâmetros

[out]	Ponteiro para uma área com o formato da estrutura stCOLETA_MSG	ptParamColetaMsg	Estrutura contendo as mensagens do operador e do cliente (tanto a linha 1 como a linha 2). As mensagens são finalizadas pelo caractere nulo ('\0')
-------	--	------------------	--

Estruturas de apoio

```
typedef struct _stCOLETA_MSG {
    char Op1[64];
    char Op2[64];
    char Cl1[64];
    char Cl2[64];
} stCOLETA_MSG, *ptCOLETA_MSG;
```

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFE01	65025	SCOPE API não foi inicializada corretamente

ScopeGetCheque

Obtém os parâmetros para impressão de cheque.

Protótipo

```
LONG EXPORT ScopeGetCheque(ptPARAM_CHEQUE ptParamCheque)
```

Parâmetros

[out]	Ponteiro para uma área com o formato da estrutura ptPARAM_CHEQUE	ptParamCheque	Dados para impressão do cheque
-------	--	---------------	--------------------------------

Estruturas de apoio

```
typedef struct _stPARAM_CHEQUE {
    char Banco[4]; // número do banco
    char Agencia[5]; // número da agência
    char NumCheque[13]; // número do cheque
    char Valor[13]; // valor do cheque
    char BomPara[9]; // data do cheque
    char CodAut[11]; // cód. de autor. retornado pelas autorizadoras
    char Municipio[41]; // município
    short Ordem; // reservado
} stPARAM_CHEQUE, *ptPARAM_CHEQUE;
```

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE04	65028	Não existe transação suspensa

ScopeAtualizaValor

Atualiza ou fornece o novo valor da transação. Esta função deve ser executada após a função de transação e antes do envio da solicitação de autorização para a Autorizadora.

Protótipo

```
LONG EXPORT ScopeAtualizaValor(char *Valor)
```

Parâmetros

[in]	String	Valor	Valor da transação. Deve ser uma string de até 12 bytes.
------	--------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno	Significado
-----------------	-------------

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFE01	65025	SCOPE API não foi inicializada corretamente

ScopeGarantiaDescontoCheque

Aciona o SCOPE client para efetuar uma transação de Garantia de Cheque ou de Desconto Antecipado de Cheque (Factoring).

Protótipo

```
LONG EXPORT ScopeGarantiaDescontoCheque (char *Valor)
```

Parâmetros

[in]	String	Valor	Valor da transação. Deve ser uma string de até 12 bytes.
------	--------	-------	--

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
OxFE0D	65037	Não há arquivo com dados da transação anterior salvo
OxFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
OxFF3A	65338	Erro interno na execução da coleta

ScopeTransacaoFinanceira

Aciona o SCOPE Client para efetuar uma transação financeira.

Protótipo

```
LONG EXPORT ScopeTransacaoFinanceira (char *Valor, WORD Servico)
```

Parâmetros

[in]	String	Valor	Valor da transação
[in]	WORD	Serviço	Informa qual o serviço a ser adotado. Caso 0 (zero) a coleta do serviço será feita pelo procedimento usual (ver Códigos dos serviços)

RetornoVer [tabela de código de retorno](#).**Estados de coleta**

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar no modo coleta para o serviço 74 – Saque.

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFE00	65024	Transação em andamento
0xFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
0xFC00	64512	Coleta cartão
0xFC85	64645	Coleta cartão digitado
0xFC01	64513	Coletar validade do cartão
0xFC34	64564	Coleta valor
0xFC09	64521	Coletar se a transação será a vista ou não
0xFC47	64583	Coleta quantidade de parcelas e aceita 1 parcela
0xFCF9	64761	Coleta opção de parcela grátis (1-Sim 0-Nao)
0xFC27	64551	Coletar ciclos a pular
0xFC1F	64543	Carência (em dias) para a 1ª parcela
0xFC26	64550	Coleta Categoria de Plano
0xFC4C	64588	Coleta se cliente deseja aderir ao seguro
0xFC1B	64539	Imprimir e/ou exibir dados da consulta
0xFC02	64514	Imprimir Cupom
0xFCFF	64767	Mostrar Informações e aguardar confirmação do operador

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar no modo coleta para o serviço 73 – Simulação de Saque.

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFE00	65024	Transação em andamento
0xFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
0xFC34	64564	Coleta valor
0xFC00	64512	Coleta cartão
0xFC85	64645	Coleta cartão digitado
0xFC01	64513	Coletar validade do cartão
0xFC47	64583	Coleta quantidade de parcelas e aceita 1 parcela
0xFC27	64551	Coletar ciclos a pular
0xFC1F	64543	Carência (em dias) para a 1ª parcela
0xFC26	64550	Coleta Categoria de Plano
0xFC1B	64539	Imprimir e/ou exibir dados da consulta
0xFC02	64514	Imprimir Cupom
0xFCFF	64767	Mostrar Informações e aguardar confirmação do operador

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar no modo coleta para o serviço 179 – Consulta Fatura Detalhada.

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFE00	65024	Transação em andamento

Oxfcfe	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
OxFC00	64512	Coleta cartão
OxFC85	64645	Coleta cartão digitado
OxFC01	64513	Coletar validade do cartão
OxFC1B	64539	Imprimir e/ou exibir dados da consulta
OxFC02	64514	Imprimir Cupom
OxFCFF	64767	Mostrar Informações e aguardar confirmação do operador

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar no modo coleta para o serviço 71 – Consulta Extrato Resumido.

Códigos Retorno		Significado
Hexadecim al	Decimal	
OxFE00	65024	Transação em andamento
OxFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
OxFC00	64512	Coleta cartão
OxFC85	64645	Coleta cartão digitado
OxFC01	64513	Coletar validade do cartão
OxFC1B	64539	Imprimir e/ou exibir dados da consulta
OxFC02	64514	Imprimir Cupom
OxFCFF	64767	Mostrar Informações e aguardar confirmação do operador

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar no modo coleta para o serviço 176 – Saque via QRCode.

Códigos Retorno		Significado
Hexadecim al	Decimal	
OxFE00	65024	Transação em andamento
Oxfcfe	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
OxFCF3	64755	Exibir QRCode (a aplicação recebe uma string para gerar a imagem)
Oxfc02	64514	Imprimir Cupom
Oxfcff	64767	Mostrar Informações e aguardar confirmação do operador

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
OxFA01	64002	Parâmetro 2 inválido
OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFB01	64257	Não foi possível criar a "thread" na coleta de dados
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual

ScopeSaque

Aciona o SCOPE Client para efetuar um Saque com cartão de Crédito. Esta função é equivalente à chamar a ScopeTransacaoFinanceira com código de serviço = "74 – Saque Crédito".

Protótipo

```
LONG EXPORT ScopeSaque (char *Valor)
```

Parâmetros

[in]	String	Valor	Valor da transação
------	--------	-------	--------------------

Retorno

Ver [tabela de código de retorno.](#)

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual

ScopelInvestimento

Aciona o SCOPE Client para efetuar um investimento.

Protótipo

```
LONG EXPORT ScopelInvestimento (char *Valor, WORD Servico)
```

Parâmetros

[in]	String	Valor	Valor da transação
[in]	WORD	Servico	Informa qual o serviço a ser adotado. Caso 0 (zero) a coleta do serviço será feita pelo procedimento usual (ver Códigos dos Serviços)

Retorno

Ver [tabela de código de retorno.](#)

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	

OxFA01	64002	Parâmetro 2 inválido
OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um handle sem ter feito nenhuma transação desde última conexão com o ScopeSRV
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFF3A	65338	Erro interno na execução da coleta
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFFFF	65535	Erro genérico

ScopeObtemCartaoInvestimento

Aciona o SCOPE Client para obter o número do cartão a partir do CPF do cliente. Viabiliza efetuar uma transação de aplicação pelo número do CPF.

Protótipo

```
LONG EXPORT ScopeObtemCartaoInvestimento (char *CPF,
                                         char *Buf,
                                         WORD TamBuf)
```

Parâmetros

[in]	String	CPF	CPF do cliente
[out]	String	Buf	Área reservada pela aplicação para receber o cartão (ou lista de cartões). O layout dos dados foi definido pelo SAB (o SCOPE não critica nem altera este layout).
[in]	WORD	TamBuf	A aplicação deve informar ao SCOPE o tamanho da área reservada

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno Hexadecim al	Decimal	Significado
OxFA01	64001	Parâmetro 1 inválido
OxFE00	65024	A transação em andamento – a aplicação deve aguardar

OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado
OxFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere “#” como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
OxFFFF	65535	Erro genérico
OxFF5E	65374	Erro ao desmontar a estrutura ISO

ScopeResumoOperacoes

Utilizada nos casos em que a rede ou correspondente bancário oferece a possibilidade de obtenção de um resumo das operações realizadas.

Protótipo

```
LONG EXPORT ScopeResumoOperacoes (WORD CodServico, WORD CodBandeira)
```

Parâmetros

[in]	WORD	CodServico	Código do serviço do produto a ser executado. Use 0 (zero) para indicar que o código do serviço deverá ser coletado durante o fluxo da transação.
[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
OxFA01	64001	Parâmetro 1 inválido
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
OxFE06	65030	Logon duplicado
OxFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
OxFE08	65032	POS não cadastrado

OxFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
OxFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
OxFE0D	65037	Não há arquivo com dados da transação anterior salvo
OxFB01	64257	Não foi possível criar a "thread" na coleta de dados
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
OxFF3A	65338	Erro interno na execução da coleta
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFF60	65376	Função indisponível
OxFFFF	65535	Erro genérico

ScopePagamento

Utilizada para pagamento de conta (título ou convênio) ou fatura, através de uma rede autorizadora ou por intermédio de correspondente bancário. Os serviços atuais disponíveis são “pagamento de conta com cartão”, “pagamento de conta sem cartão” e “pagamento de fatura”. O pagamento de conta com cartão é usado quando a rede exige uma determinada bandeira (Electron, Cheque Eletrônico), sendo que neste caso o SCOPE valide o BIN do cartão. Pagamento de conta sem cartão é quando o pagamento pode ser realizado através de dinheiro, cheque ou TEF externa (outra transação realizada separadamente). Pagamento de fatura é usado para o “recebimento próprio” de fatura de cartão de crédito. Neste caso, o código de barras não é obrigatório, já que o pagamento pode ser através do número do cartão (digitado ou lido). Para a rede Credsystem 63 juntamente com a bandeira Credsystem PL 344, foi acrescentada a opção de pagamento de fatura por CPF caso o cliente não desejar parar através do número do cartão.

No pagamento de fatura por cartão para a Credsystem, o cliente poderá pagar tanto pelo cartão do titular quanto pelo adicional. Já no pagamento de fatura por CPF, terá que ser sempre o CPF do titular. A coleta do CPF será sempre no PIN PAD.

Protótipo

```
LONG EXPORT ScopePagamento (WORD CodServiço, WORD CodBandeira)
```

Parâmetros

[in]	WORD	CodServiço	Código do serviço do grupo “Pagamento de Contas” a ser executado. Use 0 (zero) para indicar que a decisão do serviço será através de coleta.
[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE00	65024	Transação em andamento
0xFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
0xFC48	64584	Coleta código de barras
0xFC85	64645	Coleta cartão digitado
0xFC34	64564	Coleta valor
0xFC41	64577	Coleta data (formato DDMMAAAA)
0xFC09	64521	Coletar se a transação será a vista ou não
0xFC0A	64522	Decisão se irá parcelar pela administradora ou loja
0xFC0E	64526	Coleta quantidade de parcelas
0xFC02	64514	Imprime Cupom
0xFCFF	64767	Mostrar Informações e aguardar confirmação do operador

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFE0D	65037	Não há arquivo com dados da transação anterior salvo
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
0xFF3A	65338	Erro interno na execução da coleta
0xFF5E	65374	Erro ao desmontar a estrutura ISO
0xFF60	65376	Função indisponível
0xFFFF	65535	Erro genérico

ScopeServicoTecnico

Utilizada para execução de um serviço técnico disponível para determinada rede autorizadora. Temos os seguintes serviços técnicos: Baixa de OS, Teste de Comunicação, Estatística e Injeção de Chaves DUKPT.

O serviço técnico de Injeção de Chaves DUKPT insere a senha DUKPT no pinpad através de uma transação 0900/0800 Injeção de Chaves e sua resposta 0910/0810 quando o pinpad não possui a senha, isto é o serviço não atualiza senha; para isso utiliza-se a função VerificaChaveDUKPT para verificar se o pinpad possui senha DUKPT.

Protótipo

```
LONG EXPORT ScopeServicoTecnico (WORD CodServiço, WORD CodBandeira)
```

Parâmetros

[in]	WORD	CodServiço	Código do serviço do grupo "Técnico" a ser executado. Use 0 (zero) para indicar que a decisão do serviço será através de coleta.
[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado. Use 0 (zero) para indicar que o código da bandeira deverá ser coletado.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Hexadecim al	Decimal	Códigos Retorno	Significado
0xFA01	64001	Parâmetro 1 inválido	
0xFE00	65024	A transação em andamento – a aplicação deve aguardar	
0xFE01	65025	SCOPE API não foi inicializada corretamente	
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV	
0xFE06	65030	Logon duplicado	
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server	
0xFE08	65032	POS não cadastrado	
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis	
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).	
0xFE0D	65037	Não há arquivo com dados da transação anterior salvo	
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados	
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida	
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF	
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada	
0xFF0A	65290	Banco de dados off-line	
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual	

0xFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
0xFF3A	65338	Erro interno na execução da coleta
0xFF5E	65374	Erro ao desmontar a estrutura ISO
0xFF60	65376	Função indisponível
0xFFFF	65535	Erro genérico

ScopeAtualizaParametrosChip

Utilizada quando parâmetros do chip do cartão necessitam ser atualizados. Esta função inicia o fluxo de coleta necessário para a atualização de parâmetros do chip do cartão.

Redes que utilizam esta função são: Ticket Edenred e Cielo R2014.

Coleta básica da transação

- Solicita ao portador a inserção do cartão no pinpad
- Solicita a senha
- Solicita a retirada do cartão do pinpad

Protótipo

	<pre>LONG EXPORT ScopeAtualizaParametrosChip(char *UsoFuturo1, char *UsoFuturo2)</pre>
--	---

Parâmetros

[in]	char	UsoFuturo1	Reservado para uso futuro. Usar como parâmetro uma string vazia.
[in]	char	UsoFuturo2	Reservado para uso futuro. Usar como parâmetro uma string vazia.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecim	Decimal	
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida

0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
0xFFFF	65535	Erro genérico

ScopeVersao

Esta função retorna a versão do Scope.

Protótipo

	LONG EXPORT ScopeVersao(char *_VersaoScope, int _TamBufVersao)
--	--

Parâmetros

[out]	char	_VersaoScope	Informa o valor da versão do Scope em formato string e null-terminated. Em caso de erro, o buffer passado não sofre alteração.
[in]	int	_TamBufVersao	Contém o valor do tamanho da área que contém a versão. Deve conter o valor mínimo de 12 bytes.

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido

ScopeAtualizaPrecosMercadorias

Utilizada quando os Preços das Mercadorias do Estabelecimento necessitam ser atualizados na rede Ticket Edenred. Esta função inicia o fluxo de coleta necessário para a Atualização de Preços das Mercadorias.

Redes que utilizam esta função são: Ticket Edenred.

Protótipo

	LONG EXPORT ScopeAtualizaPrecosMercadorias (WORD _CodBandeira, char *UsoFuturo1)
--	---

Parâmetros

[in]	WORD	_CodBandeira	Código da Bandeira (Ex. "228" - Ticket Car)
[in]	char	UsoFuturo1	Reservado para uso futuro. Usar como parâmetro uma string vazia.

Retorno

Ver [tabela de código de retorno](#).

Estados de coleta

A tabela abaixo mostra os estados de coleta que o fluxo poderá retornar, no modo coleta

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFCCA	64714	Coleta da Lista de Atualização de Preços de Mercadorias

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida
0xFF00	65280	ScopeSrv <i>off-line</i> ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF0A	65290	Banco de dados off-line
0xFF0C	65292	Transação não implementada
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior

ScopeServicosGenericos

Utilizada para execução de um serviço genérico, que difere dos fluxos de outras funções. Temos os seguintes serviços genéricos disponíveis atualmente: Consulta de Vale Gás, Compra de Vale Gás, Troco Surpresa, Autorização de Voucher e Carga de Cartão Presente (somente com bandeira Blackhawk).

Alguns [serviços](#) são válidos somente para algumas bandeiras e devem ser usadas na combinação correta.

O valor é opcional para um serviço de consulta. Para os outros serviços, o valor se fornecido será utilizado na transação.

Serviço		
Código	Scopeapi.h	Descrição
25	S_COMPRA_VALE_GAS	Compra Vale Gás usado com B_VALEGAS
103	S_CONSULTA_VALE_GAS	Consulta Vale Gás usado com B_VALEGAS
118	S_TROCO_SURPRESA	Troco Surpresa usado com B_GETNET
128	S_AUTORIZACAO_VOUCHER	Autorização Voucher usado com B_SAVS
132	S_CARGA_CARTAO_PRESENTE	Carga de Cartão Presente usado com B_BLACKHAWK

Protótipo

```
LONG EXPORT ScopeServicosGenericos (WORD CodServiço, WORD CodBandeira, char* Valor)
```

Parâmetros

[in]	WORD	CodServiço	Código do serviço a ser executado.
[in]	WORD	CodBandeira	Código da bandeira do produto a ser executado..
[in]	Char	Valor	Valor a ser utilizado (de acordo com serviço desejado)

Retorno

Ver [tabela de código de retorno](#).

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFE00	65024	A transação em andamento – a aplicação deve aguardar
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFEOA	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFE0D	65037	Não há arquivo com dados da transação anterior salvo
0xFB01	64257	Não foi possível criar a "thread" na coleta de dados
0xFB03	64259	Erro ao verificar mensagem – mensagem inválida

OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
OxFF0A	65290	Banco de dados off-line
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
OxFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
OxFF3A	65338	Erro interno na execução da coleta
OxFF5E	65374	Erro ao desmontar a estrutura ISO
OxFF60	65376	Função indisponível
OxFFFF	65535	Erro genérico

ScopeObtemDadosAdicionais

Utilizada quando a Aplicação de Automação Comercial necessita de informações adicionais retornadas por uma transação de consulta.

Redes que utilizam esta função são: Bradescard.

Protótipo

```
LONG EXPORT ScopeObtemDadosAdicionais ( WORD _Formato, char * _Buffer, int * _Size)
```

Parâmetros

[in]	WORD	_Formato	Indica o layout dos dados retornados em _Buffer .
[out]	char	_Buffer	Contém os dados adicionais retornados pela transação de consulta.
[in/out]	int	_Size	Tamanho dos dados contidos em _Buffer .

Retorno

Ver [tabela de código de retorno](#).

Utilização

A função deve ser chamada uma primeira vez para que a Aplicação saiba a quantidade de bytes a ser alocada para o buffer.

Possíveis Retornos de Erros

Códigos Retorno Hexadecim al	Decimal	Significado
0xFF16	65302	Memória insuficiente para conter os dados adicionais.

Exemplo

```
...
unsigned char *buffer = NULL;
int Size;
...
// abre sessão
...
retorno = ScopeCompraCartaoCredito(Valor, TxServico);
```

```

...
Size = 0;
// Primeira chamada para receber o tamanho do buffer a ser alocado
retorno = ScopeObtemDadosAdicionais(0x00, Buffer, &Size);
if (Size > 0)
{
    Buffer = (unsigned char *) malloc(Size+1);
    memset(Buffer, 0x00, Size+1);

    // Segunda chamada para receber os dados da autorizadora
    retorno = ScopeObtemDadosAdicionais(0x00, Buffer, &Size);
}
...
...
// processa a transação
...
...
// fecha a sessão
...

```

ScopeObtemProdutosFrota

Utilizada quando a Aplicação de Automação Comercial necessita obter os produtos (ou mercadorias) disponíveis para um cartão do tipo "Frota".

Redes que utilizam esta função são: Ticket Log.

Protótipo

	LONG EXPORT ScopeObtemProdutosFrota (BYTE CódigoRede, char *Produtos)
--	---

Parâmetros

[in]	BYTE	CódigoRede	Atualmente, somente a rede TICKETLOG permite essa a chamada dessa função. Deve ser passado o código 157.
[out]	char *	Produtos	Buffer de destino que conterá os dados que o SCOPE retornará, conforme formato abaixo.

Retorno

Ver [tabela de código de retorno](#).

Utilização

Seu uso é opcional. Usada somente se a automação comercial precisar saber quais são os produtos Ticket Log disponíveis. Tais produtos são recebidos pelo SCOPE através da transação online Inicialização de Tabelas.

Possíveis Retornos de Erros

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFF16	65302	Memória insuficiente para conter os dados.

Formato para receber lista de produtos disponíveis

```
#define TICKETLOG_QTD_MAX_PRODUTOS_DISPONIVEIS 99

typedef struct {
    charCodigo[4 + 1];
    charDescricao[40 + 1];
} stTicketLogProduto, *LPTicketLogProduto;

typedef struct {
    charQtdProdutos[2 + 1];
    stTicketLogProduto sProduto[TICKETLOG_QTD_MAX_PRODUTOS_DISPONIVEIS];
} stTicketLogProdutosDisponiveis, *LPTicketLogProdutosDisponiveis;
```

ScopeObtemDadosCredarioCredito

Utilizada para obtenção de dados de credíario (principalmente dados de parcelamento) dentro dos fluxos de coleta de simulação e contratação de credíario com cartão de crédito.

Protótipo

LONG EXPORT ScopeObtemDadosCredarioCredito(WORD _Versao, WORD _Formato, int *_QtRegistros, char *_Buffer, int *_Size)

Parâmetros

[in]	WORD	_Versao	Versão do formato desejado. Conforme tabela abaixo.
[in]	WORD	_Formato	Formato desejado. Conforme tabela abaixo.
[in,out]	Int *	_QtRegistros	Quantidade de registros (planos) desejados. 0 – Modo de recuperação um a um. Retorna o primeiro registro disponível em _Buffer, e indica como valor de saída a quantidade ainda disponível. Esta quantidade deve ser utilizada como valor de entrada na próxima chamada, e assim por diante até que sejam retornados todos os registros. Utilizar como condição de término do laço o valor igual a zero (0). 999 – Modo completo. Solicita todos os registros disponíveis gravados e único bloco em _Buffer. Neste caso o valor de saída será a quantidade de registros presentes em _Buffer.
[out]	char *	_Buffer	Buffer de saída, conterá um ou mais registros com dados de credíario. A variável deverá ser

			pré alocada pelo chamador e possuir tamanho suficiente. Verificar o formato dos dados no tópico <i>Formato dados crediário</i> .
[int,out]	Int *	_Size	Indica o tamanho de _Buffer. Caso o tamanho seja insuficiente, a função falhará com retorno 0xFF16, e informará em _Size o tamanho necessário.

Formatos

A tabela abaixo relaciona os Formatos e Versões permitidos.

Parâmetro		Descrição
_Versao	_Formato	
0x00	0x00	Formato em TLV.
0x01	0x01	Formato em estrutura do tipo <code>stDadosCreditorioCredito</code> .

Estados de coleta

A tabela abaixo mostra o estado de coleta recomendado no fluxo para obter estes dados, no modo coleta.

Códigos Retorno		Significado
Hexadecim	Decimal	
0xFC1B	64539	Imprime consulta

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Códigos Retorno		Significado
Hexadecim	Decimal	
0xFE01	65025	SCOPE API não foi inicializada corretamente
0xFF07	65287	Parâmetros de entrada incompatíveis
0xFF41	65345	Dados indisponíveis
0xFF16	65302	Memória insuficiente
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFEOB	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFFOA	65290	Banco de dados off-line
*0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFF60	65376	Função indisponível
0xFFFF	65535	Erro genérico

Formato TLV dos Dados Crediário

Este formato é retornado quando os parâmetros _Versao e _Formato são preenchidos respectivamente com os valores 0x00 e 0x00.

Atualmente pode ser usado para as redes adquirentes GETNETLAC versão de especificação 2.99 (ou superior), REDE versão de especificação L0701 (ou superior) e CIELO versão de especificação R2024 (ou superior).

Campo	Formato	Descrição
TAG	N4	Código
LENGTH	N4	Tamanho do valor a seguir
VALUE	ANS..LENGTH	Valor

TAG		Formato	Descrição
1000		TLV	Ocorrência de dados de parcelamento
1000	1010	N2	Quantidade de parcelas
	1020	ANS..99	Texto para menu
	1030	TLV	Ocorrência de Campo
	1031	ANS..99	Nome
	1032	ANS..99	Valor

As TAGs 1000 e 1030 contém outras tags e podem **repetir** até a quantidade de ocorrências disponíveis. As TAGs subordinadas à tag 1000 (1010, 1020 e 1030) não necessariamente seguirão a **mesma ordem** especificada acima no buffer retornado. Ou seja, a ordem das TAGs não é fixa.

Exemplo 1 – Obtendo todos os registros em única vez

```

...
char *buffer = NULL;
int iQtRegistros = 999; //todos
int iTam = 0;
char szOpcão[2 + 1] = {0};
...
// obtém o tamanho necessário para todos os dados
ret= ScopeObtemDadosCredíarioCredito(0,0,&iQtRegistros,buffer, &iTam);
...
if (iTam> 0)
buffer = (char *)calloc(iTam, sizeof(char));
else
return; //sem dados

// Segunda chamada para receber a cópia dos dados em buffer
ret= ScopeObtemDadosCredíarioCredito(0,0,&iQtRegistros,buffer, &iTam);

printf("Buffer recebido: %s\n", buffer);
...
//Montar menu com as opções recebidas
...
//Coletar opção desejada (szOpcão) Ex. "01"
scanf("%2s", szOpcão);

//Exibir detalhes da opção escolhida
...
free(buffer); //libera o buffer
...
//Informar ao SCOPE o número da opção escolhida
ScopeForneceCampo(SCOPE_DADOS_CREDÍARIO, &szOpcão); //Ex. "01"

```

Exemplo 2 – Obtendo todos os registros um por vez

```

...
char *buffer = NULL;

```

```

int iQtRegistros = 0; //primeiro
int iTam = 0;
char szOpcão[2 + 1] = {0};
...
do {
    iTam = 0;
    // obtém o tamanho necessário para os dados de uma ocorrência
    ret= ScopeObtemDadosCredíarioCrédito(0,0,&iQtRegistros,buffer, &iTam);
    ...
    if (iTam> 0)
        buffer = (char *)calloc(iTam, sizeof(char));
    else
        return; //sem dados

    // Segunda chamada para receber a cópia dos dados em buffer
    ret= ScopeObtemDadosCredíarioCrédito(0,0,&iQtRegistros,buffer, &iTam);

    printf("Buffer recebido: %s\n", buffer);
    //adicionar os dados da ocorrência em uma lista dinâmica
    free(buffer); //libera o buffer
} while(iQtRegistros > 0);
...
//Montar menu com as opções recebidas
...
//Coletar opção desejada (szOpcão) Ex. "01"
scanf("%2s", szOpcão);

//Exibir detalhes da opção escolhida
...
//Informar ao SCOPE o número da opção escolhida
ScopeForneceCampo(SCOPE_DADOS_CREDÍARIO,&szOpcão); //Ex. "01"

```

Formato em Estrutura dos Dados Crediário

Este formato é retornado quando os parâmetros _Versão e _Formato são preenchidos respectivamente com os valores 0x01 e 0x01.

Para este formato, o parâmetro _QtRegistros é ignorado, visto que sempre será retornada a estrutura completa.

Atualmente pode ser usado para a rede adquirente REDE versão de especificação L0701 (ou superior) e CIELO versão de especificação R2024 (ou superior).

Observação: Caso necessário usar este formato para a rede adquirente GETNETLAC, entre em contato com a NCR.

A grande vantagem deste formato é a possibilidade da aplicação de automação comercial identificar cada campo, além de permitir um processamento mais simples se comparado com o formato TLV.

As estruturas descritas abaixo estão definidas no arquivo 'scopeapi.h'.

```

//ScopeObtemDadosCredíarioCrédito
//Formato padrão SCOPE com identificação dos campos
//Opção ao formato com campos TVL, cuja identificação não é possível
typedef struct
{
    char szQtdParcelas[2 + 1];           // Quantidade de parcelas
}

```

```

char szValorParcela[12 + 1];           // Valor da parcela, sendo 2 casas decimais
char szTaxaJurosMensal[8 + 1];         // Taxa de juros mensal, sendo 2 casas decimais
char szImpostosIOF[12 + 1];            // Impostos + IOF, sendo 2 casas decimais
char szCETAnual[12 + 1];                // CET anual, sendo 2 casas decimais
char szValorTotal[12 + 1];               // Valor total, sendo 2 casas decimais
} stOcorrenciaCredarioCredito, *LPOcorrenciaCredarioCredito;

//Dados formatados para eventual display
typedef struct
{
    char szMenu[38 + 1];      // Opção a ser exibida no Menu (qtd parcelas x valor parcela)
    char szQtdParcelas[38 + 1];
    char szValorParcela[38 + 1];
    char szTaxaJurosMensal[38 + 1];
    char szImpostosIOF[38 + 1];
    char szCETAnual[38 + 1];
    char szValorTotal[38 + 1];
} stDisplayCredarioCredito, *LPDisplayCredarioCredito;

#define CREDARIO_CREDITO_MAX_OCORRENCIAS 3

typedef struct
{
    WORD Versao;                  // Versao = 1 (para viabilizar eventuais evoluções futuras)
    WORD Formato;                 // Formato = 1 (para viabilizar eventuais evoluções futuras)
    WORD QtdOcorrencias;          // Máximo de 3 ocorrências
    stOcorrenciaCredarioCredito sOcorrencia[CREDARIO_CREDITO_MAX_OCORRENCIAS];
    stDisplayCredarioCredito sDisplay[CREDARIO_CREDITO_MAX_OCORRENCIAS];
} stDadosCredarioCredito, *LPDadosCredarioCredito;

```

Exemplo – Obtendo os dados de credíario no formato `stDadosCredarioCredito`:

```

...
stDadosCredarioCredito sDadosCredarioCredito;
LPDisplayCredarioCredito lpDisplay;
char szPlanoEscolhido[2 + 1];
int Size = sizeof(sDadosCredarioCredito);

LONG RC = ScopeObtemDadosCredarioCredito(1, 1, 0, (char*)&sDadosCredarioCredito, &Size);

if (RC == RCS_SUCESSO)
{
    //Montar menu com as opções recebidas
    char szMsg[80];
    char szLinhaMenu[50 + 1];

    for (int i = 0; i < sDadosCredarioCredito.QtdOcorrencias; i++)
    {
        lpDisplay = (LPDisplayCredarioCredito)&(sDadosCredarioCredito.sDisplay[i]);
        sprintf(szLinhaMenu, " [%d]- %s\n", i + 1, lpDisplay->szMenu);
        printf(szLinhaMenu);
    }

    //Obtem seleção do operador
    while (TRUE)

```

```

{
    printf("\nOpcão: ");
    char cldPlano = getche();
    int iPlano = (int)cldPlano - 48;

    if ((iPlano < 1) || (iPlano > sDadosCredarioCredito.QtdOcorrencias))
    {
        printf("\nOpcão invalida!\n");
        continue;
    }
    break;
}

//Exibe detalhes do plano escolhido
lpDisplay = (LPDisplayCredarioCredito)&(sDadosCredarioCredito.sDisplay[iPlano-1]);

sprintf(szMsg, "\n    PLANO ESCOLHIDO: %s", lpDisplay->szMenu); printf(szMsg);
printf("\n\n    DETALHES:");
printf("\n    -----");
sprintf(szMsg, "\n    %s%c", lpDisplay->szTaxaJurosMensal, '%'); printf(szMsg);
sprintf(szMsg, "\n    %s%c", lpDisplay->szCETAnual, '%'); printf(szMsg);
sprintf(szMsg, "\n    %s%c", lpDisplay->szImpostosIF, '%'); printf(szMsg);
sprintf(szMsg, "\n    %s", lpDisplay->szValorTotal); printf(szMsg);

//Aguarda confirmação do operador
...
...

//Informar ao SCOPE o número da opção escolhida
sprintf(szPlanoEscolhido, "%02d", iPlano);
ScopeForneceCampo(SCOPE_DADOS_CREDARIO, szPlanoEscolhido);
}
...

```

ScopeCompraCartaoCreditoCPF

Com comportamento similar à função ScopeCompraCartaoCredito, descrita no tópico Cartão de Credito, é utilizada para compra com uso do CPF do cliente para consulta prévia de seu(s) cartão(ões) disponível(eis) junto ao emissor de forma online. A compra ocorre com o número do cartão obtido ou selecionado, além da coleta de dados adicionais para validação positiva (e opcionalmente a senha do cartão).

Protótipo

LONG EXPORT ScopeCompraCartaoCreditoCPF (char *Valor, char *TxServico, WORD CodBandeira, char *CPF, BYTE OpcColetaCPF)

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

[in]	String	TxServiço	Valor da taxa de serviço
[in]	WORD	CodBandeira	Código da bandeira solicitante (exemplo: B_BRADESCARD)
[in]	String	CPF	CPF do portador cartão (opcional). Caso não informado será coletado durante o fluxo através do PINPad ou Teclado Operador conforme indicado no próximo parâmetro (OpcColetaCPF).
[in]	BYTE	OpcColetaCPF	Opção de coleta do CPF. Caso um CPF tenha sido informado no parâmetro anterior (CPF) o valor deste parâmetro será ignorado. Valores possíveis: 0 - Deve ser coletado no PINPad 1 - Deve ser coletado pelo operador (Teclado PDV).

Estados de coleta

A tabela abaixo destaca alguns estados excepcionais de coleta que poderão ocorrer no fluxo, no modo coleta. O conjunto completo dos estados de coleta se encontram na descrição da função ScopeCompraCartaoCredito.

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFC66	64614	Coleta CPF no teclado operador
0xFCF4	64756	Coleta dado especial (seleção do número do cartão). Através do modo menu dinâmico, utilizar as funções ScopeMenuRecuperaltens e ScopeMenuSelecionaltem para montagem de menu de seleção do número do cartão. O tipo de dado especial será TDE_SEL, e o FormatoDado será TM_SELECAO.

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Para demais retornos vide ScopeCompraCartaoCredito no tópico Cartão de Crédito.

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFA04	64004	Parâmetro 4 inválido
0xFA05	64005	Parâmetro 5 inválido

ScopeConsultaPgtoFatura

Consulta que retorna informações de uma fatura de cartão, principalmente o valor a pagar, entre outros dados disponibilizados pela rede adquirente/emissor.

Protótipo

```
LONG EXPORT ScopeConsultaPgtoFatura(WORD Servico, WORD CodBandeira)
```

Parâmetros

[in]	WORD	Servico	Informa qual o serviço a ser adotado. Atualmente usar somente: S_CONSULTA_PGTO_FATURA=178
[in]	WORD	CodBandeira	Código da bandeira solicitante (exemplo: B_BRADESCARD)

Estados de coleta

A tabela abaixo lista os estados de coleta que poderão ocorrer no fluxo, no modo coleta.

Códigos Retorno		Identificador	Significado
Hexadecimal	Decimal		
0xFC48	64584	TC_COLETA_COD_BARRAS	Coleta Código de barras
0xFC61	64609	TC_COLETA_COD_BANDEIRA	Coleta Código de Bandeira
0xFC00	64512	TC_CARTAO	Coleta cartão
0xFCB6	64694	TC_COLETA_CARTAO_DIGITADO_PP	Coleta cartão digitado no PINPAD
0FCB9	64697	TC_CONFIRMA_CARTAO_DIGITADO_PP	Coleta confirmação de cartão digitado no PINPAD
0xFC85	64645	TC_CARTAO_DIGITADO	Coleta cartão digitado
0FCFE	64766	TC_INFO_RET_FLUXO	Mostrar informações e retornar fluxo para o cliente SCOPE
0FC34	64564	TC_COLETA_VALOR	Coleta Valor
0FC01	64513	TC_VALIDADE_CARTAO	Coleta validade do cartão
0FC11	64529	TC_SENHA	Coleta senha
0FCFD	64765	TC_COLETA_EM_ANDAMENTO	Coleta em andamento
0FCFC	64764	TC_COLETA_CARTAO_EM_ANDAMENTO	Coleta de cartão em andamento
0FC02	64514	TC_IMPRIME_CUPOM	Imprime cupom
0FCFF	64767	TC_INFO_AGU_CONF_OP	Mostrar informações e aguardar confirmação do operador
0FC7D	64637	TC_COLETA_CANCELAR_TRANSACAO	Coleta confirmação de cancelamento da operação.

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Alguns dos principais retornos.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFE01	65025	SCOPE API não inicializada
0xFE00	65024	Transação em andamento

ScopeConsultaPgtoFaturaCPF

Com comportamento similar à função ScopeConsultaPgtoFatura, é utilizada para consulta de pagamento de fatura com uso do CPF do cliente. Com o CPF é realizada uma consulta prévia e online de seu(s) cartão(ões) disponível(eis) junto ao emissor. A consulta de pagamento de fatura ocorre então com o número do cartão obtido ou selecionado, além da coleta de dados adicionais para validação positiva.

Protótipo

```
LONG EXPORT ScopeConsultaPgtoFaturaCPF (WORD Servico, WORD CodBandeira, char *CPF, BYTE OpcColetaCPF)
```

Parâmetros

[in]	WORD	Servico	Informa qual o serviço a ser adotado. Atualmente usar somente: S_CONSULTA_PGTO_FATURA=178
[in]	WORD	CodBandeira	Código da bandeira solicitante (exemplo: B_BRADESCARD)
[in]	String	CPF	CPF do portador cartão (opcional). Caso não informado será coletado durante o fluxo através do PINPad ou Teclado Operador conforme indicado no próximo parâmetro (OpcColetaCPF).
[in]	BYTE	OpcColetaCPF	Opção de coleta do CPF. Caso um CPF tenha sido informado no parâmetro anterior (CPF) o valor deste parâmetro será ignorado. Valores possíveis: 0 - Deve ser coletado no PINPad 1 - Deve ser coletado pelo operador (Teclado PDV).

Estados de coleta

A tabela abaixo destaca estados adicionais de coleta que poderão ocorrer no fluxo, no modo coleta. Os demais estados se encontram na descrição da função ScopeConsultaPgtoFatura.

Códigos Retorno		Significado
Hexadecim al	Decimal	

OxFC66	64614	Coleta CPF no teclado operador
OxFCF4	64756	Coleta dado especial (seleção do número do cartão). Através do modo menu dinâmico, utilizar as funções ScopeMenuRecuperaltens e ScopeMenuSelecionaltem para montagem de menu de seleção do número do cartão. O tipo de dado especial será TDE_SEL, e o FormatoDado será TM_SELECAO.

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Para demais retornos vide ScopeConsultaPgtoFatura.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFA04	64004	Parâmetro 4 inválido

ScopeConsultaSaldoCreditoCPF

Com comportamento similar à função ScopeConsultaSaldoCredito, é utilizada para consulta de saldo de cartão de crédito com uso do CPF do cliente. Com o CPF é realizada uma consulta prévia e online de seu(s) cartão(ões) disponível(eis) junto ao emissor. A consulta de saldo crédito ocorre então com o número do cartão obtido ou selecionado, além da coleta de dados adicionais para validação positiva.

Protótipo

```
LONG EXPORT ScopeConsultaSaldoCreditoCPF (WORD CodBandeira, char *CPF, BYTE OpcColetaCPF)
```

Parâmetros

[in]	WORD	CodBandeira	Código da bandeira solicitante (exemplo: B_BRADESCARD)
[in]	String	CPF	CPF do portador cartão (opcional). Caso não informado será coletado durante o fluxo através do PINPad ou Teclado Operador conforme indicado no próximo parâmetro (OpcColetaCPF).
[in]	BYTE	OpcColetaCPF	Opção de coleta do CPF. Caso um CPF tenha sido informado no parâmetro anterior (CPF) o valor deste parâmetro será ignorado. Valores possíveis: 0 - Deve ser coletado no PINPad

			1 - Deve ser coletado pelo operador (Teclado PDV).
--	--	--	--

Estados de coleta

A tabela abaixo destaca estados adicionais de coleta que poderão ocorrer no fluxo, no modo coleta. Os demais estados se encontram na descrição da função ScopeConsultaSaldoCredito.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFC66	64614	Coleta CPF no teclado operador
0xFCF4	64756	Coleta dado especial (seleção do número do cartão). Através do modo menu dinâmico, utilizar as funções ScopeMenuRecuperaltens e ScopeMenuSelecionaltem para montagem de menu de seleção do número do cartão. O tipo de dado especial será TDE_SEL, e o FormatoDado será TM_SELECAO.

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Para demais retornos vide ScopeConsultaSaldoCredito.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido

ScopeTransacaoFinanceiraCPF

Com comportamento similar à função ScopeTransacaoFinanceira, é utilizada para transações com uso do CPF do cliente. Com o CPF é realizada uma consulta prévia e online de seu(s) cartão(ões) disponível(eis) junto ao emissor. A transação financeira escolhida ocorre então com o número do cartão obtido ou selecionado, além da coleta de dados adicionais para validação positiva.

Protótipo

```
LONG EXPORT ScopeTransacaoFinanceiraCPF (char * Valor, WORD Servico, WORD CodBandeira, char *CPF,
BYTE OpcColetaCPF)
```

Parâmetros

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

[in]	String	Valor	Valor da transação, sem separadores. (opcional, pode ser "000")
[in]	WORD	Servico	Informa qual o serviço a ser adotado. Atualmente usar somente: S_CONSULTA_EXTRATO_RESUMIDO (71) S_CONSULTA_FATURA_DETALHADA (179)
[in]	WORD	CodBandeira	Código da bandeira solicitante. Exemplo: B_BRADESCARD (402)
[in]	String	CPF	CPF do portador cartão (opcional). Caso não informado será coletado durante o fluxo através do PINPad ou Teclado Operador conforme indicado no próximo parâmetro (OpcColetaCPF).
[in]	BYTE	OpcColetaCPF	Opção de coleta do CPF. Caso um CPF tenha sido informado no parâmetro anterior (CPF) o valor deste parâmetro será ignorado. Valores possíveis: 0 - Deve ser coletado no PINPad 1 - Deve ser coletado pelo operador (Teclado PDV).

Estados de coleta

A tabela abaixo destaca estados adicionais de coleta que poderão ocorrer no fluxo, no modo coleta. Os demais estados se encontram na descrição da função ScopeTransacaoFinanceira.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFC66	64614	Coleta CPF no teclado operador
0xFCF4	64756	Coleta dado especial (seleção do número do cartão). Através do modo menu dinâmico, utilizar as funções ScopeMenuRecuperaltens e ScopeMenuSelecionaltem para montagem de menu de seleção do número do cartão. O tipo de dado especial será TDE_SEL, e o FormatoDado será TM_SELECAO.

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Para demais retornos vide ScopeTransacaoFinanceira.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFA04	64004	Parâmetro 4 inválido
0xFA05	64005	Parâmetro 5 inválido

ScopePagamentoCPF

Com comportamento similar à função ScopePagamento, é utilizada para transações com uso do CPF do cliente. Com o CPF é realizada uma consulta prévia e online de seu(s) cartão(ões) disponível(eis) junto ao emissor. A transação de pagamento escolhida ocorre então com o número do cartão obtido ou selecionado, além da coleta de dados adicionais para validação positiva.

Protótipo

```
LONG EXPORT ScopePagamentoCPF (WORD Servico, WORD CodBandeira, char *CPF, BYTE OpcColetaCPF)
```

Parâmetros

[in]	WORD	Servico	Informa qual o serviço a ser adotado. Atualmente usar somente: S_PAGAMENTO_FATURA (91)
[in]	WORD	CodBandeira	Código da bandeira solicitante. Exemplo: B_BRADESCARD (402)
[in]	String	CPF	CPF do portador cartão (opcional). Caso não informado será coletado durante o fluxo através do PINPad ou Teclado Operador conforme indicado no próximo parâmetro (OpcColetaCPF).
[in]	BYTE	OpcColetaCPF	Opção de coleta do CPF. Caso um CPF tenha sido informado no parâmetro anterior (CPF) o valor deste parâmetro será ignorado. Valores possíveis: 0 - Deve ser coletado no PINPad 1 - Deve ser coletado pelo operador (Teclado PDV).

Estados de coleta

A tabela abaixo destaca estados adicionais de coleta que poderão ocorrer no fluxo, no modo coleta. Os demais estados se encontram na descrição da função ScopePagamento.

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFC66	64614	Coleta CPF no teclado operador
0xFCF4	64756	Coleta dado especial (seleção do número do cartão). Através do modo menu dinâmico, utilizar as funções ScopeMenuRecuperaltens e ScopeMenuSelecionaltem para montagem de menu de seleção do número do cartão. O tipo de dado especial será TDE_SEL, e o FormatoDado será TM_SELECAO.

Retorno

Ver [tabela de código de retorno.](#)

Possíveis retornos de erros

Para demais retornos vide ScopePagamento.

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFA02	64002	Parâmetro 2 inválido
0xFA03	64003	Parâmetro 3 inválido
0xFA04	64004	Parâmetro 4 inválido

ScopeConsultaPreAutorizacao

Consulta que retorna informações de uma pré-autorização de cartão de crédito, principalmente o status indicativo de pendência, cancelamento ou inexistência, entre outros dados disponibilizados pela rede adquirente/emissor.

Protótipo

```
LONG EXPORT ScopeConsultaPreAutorizacao (void)
```

Parâmetros

Não há parâmetros.

Estados de coleta

A tabela abaixo lista os estados de coleta que poderão ocorrer no fluxo, no modo coleta.

Códigos Retorno		Identificador	Significado
Hexadecimal	Decimal		
0xFC00	64512	TC_CARTAO	Coleta cartão
0xFCB6	64694	TC_COLETA_CARTAO_DIGITADO_PP	Coleta cartão digitado no PINPAD
0xFCB9	64697	TC_CONFIRMA_CARTAO_DIGITADO_PP	Coleta confirmação de cartão digitado no PINPAD
0xFC85	64645	TC_CARTAO_DIGITADO	Coleta cartão digitado
0xFCFE	64766	TC_INFO_RET_FLUXO	Mostrar informações e retornar fluxo para o cliente SCOPE
0FCFD	64765	TC_COLETA_EM_ANDAMENTO	Coleta em andamento
0FCFC	64764	TC_COLETA_CARTAO_EM_ANDAMENTO	Coleta de cartão em andamento
0FC02	64514	TC_IMPRIME_CUPOM	Imprime cupom
0FCFF	64767	TC_INFO_AGU_CONF_OP	Mostrar Informações e aguardar confirmação do operador
0FC7D	64637	TC_COLETA_CANCEL_TRANSACAO	Coleta confirmação de cancelamento da operação.

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Alguns dos principais retornos.

Códigos Retorno		Significado
Hexadecim al	Decimal	
0xFE01	65025	SCOPE API não inicializada
0xFE00	65024	Transação em andamento

Resultado

O resultado pode ser obtido por meio de um cupom a ser impresso ou exibido na tela, ou recuperado a partir dos dados da consulta de pré-autorização utilizando a função ScopeObtemCampoExt3 com Máscara 4, Bit 0x00080000.

Pode ser usada a estrutura stDadosConsultaPreAut definida em scopeapi.h.

ScopeCreditoPagamentoFatura

Transação de crédito para pagamento na modalidade fatura, conta ou carnê. É chamada de 'Compra Crédito Carnê' por algumas redes, como por exemplo a PagSeguro.

Protótipo

```
LONG EXPORT ScopeCreditoPagamentoFatura (char *Valor)
```

Parâmetros

[in]	String	Valor	Valor da transação, sem separadores. (opcional, pode ser "000")
------	--------	-------	---

Estados de coleta

A tabela abaixo lista os estados de coleta que poderão ocorrer no fluxo, no modo coleta.

Códigos Retorno		Identificador	Significado
Hexadecimal	Decimal		
0xFC00	64512	TC_CARTAO	Coleta cartão
0FCB6	64694	TC_COLETA_CARTAO_DIGITADO_PP	Coleta cartão digitado no PINPAD
0FCB9	64697	TC_CONFIRMA_CARTAO_DIGITADO_PP	Coleta confirmação de cartão digitado no PINPAD

0xFC85	64645	TC_CARTAO_DIGITADO	Coleta cartão digitado
0xFCFE	64766	TC_INFO_RET_FLUXO	Mostrar informações e retornar fluxo para o cliente SCOPE
0xFC01	64513	TC_VALIDADE_CARTAO	Coleta validade do cartão
0xFC11	64529	TC_SENHA	Coleta senha
0xFC29	64553	TC_CVV_CVC_2	Coleta do código de segurança
0xFCFD	64765	TC_COLETA_EM_ANDAMENTO	Coleta em andamento
0xFCFC	64764	TC_COLETA_CARTAO_EM_ANDAMENTO	Coleta de cartão em andamento
0xFC02	64514	TC_IMPRIME_CUPOM	Imprime cupom
0xFCFF	64767	TC_INFO_AGU_CONF_OP	Mostrar Informações e aguardar confirmação do operador
0xFC7D	64637	TC_COLETA_CANCELAR_TRANSACAO	Coleta confirmação de cancelamento da operação.

Retorno

Ver [tabela de código de retorno](#).

Possíveis retornos de erros

Alguns dos principais retornos.

Códigos Retorno		Significado
Hexadecim	Decimal	
0xFA01	64001	Parâmetro 1 inválido
0xFE01	65025	SCOPE API não inicializada
0xFE00	65024	Transação em andamento

ScopeStartLog

Função responsável por carregar a biblioteca csmsg.dll e permitir a habilitação dos logs do Scope via parâmetros das seções SCOPELOGAPI, SCOPELOGPRF, SCOPELOGSRL do scope.ini (Veja no item “Configuração do arquivo scope.ini”).

Protótipo

```
LONG EXPORT ScopeStartLog ()
```

Essa função deve ser chamada no início da execução da aplicação.

ScopeStopLog

Função responsável por liberar a biblioteca csmsg.dll.

Protótipo

```
LONG EXPORT ScopeStopLog ()
```

Essa função deve ser chamada ao final da execução da aplicação.

Operações com tratamentos específicos

O objetivo de capítulo é abordar peculiaridades de tratamentos para algumas operações como a coleta de dados no fluxo do SCOPE, interface com aplicação ou o relacionamento dos produtos da rede e as transações do SCOPE.

CORBAN – Pagamento de Contas

CORBAN – Correspondente Bancário – oferece várias funcionalidades implementadas por várias autorizadores. Porém para cada autorizadora pode haver alguma particularidade que será explorada neste tópico.

As autorizadoras que hoje implementam CORBAN pelo Scope são: REDE (V6.01 em diante), BRADESCO, UNISYS dentre outras.

O CORBAN oferece a modalidade de pagamento de contas com cartão e dinheiro para Títulos e outros Boletos, podendo ser estes de Concessionárias e Tributos, Títulos de Cobrança de vários bancos. Cada autorizadora pode ter alguma ou todas estas funcionalidades implementadas, dependendo de sua capacidade.

Para títulos bancários Scope pode necessitar realizar uma transação de consulta do título durante a transação de pagamento. Esta consulta geralmente é transparente para a aplicação e não necessita de interferência do operador.

Identificando a bandeira do Correspondente

O Pagamento de Contas deve ser feito com a identificação da bandeira na chamada da função [ScopePagamento](#). Use este código de bandeira para qualquer tipo de pagamento por um CORBAN de uma determinada autorizadora.

OBSERVAÇÃO: Esta bandeira é o equivalente ao correspondente bancário e não corresponde à bandeira do cartão que vai ser pago o título

Siga a seguinte tabela para identificar a bandeira correta a ser usada para cada autorizadora:

Rede (Código)	Código Bandeira a ser usado
BRADESCO (04)	92

REDE (103)	22
UNISYS (143)	335

Por exemplo, se necessito realizar pagamentos pelo Bradesco, tenho que indicar na função ScopePagamento além do código do serviço desejado, também o código de bandeira **92**.

Coleta de Dados do Pagamento de Contas

A transação deve ser iniciada normalmente como uma transação de pagamento [ScopePagamento\(\)](#) e com a identificação da bandeira Caso a bandeira não seja fornecida antes da transação, ela será coletada durante o fluxo (TC_COLETA_COD_BANDEIRA).

No fluxo da transação os seguintes dados podem ser coletados:

- Código de Barras da conta
- Valor a ser pago, se o código de barra não contiver o valor ou se o valor puder ser alterado pelo operador
- Vencimento do título, se o código de barra não contiver a data
- Valores de desconto ou multa
- Dados do cartão, se não for pagamento em dinheiro
- Para cobrança de alguns títulos, dados do Beneficiário, Sacador e Pagador

OBSERVAÇÃO: Para títulos sendo pagos através da REDE, a consulta do título quando realizada devolve a situação do título que pode ser Registrado ou não. Para cada situação determinadas coletas serão realizadas de acordo com regras definidas pela autorizadora ou instituição.

OBSERVAÇÃO: Para títulos sendo pagos através da Rede BRADESCO, são realizadas 2 consultas: a primeira devolve o nome da instituição e o nome do cedente e, após a confirmação do operador, a segunda consulta é enviada o código de barras do título juntamente com os valores capturados no checkout.

Se a aplicação decide responder automaticamente a coleta, ela pode responder, dentre outros, aos estados:

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFC48	64584	Coleta código de barras
0xFC34	64564	Coleta valor do título
0xFC41	64577	Coleta data vencimento (formato DDMMAAAA)
0xFCB3	64691	Coleta o valor de acréscimo / multa
0xFCB4	64692	Coleta o valor de dedução / desconto
0xFC8B	64651	Disponibiliza valores de Pagamentos de Contas
0XFCE8	64744	Coleta CPF/CNPJ do Beneficiário
0XFCE9	64745	Coleta CPF/CNPJ do Sacador
0XFCEA	64746	Coleta CPF/CNPJ do Pagador
0XFCEB	64747	Coleta valor do documento

Obtendo dados do Pagamento de Contas

A aplicação pode verificar os valores da transação de pagamento de Contas através da função [ObtemCampoExt2\(\)](#). Para isso deve tratar o estado TC_DISP_VALOR (0xFC8B), pois somente neste estado a consulta foi realizada com sucesso.

IMPORTANTE: o tratamento do estado TC_DISP_VALOR se torna obsoleta a partir da especificação REDE **V0602e** não é utilizada por nenhum outro CORBAN. No entanto, a obtenção dos dados ainda pode ser realizada no final da transação ou após ter sido realizada a consulta para CORBAN da REDE e outras autorizadoras.

OBSERVAÇÃO: no caso específico do CORBAN REDE **V0601** a consulta somente é realizada quando se trata de um boleto/título do Itaú. Para outros bancos não será realizada a consulta e os dados não estarão disponíveis.

O campo relativo para recuperar dados da conta:

Máscara 2	Dados da Consulta Fatura	0x00000080
-----------	--------------------------	------------

Veja outros dados sobre a transação no tópico [ObtemCampoExt2\(\)](#)

Esta consulta retorna os dados na seguinte estrutura *stConsultaFatura* documentada no arquivo *ScopeApi.h*. A estrutura contém os vários valores do título como por exemplo Valor a ser pago, Valor de acréscimo (multa ou juros de mora), valor do desconto além da data de vencimento.

```
#define TAM_PADRAO_VALOR 12

typedef struct {
    char szValorPagto[TAM_PADRAO_VALOR];
    char szValorOrig[TAM_PADRAO_VALOR];
    char szValorAcres[TAM_PADRAO_VALOR];
    char szValorAcresOpc[TAM_PADRAO_VALOR];
    char szValorMinimo[TAM_PADRAO_VALOR];
    char szValorDesc[TAM_PADRAO_VALOR];
    char szDataVenc[8];
} stConsultaFatura, *LPConsultaFatura;
```

onde:

- szValorPagto: valor do pagamento recalculado
- szValorOrig: valor original do pagamento – consta no título
- szValorAcres: valor do acréscimo – multa
- szValorAcresOpc: valor do acréscimo opcional – juros de mora
- szValorMinimo: valor mínimo do pagamento (sempre zero)
- szValorDesc: valor do desconto / dedução
- szDataVenc: data do vencimento no formato AAAAMMMDD

A consulta realizada também pode determinar se estes valores podem ser modificados ou não pela aplicação, mudando o fluxo caso necessário. Caso seja possível, os respectivos dados serão coletados. Em caso contrário, os valores podem ser somente exibidos.

IMPORTANTE: Caso a aplicação tente obter estas informações, mas a consulta não tenha sido realizada ainda ou os dados não estiverem disponíveis, a estrutura não será preenchida ou estará em branco.

Obtendo dados do Título - BACEN

A aplicação pode verificar as informações da transação de pagamento de Contas Registradas (BACEN) através da função [ObtemCampoExt2\(\)](#). Estes dados estarão disponíveis após uma consulta e/ou no final da transação de pagamento, pois somente neste momento a consulta foi realizada com sucesso.

O campo relativo para recuperar dados da conta:

Máscara 3	Dados Consulta BACEN	0x04000000
-----------	----------------------	------------

Veja outros dados sobre a transação no tópico [ObtemCampoExt2\(\)](#)

Esta consulta retorna os dados na seguinte estrutura stDadosConsultaBACEN documentada no arquivo *ScopeApi.h*. A estrutura é definida por layouts, e o **Layout 01 – “L01”** – corresponde ao resultado da consulta realizada e contém os seguintes dados:

```
#define TAM_PADRAO_VALOR    12
#define TAM_MAX_INSTITUICAO_EMISSORA_TIT   30
#define TAM_MAX_RAZAO_SOCIAL      50
#define TAM_MAX_NOME_FANTASIA     80
#define TAM_MAX_CPF_CNPJ         14
#define TAM_DATA_AAAAMMDD        8
#define TAM_MAX_VALOR_TITULO     17

typedef struct {
    char sInstituicaoEmissora[TAM_MAX_INSTITUICAO_EMISSORA_TIT];
    char sBeneficiarioNomeFantasia[TAM_MAX_NOME_FANTASIA];
    char sBeneficiarioRazaoSocial[TAM_MAX_RAZAO_SOCIAL];
    char sBeneficiarioCPFCNPJ[TAM_MAX_CPF_CNPJ];
    char sSacadorRazaoSocial[TAM_MAX_RAZAO_SOCIAL];
    char sSacadorCPFCNPJ[TAM_MAX_CPF_CNPJ];
    char sPagadorRazaoSocial[TAM_MAX_RAZAO_SOCIAL];
    char sPagadorCPFCNPJ[TAM_MAX_CPF_CNPJ];
} stDadosConsultaBACEN_L01, * LPDadosConsultaBACEN_L01;
```

onde:

- sInstituicaoEmissora: Instituição Emissora do título
- sBeneficiarioNomeFantasia: Nome fantasia - Beneficiário
- sBeneficiarioRazaoSocial: Razão Social – Nome Beneficiário
- sBeneficiarioCPFCNPJ: CPF/CNPJ - Beneficiário
- sSacadorRazaoSocial: Razão Social – Nome Sacador
- sSacadorCPFCNPJ: CPF/CNPJ - Sacador
- sPagadorRazaoSocial: Razão Social – Nome Pagador

- sPagadorCPFCNPJ: CPF/CNPJ - Pagador

OBSERVAÇÃO: Todos os campos de numéricos ou alfanuméricos sem conteúdo são preenchidos com brancos. Dados que não foram recuperados na consulta estarão em branco.

IMPORTANTE: Caso a aplicação tente recuperar dados de uma autorizadora que não tem consulta a títulos registrados (BACEN), ou a consulta ainda não foi realizada, a estrutura não será devolvida no ObtemCampo.

A consulta também pode determinar se estes valores podem ser modificados ou não pela aplicação, mudando o fluxo caso necessário. Caso seja possível, os respectivos dados serão coletados. Em caso contrário, os valores podem ser somente exibidos.

Além das informações da consulta realizada, a aplicação também pode obter outros dados da transação de pagamento de contas através da função ObtemCampoExt2. Com as seguintes máscaras:

Máscara 1	Nome da máscara	Descrição
0x00000002	Valor_transacao	Valor a ser pago
0x00008000	Cod_Banco	Banco
0x00010000	Cod_Agencia	Agência
0x00000080	Numero_cheque	Número do Cheque
0x10000000	Numero_CMC7	Número do CMC7
Máscara 2	Nome da máscara	Descrição
0x00000080	Dados_Consulta_Fatura	Dados da Consulta Fatura/Pagamento
Máscara 3	Nome da máscara	Descrição
0x08000000	Valor_Documento	Valor no código de barras
0x00080000	Valor_Desconto	Valor de descontos retornado pela autorizadora ou coletada.
0x00100000	Valor_Acrescimo	Valor de acréscimos retornado pela autorizadora ou coletada.
0x04000000	Dados_Consulta_BACEN	Retorna as informações de uma consulta ao título registrado - BACEN.
0x20000000	Modo_Pagamento	BRADESCO (somente) Modo de Pagamento: "01" : cheque "02": dinheiro
0x40000000	Consulta_Cedente	BRADESCO (somente) Resultado da 1ª consulta, com o seguinte layout: <code>stDadosBRADESCO_subcampo23</code> { char nome_instituicao[44] char nome_cedente_empresa[50] }
0x80000000	Data_Vencimento_CORBAN	Retorna no formato DDMMAAAA a data do Vencimento do título/conta sendo pago

Obtendo comprovante com dados do Título Registrado - BACEN

Em 2017 o SCOPE passou a suportar as novas regras de cobrança registrada BACEN para algumas autorizadoras que disponibilizam CORBAN.

A aplicação pode verificar as informações de um título registrado (dados BACEN) através da função [ObtemCampoExt2\(\)](#). Para isso deve tratar o estado TC_RESP_CONS_BACEN (0xFCEC), pois somente neste estado a consulta foi realizada com sucesso e que há informações do BACEN.

Este estado de coleta TC_RESP_CONS_BACEN necessita de confirmação do operador para continuar o fluxo de pagamento. Caso operador opte por não continuar, a transação será cancelada e o título não será pago.

O campo relativo para recuperar dados BACEN:

Máscara 3	Comprovante da Consulta BACEN Resposta_Consulta_BACEN	0x10000000
-----------	--	------------

Veja outros dados sobre a transação no tópico [ObtemCampoExt2\(\)](#)

A aplicação pode verificar os dados BACEN disponibilizados da seguinte maneira:

- em formato de comprovante;
- texto final de acordo com autorizadora;
- com linhas de no máximo 38 colunas;
- com LF (0x0A) como separador de linha.

PIX – Pagamento via QRCode, PIX Saque e PIX Troco

PIX é mais um meio de pagamento utilizado pelas carteiras virtuais. No SCOPE ele é tratado através da bandeira 432=PIX implementada por várias autorizadoras que podem seguir a especificação definida pelo BACEN (Banco Central).

Algumas autorizadoras que hoje implementam PIX pelo Scope são: ITAU (002), MERCADO PAGO (154), PAGBANK (160), Banco do Brasil (166) dentre outras.

O PIX oferece as seguintes funcionalidades:

- Pagamento via QRCode
- PIX Saque
- PIX Troco

Pagamento via QRCode

O usuário pode pagar uma compra realizada em um estabelecimento comercial utilizando sua conta PIX.

Para isso, o PDV exibe o QRCode que será lido por um aparelho celular que contem o aplicativo da conta PIX.

O serviço oferecido pelo SCOPE é o *QRCode Vendendor* (169) e é ativado através da função:

ScopeCarteiraVirtualEx descrita neste documento.

Pix Saque

Sem ter realizado qualquer compra no estabelecimento comercial, o usuário pode apenas sacar dinheiro de um PDV utilizando sua conta PIX.

Para isso, o PDV exibe o QRCode que será lido por um aparelho celular que contem o aplicativo da conta PIX.

O serviço oferecido pelo SCOPE é o *Saque via QRCode* (176), e é ativado através da função

ScopeSaqueCarteiraVirtual descrita neste documento.

Pix Troco

O usuário pode receber dinheiro como troco associado à um pagamento da compra realizada no estabelecimento comercial utilizando sua conta PIX.

O procedimento realizado pelo usuário é o mesmo do **Pagamento via QRCode**, com a diferença que ele informa um **valor de saque** além do valor da compra realizada.

O serviço e função oferecidos pelo SCOPE para executar o PIX Troco é o mesmo **do Pagamento via QRCode**. Na configuração do produto *QRCode Vendendor* (169), é informado que o PDV permite realizar essa operação juntamente com o pagamento. Para mais detalhes sobre a configuração dessa opção, consulte o Manual de Instalação Scope-NA.

Redes com tratamentos específicos

O objetivo de capítulo é abordar peculiaridades de tratamentos para algumas redes autorizadoras como a coleta de dados no fluxo do SCOPE ou o relacionamento dos produtos da rede e as transações do SCOPE.

VeroBanrisul

Esta é a implementação da nova especificação do Banrisul (o antigo nome Banrisul EMV foi substituído para VEROBANRISUL a partir da versão 3.01.12.xxx) e que trata principalmente a utilização de cartões com chip. Com esta implementação, será possível realizar as seguintes transações do Banrisul no SCOPE:

- Débito à vista, parcelado e pré-datado com o cartão Banricompras;
- Débito à vista e parcelado com o cartão Banco SIM;
- Crédito 1 minuto, que é uma transação de CDC;

- Débito voucher de convênio Refeisul alimentação e refeição;
- Consulta saldo de convênio Refeisul alimentação e refeição;
- Débito voucher de convênio Refeisul combustível
- Crédito com cartão convênio Private Label
- Recarga de Celular BANRISUL VERO (a partir da versão 3.01.12.xxx)

PINPads com suporte a rede

Para a realização das transações com cartões com chip nos PINPads deve-se garantir que o PINPad saiba tratar estes cartões do Banrisul. Os PINPad com a versão da especificação da biblioteca compartilha 1.06^a já suportam, porém aqueles com a versão 1.05g pode não suportar. Nestes casos deverá ser efetuada a transação com a tarja magnética do cartão. Para saber se o PINPad suporta o tratamento do cartão com chip Banrisul, deve-se entrar em contato com o banco Banrisul.

Associação dos produtos e as funções do SCOPE

Abaixo está uma tabela onde são relacionados os produtos do Banrisul e as funções da API do SCOPE que deve ser invocada para realizar a transação.

Produto Banrisul	Função da API do SCOPE
Convênio Banrisul Combustível	ScopeCompraCartaoDebito
Cancelamento	ScopeCancelamento
Pagamento à vista, à prazo e parcelado	ScopeCompraCartaoDebito
Crédito 1 minuto	ScopeCompraCDC
Banco SIM parcelado	ScopeCompraCartaoDebito
Convênio Banrisul Alimentação e Refeição	ScopeCompraCartaoDebito
Convênio Private label	ScopeCompraCartaoCredito
Consulta de saldo de convênio Refeisul alimentação/refeição	ScopeConsultaDebito
Recarga de Celular Banrisul Vero	ScopeRecargaCelular

Transação convênio combustível

Durante o fluxo de coleta desta transação, surgirão coletas específicas para a realização desta com o Banrisul, que são:

- Código de serviço: este é um código de 2 dígitos que informa o tipo de serviço realizado previsto pelo Banrisul. Os possíveis códigos e seus significados são:
 - 01 - Gasolina
 - 02 - Álcool
 - 03 - Diesel
 - 04 - GNV
 - 05 - Combustível Aviação
 - 06 - Óleo
 - 07 - Lavagem
 - 08 - Borracharia
 - 09 - Óleo Marítimo
 - 10 - Graxa
 - 11 - Outros
 - 12 - Gás GLP
 - 13 - Filtro de Ar
 - 14 - Filtro de Óleo

- 15 - Óleo Lubrificante
- 16 - Filtro de Combustível
- 17 - Gasolina Aditivada
- 18 - Serviço de Oficina
- Quantidade de serviço: é um valor numérico de até 12 dígitos que representa a quantidade do serviço prestado. Exemplo: se o código do serviço escolhido foi 03, então a quantidade solicitada é a quantidade de litros de diesel. Se o código foi 08, então é a quantidade de serviço de borracharia executado, que nesse caso poderia ser apenas 1.
- Matrícula: valor alfanumérico de até 11 dígitos.
- Hodômetro: valor numérico que é o hodômetro que está registrando no carro.
- Placa do veículo: campo numérico apenas de até 8 dígitos e deve ser entrada apenas a parte numérica da placa do carro.

Ticket Edenred

A implementação desta trata da especificação das transações com o cartão com chip Ticket Car, na modalidade Benefício e Gestão de Frotas. Transações com cartão são suportadas apenas utilizando o chip do cartão Ticket Car. As transações disponíveis no SCOPE que podem ser realizadas com esta rede são:

- Débito Voucher;
- Estorno Voucher;
- Consulta saldo;
- Atualização de parâmetros (Chip).
- Atualização de preços.

Para trabalhar com Gestão de Frotas é necessária configuração descrita na seção "Parametrização de Gestão de Frota" do "Manual de Instalação e Configuração" do Scope.

O SCOPE suporta a aprovação off-line, efetuada em POS, e uma característica deste cartão é que as transações aprovadas off-line são armazenadas no próprio chip do cartão para ser descarregado numa próxima transação realizada noutro estabelecimento. Portanto o SCOPE suporta o envio de transações realizadas em POS e gravadas no chip. Para a transação de envio destas off-line, é verdade que:

- Pode acontecer em qualquer uma das transações desta rede;
- Não é visível para o operador do caixa ou portador do cartão, ou seja, nenhum dos dois sabem que está enviando tais mensagens;
- Não é impresso comprovante para elas;
- Não é visto em relatórios do SCOPEADM;
- Apenas é possível verificar a existência do envio da offline pelos logs do SCOPE Server.

Não há transações com os cartões Ticket Alimentação e Ticket Restaurante roteadas para esta rede no SCOPE.

PINPads com suporte a rede

Para a realização das transações com cartões com chip nos PINPads deve-se garantir que o PINPad saiba tratar os cartões Ticket Car. Para isto deve-se utilizar PINPads Gertec modelo PPC900 com a versão da especificação da biblioteca compartilhada 1.07^a (ou superior) contendo a chave Masterkey 3.

Associação dos produtos e as funções do SCOPE

Abaixo está uma tabela onde são relacionados os produtos da rede Ticket Edenred e as funções da API do SCOPE que deve ser invocada para realizar a transação.

Produto Ticket Edenred	Função da API do SCOPE
Compra online (Débito Voucher)	ScopeCompraCartaoDebito (vide item <i>Compra com cartão de débito</i>)
Estorno online	ScopeCancelamento (vide item <i>Estornando a transação</i>)
Consulta saldo	ScopeConsultaSaldoDebito (vide item <i>ScopeConsultaSaldoDebito</i>)
Atualização de parâmetros	ScopeAtualizaParametrosChip (vide item <i>ScopeAtualizaParametrosChip</i>)
Atualização de preços	ScopeAtualizaPrecosMercadorias (vide item <i>ScopeAtualizaPrecosMercadorias</i>)

Dados coletados

Abaixo está a relação do que pode ser solicitado em cada transação:

Transação	Coletas
Compra online (Débito Voucher)	<ul style="list-style-type: none"> • Cartão • Dados do ECF (veja mais abaixo) • Senha • Imprimir cupons • Dados Livres (Dados Adicionais)
Estorno online	<ul style="list-style-type: none"> • Número de controle da compra • Cartão • Imprimir cupons
Consulta saldo	<ul style="list-style-type: none"> • Cartão • Senha • Imprimir cupom do cliente
Atualização de parâmetros	<ul style="list-style-type: none"> • Cartão • Senha
Atualização de preços	<ul style="list-style-type: none"> • Nenhuma coleta

Consulta saldo

A função utilizada para esta transação é a ScopeConsultaSaldoDebito que como parâmetro espera receber um valor. No entanto, para os cartões Ticket Car não é utilizado este, portanto, a automação pode passar uma string vazia.

Esta transação não possui cupom da loja, mas apenas do cliente.

Dados do ECF

Na transação de compra online (débito Voucher) é necessário enviar dados do cupom e do ECF. O SCOPE retorna, no modo coleta, então o estado de coleta TC_COLETA_DADOS_ECF (FCBEh) e a automação deve invocar a função ScopeForneceCampo com os seguintes parâmetros:

- TypeField: tipo de dado:
 - SCOPE_DADOS_ECF_BENEFICIO à Fornecer os dados da impressora fiscal. A utilização desta constante deve ser feita na transação de débito, num estado de coleta TC_COLETA_DADOS_ECF. Informações adicionais na seção "Fornecendo informações extras para a transação"

- StructField: dados fornecidos pela aplicação, cujo formato para a constante utilizada na tabela abaixo.

Posição	Formato	Descrição
001-040	ans40	Número de série de fabricação do ECF <i>(alinhado à esquerda com brancos à direita)</i>
041-044	n4	Número do PDV atribuído ao ECF <i>(alinhado à direita com zeros à esquerda)</i>
045-064	n20	Número do documento fiscal <i>(alinhado à esquerda com brancos à direita)</i>
065-072	n8	Data de emissão do documento fiscal no formato AAAAMMDD.

Dados da Lista de Mercadorias Consumidas

Na transação de compra online (débito Voucher) é necessário enviar dados da Lista de Mercadorias Consumidas pela Aplicação. O SCOPE retorna, no modo coleta, o estado de coleta TC_COLETA_LISTA_MERCADORIAS (FCCBh) e a automação deve invocar a função ScopeForneceCampo(char TypeField, void *StructField), com os seguintes parâmetros:

- TypeField: tipo de dado:
 - SCOPE_DADOS_LISTA_MERCADORIAS à Fornecer os dados da Lista de Mercadorias Consumidas. A utilização desta constante deve ser feita na transação de débito, no estado de coleta TC_COLETA_LISTA_MERCADORIAS. Informações adicionais na seção "Fornecendo informações extras para a transação"
- StructField: dados fornecidos pela aplicação, cujo formato para a constante utilizada na tabela abaixo.

Posição	Formato	Descrição
001-004	an4	Header ("LM01") - Identificador do layout da lista
005-005	n1	Quantidade de Casas Decimais para Valor Unitário das mercadorias. Enviado no ID126 (Compreendido entre 0 e 7, inclusive)
006-006	n1	Quantidade de Casas Decimais para Quantidade das mercadorias. Enviado no ID145 (Compreendido entre 0 e 8, inclusive)
007-008	n2	Ramo Atividade "01" (Posto de Combust.) Código do Ramo de Atividade do Checkout. <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
009-013	an5	ID - "SC101" Contém identificador do formato do registro que será passado
'014-016	n3	Qty. Registros - Todos os registros a seguir deverão ser do mesmo formato (SC101): <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
017-106	ans90	Registro SC101 - Conforme descrição do formato.
107-196	ans90	Registro SC101 - Conforme descrição do formato.
...		

A consulta aos códigos das mercadorias está descrita posteriormente no ítem *Código das Mercadorias*

Descrição do formato do registro “SC101”:

O formato deste registro foi baseado no formato do registro ID101 da especificação TicketCar, com o acréscimo da informação da descrição da automação para formatar o registro ID161 da TicketCar.

Posição	Formato	Descrição
001-040	an40	Descrição da Mercadoria da Automação Comercial.
041-045	n5	Código da Mercadoria consumida (Código TicketCar). <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
045-052	n8	Quantidade de Mercadoria consumida. <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
053-061	n9	Valor da Transação para a mercadoria consumida. <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
062-070	n9	Valor unitário ou valor unitário teto da mercadoria consumida (zerado, se transação for OFF-LINE). <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
071-079	n9	Valor da Transação para a mercadoria consumida (com desconto). <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
080-088	n9	Valor do desconto (\$) ou %) Se (%) deverá ser enviado (x 100). Exemplo: 10,15% à 000001015 Se (\$) deverá utilizar expoente de moeda da transação. Exemplo (expoente = 2): \$ 15,66 à 000001566 <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
089-089	n1	Significado do campo “Valor do desconto (moeda ou percentual)”. “0” = Valor (\$) dado como desconto “1” = Valor (%) dado como desconto

Observação: Repetir este bloco tantas quantas forem as mercadorias que os preços foram atualizados.

Coleta de Lista de SKU's

Nas transações em que é necessário o envio de uma lista de SKU foi criada uma estrutura genérica (lista genérica) na qual contempla o máximo de campos necessários para o envio dessa lista. O processo é o mesmo, é necessário enviar dados da Lista de SKU's pela Aplicação. O SCOPE retorna, no modo coleta, o estado de coleta TC_COLETA_LISTA_MERCADORIAS (FCCBh) e a automação deve invocar a função ScopeForneceCampo(char TypeField, void *StructField), com os seguintes parâmetros:

- TypeField: tipo de dado:
 - SCOPE_DADOS_LISTA_MERCADORIAS à Fornecer os dados da Lista de Mercadorias Consumidas. A utilização desta constante deve ser feita na transação de débito, no estado de coleta TC_COLETA_LISTA_MERCADORIAS. Informações adicionais na seção “Fornecendo informações extras para a transação”
- StructField: dados fornecidos pela aplicação, cujo formato para a constante utilizada na tabela abaixo.

O layout da lista de SKU's é o seguinte:

```
#define QTD_MAX_MERCADORIAS_GENERICA 70
```

#define HEADER_LISTA_MERCADORIAS_GENERICA_01

"LG01"

```

struct stMercadoriaGenerica {
    char EAN[14];
    char SKU[14];
    char Descricao[40];
    char QtdMercadoria[13];
(ex: 1000 significa 1,000) */
    char ValorUnitario[9];
    char TipoUnidade[2];
':Litro, 'PK': Empacotado como Pack/Kit) */
    char ValorDesconto[9];
};

struct stListaMercadoriasGenerica{
    char IdLayout[4];           // Identificador do layout da lista ('CV01')
    char Continua[1];          // 'S'= tem proxima lista com mais mercadorias, 'N' = não tem
continuação
com 1)
    char Sequencial[2];        // numero sequencial incrementado a cada lista enviada (sempre começa
                                // com 1)
    char Moeda[3];             // Moeda utilizada (BRL, USD, EUR, etc.)
    char ValorTotal[12];
    char TotalDesconto[12];
    char QtdReg[3];
    struct stMercadoriaGenerica sProduto[QTD_MAX_MERCADORIAS_GENERICA];
};

ScopeForneceCampo(SCOPE_DADOS_LISTA_MERCADORIAS, sListaMercadorias);

```

Observação - Os campos necessários para enviar para SCOPE enviar para a GIVEX são somente. (EAN ou SKU, valor e quantidade).

Dados da Lista de Atualização de Preços de Mercadorias

Na transação de Atualização de Preços de Mercadorias é necessário enviar dados da Lista de Atualização de Preços de Mercadorias. O SCOPE retorna, no modo coleta, o estado de coleta TC_COLETA_LISTA_PRECOS (FCCAh) e a automação deve invocar a função ScopeForneceCampo com os seguintes parâmetros:

- TypeField: tipo de dado:
 - SCOPE_DADOS_LISTA_PRECOS à Fornecer os dados da Lista de Atualização de Preços de Mercadorias. A utilização desta constante deve ser feita na transação de Atualização de Preços de Mercadorias, no estado de coleta TC_COLETA_LISTA_MERCADORIAS. Informações adicionais na seção "Fornecendo informações extras para a transação"
- StructField: dados fornecidos pela aplicação, cujo formato para a constante utilizada na tabela abaixo.

Posição	Formato	Descrição
001-004	an4	"LP01" Identificador do layout da lista
005-009	an5	"ID504" Contém identificador do formato do registro que será passado:

010-012	n3	Qtd. Registros - Todos os registros a seguir deverão ser do mesmo formato (ID504). <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
013-044	ans32	Registro ID504 - Conforme descrição do formato.
045-076	ans32	Registro ID504 - Conforme descrição do formato.

Descrição do formato do registro "ID504":

Posição	Formato	Descrição
001-008	n8	Data da atualização de preço (Formato AAAAMMDD)
009-014	n6	Hora da atualização de preço (Formato HHMMSS)
015-019	n5	Código da mercadoria <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
020-031	n12	Valor do preço unitário <i>(alinhado a direita e preenchidos com zeros à esquerda, caso necessário)</i>
032-032	n1	Número de casas decimais para o valor unitário da mercadoria

Observação: Repetir este bloco tantas quantas forem as mercadorias que os preços foram atualizados.

A consulta aos códigos das mercadorias está descrita posteriormente no ítem Código das Mercadorias.

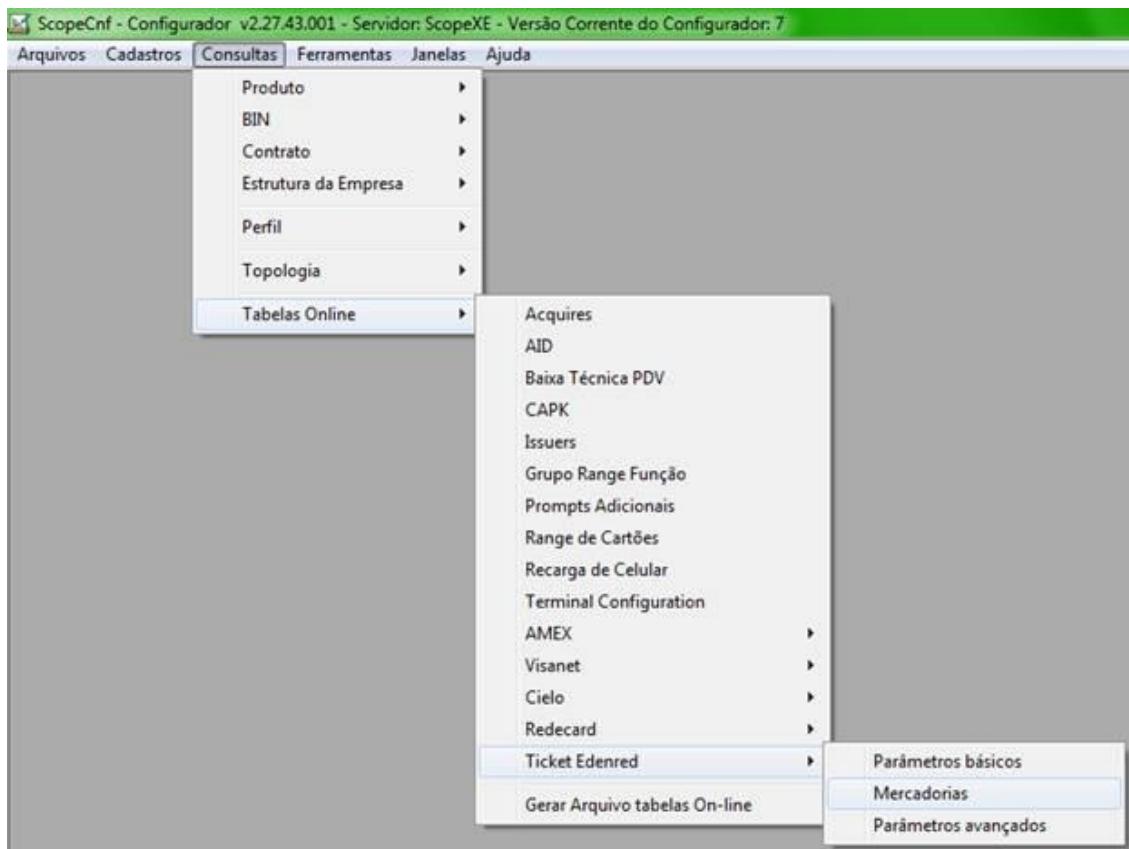
Comprovantes

Abaixo segue os comprovantes existentes para cada transação:

Transação	Coletas		
	Loja	Cliente	Reduzido
Compra online (Débito Voucher)	X	X	
Estorno online	X	X	
Consulta saldo		X	
Atualização de parâmetros (Dados Chip)			
Atualização de preços			

Código das Mercadorias

A Ticket Edenred envia Inicialização de Tabelas com as mercadorias e seus respectivos códigos. É possível consultar as mercadorias parametrizadas pela Ticket Edenred, através do ScopeCNF, acessando o menu exibido a seguir:



Cielo Premia

O CIELO Premia é um programa de premiação, através do qual, o comprador pode receber prêmios no momento da compra quando a TEF é transacionada pela Cielo. Também é conhecido como Plataforma Promocional Online da Cielo, ou, abreviadamente PP Online.

Como é a gestão da campanha?

Toda a gestão da campanha, inclusive sua temporalidade, é controlada pela autorizadora, ou seja, é a Cielo quem irá definir o período de validade da promoção, quais brindes serão oferecidos, regras de sorteio, cartões envolvidos, etc. As configurações são internas da Cielo e/ou Emissores e exigem uma atualização de tabelas (Inicialização). A atualização das tabelas é feita pelo SCOPE de forma online e automática, mediante sinalização da Cielo.

Quais tipos de prêmios o comprador pode receber?

Estão previstos os seguintes tipos de prêmios:

- Brindes, a serem resgatados posteriormente através de transação específica (Resgate de Prêmio); Chamado também de “resgate não monetário”;
- Desconto instantâneo, a ser aplicado no momento da compra, sobre o produto comprado. Chamado também de “resgate monetário”;
- Cupom eletrônico, que é apenas uma notificação de que o cliente está participando de uma campanha promocional. É chamado também de “e-cupom de participação”.

IMPORTANTE: Todos os tipos de prêmios podem ou não ocorrer simultaneamente em uma mesma transação.

Que tipos de transações podem receber os prêmios?

- Crédito;
- Débito;
- Voucher Alimentação;
- Voucher Refeição.

É possível restringir para tratar apenas prêmios não monetários, ou seja, não permitir descontos?

Caso o segmento de atuação ou os estados do Brasil em que sua automação é utilizada possuam restrições fiscais à concessão de descontos monetários, sugerimos consultar a empresa certificadora da CIELO e validar de antemão o tratamento que o tratamento que se pretende fazer na automação será adequado para a certificação.

O que é preciso configurar no SCOPE CNF?

É necessário habilitar o produto abaixo no Perfil de Serviços.

<input checked="" type="checkbox"/> Consulta Resgate de Premios	CIELO	CIELO
---	-------	-------

Bem como a bandeira CIELO na tela de Contrato:

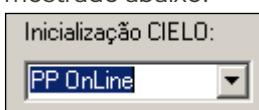
	Bandeira
<input checked="" type="checkbox"/>	CIELO

A Cielo precisa habilitar o Cielo Premia?

Sim, a Inicialização de Tabelas precisa contemplar tais produtos. O Lojista precisa entrar em contato com a CIELO e negociar a implantação do Cielo Premia.

É possível integrar e testar através do SCOPE Autorizador?

Sim. No ScopeAut há uma Inicialização de Tabelas que contempla o Cielo Premia, chamada "PP OnLine", conforme mostrado abaixo:



IMPORTANTE:

1- Antes de iniciar os testes lembre-se de "forçar" uma Inicialização para o estabelecimento em questão para garantir que a carga "PP OnLine" seja carregada no banco de dados pelo ScopeSRV.

2- Os prêmios e cupons gerados pelo ScopeAUT possuem conteúdos fixos, e portanto, não representam efetivamente o que irá ocorrer em produção.

É possível fazer um teste usando o HotKey?

Sim. O HotKeyMP e TstColeta estão preparados, porém deve ser configurada uma nova chave no SCOPE.INI do client, conforme exemplo abaixo (considerando empresa "0001" e filial "0001"):

[00010001]
AutomacaoParticipaPPOnline=S

A Automação Comercial deve se adequar?

Sim. A Automação Comercial precisa informar que está preparada para o Cielo Premia, adequar os fluxos de Crédito e Débito e permitir o Resgate de Prêmio.

O que a Automação Comercial precisa fazer para estar aderente?

Para habilitar o Cielo Premia, ou seja, configurar que está preparada:

Após o ScopeOpen, chamar a função ScopeForneceCampo passando como parâmetros o tipo de dado SCOPE_AUTOMACAO_PARTICIPA_PPONLINE e o conteúdo 'S' ou '1' para indicar que SIM, ou seja, a automação está preparada para participar da plataforma Cielo Premia. Maiores detalhes sobre esta função estão no Manual do Desenvolvedor.

Note que o default (se não executar a função acima) é não habilitado. Neste caso a CIELO não irá retornar nenhum tipo de prêmio e tudo deve funcionar como antes.

Também é possível chamar a mesma função/dado passando 'N' ou '0' para deixar o Cielo Premia desabilitado.

Enquanto não é executado o ScopeClose, o SCOPE mantém a informação se a AC está ou não participando do Cielo Premia válida para as próximas transações.

Também é possível que a AC defina se participa ou não do Cielo Premia a cada transação. Assim, pode-se chamar a ScopeForneceCampo passando SCOPE_AUTOMACAO_PARTICIPA_PPONLINE com SIM ou NÃO antes de cada ScopeCompraCartaoCredito e ScopeCompraCartaoDebito.

Para saber se a Cielo premiou com desconto monetário:

Tratar o novo código de retorno 263 (0x107) retornado pela ScopeStatus para indicar que a transação foi APROVADA COM DESCONTO. Caso o retorno da transação seja 000 (0x000), indica que a transação foi APROVADA (pode ou não ter recebido prêmio não monetário), mas com certeza, não houve desconto. Importante notar que ambos os retornos do ScopeStatus (000 e 263) indicam transação APROVADA.

Não é preciso esperar a finalização da transação para saber se houve ou não o desconto. Normalmente essa informação é necessária ao receber o cupom de TEF, para decidir como finalizar o cupom fiscal. A informação se houve ou não desconto e o valor do desconto podem ser obtidas nos estados de coleta de impressão de cupom.

IMPORTANTE: Quando há desconto na transação, o valor com desconto será exibido no pinpad para o cliente visualizar o valor final. Mensagem "VALOR COM DESCONTO".

Para saber o valor do desconto:

Para obter o valor do desconto, usar a função ScopeObtemCampoExt2, máscara 3, "Informação do desconto do resgate monetário" (0x00000002). Esta informação pode ser solicitada antes da impressão/solicitação do Cupom (ScopeGetCupomEx). Caso o valor retornado seja zero, então não houve desconto.

Como tratar o desconto no Cupom Fiscal:

Veja fluxo sugerido pela Cielo no documento "FLUXO BASICO CIELO PREMIA.pdf".

Como saber se houve um prêmio não monetário (brinde) recebido durante uma compra?

Se houve algum prêmio não monetário (um brinde), esta informação pode ser recuperada através da função ScopeObtemCampoExt2, máscara 3, "Informações sobre a quantidade e e-cupons disponíveis" (0x00000001). A informação segue a seguinte estrutura:

```
typedef struct {  
    char QtdeOcorrenciasECupom[2];
```

```

char DadosResgate[3000];
} stPPResgateNaoMonetario

```

Esta informação pode ser solicitada antes da impressão/solicitação do Cupom (ScopeGetCupomEx).

Os brindes serão acumulados pela Cielo através do número de cartão, portanto, o cupom da promoção é apenas informativo, não necessário para o resgate.

Note que, durante os fluxos de compra (crédito e débito), a informação se ocorreu um prêmio não monetário pode não ser relevante para automação, já que não há nenhum tratamento a fazer.

IMPORTANTE: Para cupom de tamanho zero, descartar o prêmio.

Como fazer um Resgate de Prêmio?

O resgate é feito no SCOPE através da função ScopeMenu. Esta função irá exibir os produtos disponibilizados pela Cielo para o chamado “Menu Outros” e poderá conter outras opções além do resgate de prêmio.

O operador deverá selecionar a opção correspondente, que irá acionar no SCOPE o serviço Consulta Resgate de Prêmio.

O fluxo do Resgate de Prêmio já prevê uma consulta automática à Cielo para buscar os prêmios disponíveis para um determinado cartão, lido no início do fluxo.

Para obter a lista de brindes disponíveis, AC pode tratar o estado de coleta TC_EXIBE_MENU_RESPGATE_PREMIO retornado pela função ScopeStatus.

IMPORTANTE: Para que a opção “Resgate de Prêmio” esteja disponível no Menu, a inicialização de tabelas do estabelecimento em questão deverá estar atualizada de forma a contemplar o Cielo Premia.

Para os resgates não monetários, deve ser seguido basicamente o fluxo:

1. Recuperar do SCOPE os dados dos prêmios e exibir;
2. Coletar o prêmio escolhido e a quantidade;

Para recuperar e exibir os dados dos prêmios:

1. Tratar o estado TC_EXIBE_MENU_RESPGATE_PREMIO;
2. Chamar a função ScopeObtemCampoExt2 pedindo pelos dados de resgate não monetário;
3. Exibir sequencialmente os prêmios de acordo com exemplo abaixo.

Exemplo:

```

char      aux[5000];
...
switch (RC)
{
case TC_EXIBE_MENU_RESPGATE_PREMIO:

    h = ScopeObtemHandle(0);
    if (h > 0xFFFF)
    {

        ScopeObtemCampoExt2(h, 0x00, 0x00, Dados_Resgate_Nao_Monetario, '\0', aux);
        demo_ExibeResgateNaoMonetario(aux);
    }
    ...
}

```

Onde "aux" contém um buffer que contenha espaço necessário para ser armazenado o número de prêmios e os dados relativos aos prêmios. O buffer será preenchido com a seguinte estrutura:

```
typedef struct {
    char QtdeOcorrenciasECupom[2];
    char DadosResgate[3000];
} stPPResgateNaoMonetario
```

Os dois primeiros bytes do buffer correspondem à quantidade de prêmios e os bytes restantes serão os dados dos prêmios. Para cada prêmio, ele será prefixado com 2 bytes que representam o tamanho do texto:

Posição	Formato	Descrição
01-02	N2	Número de E-cupons
03-04	N2	Tamanho do 1º E-cupom
05-x	A(n)	Título do 1º Prêmio, de tamanho n variável definido nos 2 bytes anteriores
...	...	Obs. Repetem-se tamanho e título para os N cupons

IMPORTANTE: A quantidade de prêmios pode ser igual a zero e corresponde que não há premiação.

Exemplo:

```
void demo_ExibeResgateNaoMonetario (char *aux)
{
    stPPResgateNaoMonetario sDadosResgateNaoMonetario;
    short i, nTamanhoEcupom = 0, j = 0, nQuantidade = 0;
    char sTamanhoEcupom[2+1], sNomeEcupom[99+1], sQuantidade[2+1];

    memset (&sDadosResgateNaoMonetario, 0, sizeof(sDadosResgateNaoMonetario));
    memset (&sNomeEcupom, 0, sizeof(sNomeEcupom));
    memcpy(sQuantidade, aux, 2);
    sQuantidade[2] = 0;
    strcpy(sDadosResgateNaoMonetario.DadosResgate, &aux[2]);

    nQuantidade = atoi(sQuantidade);

    if (nQuantidade > 0)
    {
        printf ("\n\nPLATAFORMA PROMOCIONAL:\n\n");
    }
    else
    {
        printf ("\n\nNAO HA PREMIACAO, CONSULTE REGULAMENTO");
        return;
    }

    for(i = 0; i < nQuantidade && i < 20; i++)
    {
        memcpy(sTamanhoEcupom, &sDadosResgateNaoMonetario.DadosResgate[j], 2);
        sTamanhoEcupom[2] = 0;
        nTamanhoEcupom = atoi(sTamanhoEcupom);
        j += 2;
        memcpy(sNomeEcupom, &sDadosResgateNaoMonetario.DadosResgate[j], nTamanhoEcupom);

        sNomeEcupom[nTamanhoEcupom] = 0;

        j += nTamanhoEcupom;

        printf ("%02d - %s", i + 1, sNomeEcupom);
    }
}
```

```

    printf ("\n");
}

printf ("\n\n");

return;
}

```

Após a exibição dos prêmios, a automação pode fazer:

A coleta do índice/número do prêmio escolhido pelo cliente (TC_NRO_ESGATE_PREMIO 0xFCA4)

A coleta de quantidade a ser resgatada (TC_CONFIRMA_OPCAO_ESGATE_PREMIO 0xFCA2)

IMPORTANTE: A quantidade de prêmios não é repassada para a Automação Comercial, somente é exibida no display.

Existe uma forma de tratar os itens do Menu Dinâmico pela aplicação de AC?

Sim. Os itens exibidos pela função ScopeMenu podem ser manipulados pela AC. Para isso existem as funções:

- ScopeMenuRecuperaltens;
- ScopeMenuSelecionaltem.

Através da função ScopeMenuRecuperaltens a AC pode tratar o estado TC_EXIBE_MENU para obter os itens de forma a exibi-los todos de uma só vez, ou então, exibir apenas os que interesse para a Automação Comercial.

A seleção também pode ser controlada através da ScopeMenuSelecionaltem. A Automação Comercial pode usar essa função após a seleção do operador de caixa, ou ainda, selecionar um item de forma automática, sem exibir a lista ao operador.

Existe um código exemplo?

Sim. Um código em C de exemplo (TstColeta) acompanha este documento.

Qual é a versão mínima do SCOPE que contempla Cielo Premia?

A versão mínima é 2.27.23.02 (server e client), embora o recomendável seja o uso da versão mais recente.

Cielo Auto

Esta transação é efetuada com cartões do tipo voucher que permitem o pagamento do abastecimento de combustível e outros serviços prestados aos veículos automotivos em estabelecimentos como postos de gasolina.

Na especificação CIELO – Release 2014, corresponde ao Fluxo 32 – “Transação Auto”. Transações são suportadas apenas utilizando o chip do cartão. A bandeira que irá tratar esta transação será:

- 278 - CIELO AUTO

As transações disponíveis no SCOPE que podem ser realizadas com este cartão são:

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

- Débito Voucher;
- Estorno Voucher;
- Consulta saldo;
- Atualização de parâmetros (Chip).

Lista de produtos e/ou serviços principais Cielo Auto

Esta lista poderá conter os seguintes ramos principais:

- COMBUSTÍVEL
- OUTROS PRODUTOS E SERVIÇOS
- ESTACIONAMENTO

Esta lista será exibida de acordo com as informações enviadas na Carga de Tabelas da CIELO e nas informações habilitadas no chip do cartão. O SCOPE disponibiliza os itens do Menu para a aplicação de automação comercial para a melhor exibição dessas informações. Esta lista é fornecida durante o fluxo de coleta através do estado abaixo:

TC_COLETA_RAMO_PRINC_CIELO_AUTO	0xFCD9	Coleta Ramo Principal Cielo Auto
---------------------------------	--------	----------------------------------

Num fluxo de Compra com Cartao de Debito, recebendo o código de coleta 64729 (TC_COLETA_RAMO_PRINC_CIELO_AUTO), a aplicação de PDV pode realizar o mesmo tratamento do estado TC_EXIBE_MENU (que já existe hoje), ou seja, opcionalmente chamar a função [ScopeMenuRecuperaltens](#) do SCOPE para receber os itens do Menu a serem exibidos e em seguida chamar a [ScopeMenuSelecionalitem](#) para indicar ao SCOPE qual item foi selecionado.

Desta forma, o PDV poderá, por exemplo:

- Exibir todos os itens na mesma tela (hoje, pelo fluxo do SCOPE Client, é apresentado um item por vez);
- Excluir algum item da lista que eventualmente não deva ser exibido ao operador;
- Alterar a descrição de algum item para melhor entendimento do operador;
- Fazer a seleção automática de um determinado item sem apresentá-lo ao operador;

Caso o aplicativo de automação comercial opte por não tratar este estado de coleta, o SCOPE exibirá os itens na forma de menu rotativo.

Vale ressaltar que este menu somente será exibido se existir mais que um ramo principal a ser selecionado. Caso exista apenas um item, o SCOPE assume-o automaticamente sem exibir o menu.

Lista de produtos e/ou serviços secundários Cielo Auto

Cada ramo principal poderá conter sua própria lista de produtos e/ou serviços secundários que atuam como **mercadorias** para automação comercial. Esta lista será exibida de acordo com as informações enviadas na Carga de Tabelas da CIELO e nas informações habilitadas no chip do cartão. Esta lista poderá conter as seguintes mercadorias:

COMBUSTÍVEL
Gasolina comum
Etanol

Diesel	
Gasolina Aditivada	
Gasolina Premium	
Etanol Aditivado	
Diesel S50	
Diesel Aditivado	
GNV (m3)	
Biodiesel	
OUTROS PRODUTOS E SERVIÇOS	
Lava-Rápido	
Aditivos e Lubrificantes	
Filtros	
Borracharia	
Manutenção Geral	
Equipamentos elétricos	
ESTACIONAMENTO	
Estacionamento	

Esta lista é fornecida durante o fluxo de coleta através do estado abaixo:

TC_COLETA_COD_MERC_CIELO_AUTO	0xFCDA	Coleta Código de Mercadoria Cielo Auto
-------------------------------	--------	--

Caso exista apenas uma mercadoria habilitada na Carga de Tabelas ou no cartão, o SCOPE assume automaticamente essa mercadoria.

No fluxo de Compra com Cartão de Débito, recebendo o código de coleta 64730 (TC_COLETA_COD_MERC_CIELO_AUTO), a aplicação de PDV deverá obter a lista de mercadorias disponíveis, através da função ScopeRecuperaBufTabela().

Protótipo

```
LONG EXPORT ScopeRecuperaBufTabela (BYTE _TipoTabela,
char *Qtd,
char *ListaBuffer,
WORD TamLista)
```

Descrição

- Esta rotina servirá para disponibilizar para a aplicação de automação comercial, uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada.
- Ela servirá para facilitar futuras implementações que atendam a mesma finalidade.
- O parâmetro _TipoTabela indicará o formato dos dados que serão fornecidos.

Parâmetros

[out]	String [2+1]	Qtd	Retorna a quantidade de elementos da lista.
[out]	String	ListaBuffer	Retorna os registros da lista.
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaBuffer"

Retorno

Ver [tabela de código de retorno](#).

Parâmetro TipoTabela:

ID layout	Usar estrutura	Descrição
BUF_TAB_MERCADORIA_CIELOAUTO (ID 3)	stREG_MERCADORIA_CIELOAUTO	Contém os campos, conforme a transação AUTO da especificação CIELO .

Parâmetro ListaBuffer:

O SCOPE está preparado para tratar MAX_MERCADORIA_CIELOAUTO=72 mercadorias da CIELO AUTO.

A lista é formada por registros stREG_MERCADORIA_CIELOAUTO com os seguintes campos:

Posição	Formato	Descrição
01 a 02	char	Código do Ramo Principal.
03 a 04	char	Código da Mercadoria.
05 a 45	String	Descricao da Mercadoria.

Exemplo

```

int i = 0;
char qtd [2+1] = {0};
LONG lRet;
stREG_MERCADORIA_CIELOAUTO lista [MAX_MERCADORIA_CIELOAUTO] = {0};

lRet = ScopeRecuperaBufTabela (BUF_TAB_MERCADORIA_CIELOAUTO,
                                qtd, (char*) lista,
                                sizeof (lista));

if (!lRet == RCS_SUCESSO)
{
    // Sucesso no recebimento da lista de mercadorias
    for (i = 0; i < atoi(qtd); i++)
    {
        // exibe os membros das estruturas que estarão preenchidos
        // lista[i].CodRamoPrinc,
        // lista[i].CodMercadoria,
        // lista[i].Descricao
    }
}
else
    // Erro ...

```

Informações Adicionais Cielo Auto

Estas informações serão exibidas para coleta de acordo com a parametrização da Carga de Tabelas da CIELO e de acordo com as informações habilitadas no chip do cartão.

Informações Adicionais

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Captura da placa
Hodômetro
Horímetro
Código do Motorista

A aplicação de PDV não necessita realizar qualquer tratamento para a coleta desses campos pois o SCOPE exibe de forma dinâmica, tanto a mensagem quanto os dados coletados, dependendo do cartão utilizado.

SAVS

Esta seção trata da especificação das transações com a rede SAVS (Sistema Autorizador de Vouchers e Serviços).

As transações implementadas no SCOPE que podem ser realizadas com esta rede são:

- Autorização de Voucher;
- Estorno de Autorização Voucher;

Associação dos produtos e as funções do SCOPE

Abaixo está uma tabela onde são relacionados os produtos da rede SAVS e as funções da API do SCOPE que deve ser invocada para realizar a transação.

Produto SAVS	Função da API do SCOPE
Autorização de Voucher	ScopeServicosGenericos
Estorno	ScopeCancelamento

Autorização de Vouchers e Serviços

Para a transação de Autorização de Vouchers e Serviços a automação deverá chamar a função [ScopeServicosGenericos](#), passando como parâmetro uma string contendo o código de Serviço e uma string vazia.

Durante transação de Autorização de Voucher, como a rede SAVS trabalha com Carga de Tabelas, algumas coletas podem ocorrer, ou não, de acordo com a configuração enviada ao SCOPE na Inicialização de Tabelas.

O SCOPE seleciona internamente o fluxo de coleta, de acordo com a configuração enviada pela rede Autorizadora. Para facilitar o entendimento, seguem as coletas afetadas pela parametrização da Carga de Tabelas e a descrição dos Estados de Coleta que serão desviados pelo SCOPE:

- Parametrização de Fornecedor

De acordo com a parametrização do Fornecedor selecionado, as coletas solicitadas estão descritas na tabela a seguir:

- Coleta de Fornecedor:

Parâmetro da Carga		Coleta
Quantidade de Fornecedores Disponíveis no Segmento Selecionado	1	<ul style="list-style-type: none"> • Seleciona automaticamente o único Fornecedor disponível e pula para: • Coleta de Produto - TC_COLETA_PRODUTO_SAV (0xFCD6)
	>1	<ul style="list-style-type: none"> • Coleta de Fornecedor - Estado TC_COLETA_FORNECEDOR_SAV (0xFCD5)

- Coleta de códigos, valor, quantidade e Cliente Preferencial:

Parâmetro da Carga	Coleta
Tipo de coleta de dados do Voucher	<u>Código do Produto</u> - Estado TC_COLETA_PRODUTO_SAV (0xFCD6) <u>Valor</u> - Estado TC_COLETA_VALOR (0xFC34) ou TC_INFO_AGU_CONF_OP(0xFCFF) <u>Quantidade</u> - Estado TC_COLETA_QUANTIDADE (0xFCD7)
	<u>Código EAN</u> - Estado TC_COLETA_PRODUTO_SAV (0xFCD6) <u>Valor</u> - Estado TC_COLETA_VALOR (0xFC34) ou TC_INFO_AGU_CONF_OP(0xFCFF)
	<u>Código do Produto</u> - Estado TC_COLETA_PRODUTO_SAV (0xFCD6) <u>Valor</u> - Estado TC_COLETA_VALOR (0xFC34) ou TC_INFO_AGU_CONF_OP(0xFCFF) <u>Número do Voucher</u> - Estado TC_COLETA_NRO_VOUCHER (0xFCA5)
	<u>Código EAN</u> - Estado TC_COLETA_PRODUTO_SAV (0xFCD6) <u>Valor</u> - Estado TC_COLETA_VALOR (0xFC34) ou TC_INFO_AGU_CONF_OP(0xFCFF) <u>Número do Voucher</u> - Estado TC_COLETA_NRO_VOUCHER (0xFCA5)
Cliente preferencial	<u>Não solicita Coleta de Cliente Preferencial</u>
	<u>Coleta: Cliente Preferencial? (Sim/Não)</u> - Estado TC_COLETA_CLIENTE_PREFERENCIAL (0xFCD8)

Para cada Segmento selecionado no Estado de Coleta TC_COLETA_SEGMENTO_SAV (0xFCD4) podem estar disponíveis alguns Fornecedores. A lista de Fornecedores associada ao segmento selecionado, em passo anterior do fluxo de coleta, está descrita posteriormente neste documento na seção "Lista de Fornecedores".

- Parametrização de Produtos

De acordo com a parametrização do Produto (Produto, Serviço ou Código EAN) selecionado podem ser habilitados:

- Coleta de dados dinâmicos (Numérico, alfabético, alfanumérico, menu de opções, Trilha de Cartão, Data, etc.) - Estado TC_COLETA_DADOS_ADICIONAIS (0xFC7C)
- Exibição de valor sugerido - Estado TC_COLETA_VALOR (0xFC34)

- Confirmação do Valor Assumido por Parametrização – Estado TC_INFO_AGU_CONF_OP (0xFCFF)

Para cada Fornecedor selecionado no Estado de Coleta TC_COLETA_FORNECEDOR_SAV (0xFCD5) podem estar disponíveis alguns Produtos. A lista de Produtos associada ao Fornecedor selecionado, em passo anterior do fluxo de coleta, está descrita posteriormente neste documento na seção “Lista de Produtos”.

Estorno de Autorização de Vouchers e Serviços

Para estorno da transação de Autorização de Vouchers e Serviços a automação deverá chamar a função ScopeCancelamento passando como parâmetro o valor total da transação original.

Lista de Fornecedores

A lista de fornecedores pode ser retornada para a automação comercial no novo layout, como descrito a seguir:

Num fluxo de Autorização de Serviços Voucher, recebendo o código de coleta 64725 (TC_COLETA_FORNECEDOR_SAV), a aplicação deverá obter a lista de fornecedores disponíveis, através da função ScopeRecuperaBufTabela () .

Protótipo

```
LONG EXPORT ScopeRecuperaBufTabela (BYTE _TipoTabela,
char *Qtd,
char *ListaBuffer,
WORD TamLista)
```

Descrição

- Esta rotina servirá para disponibilizar para a aplicação de automação comercial, uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada.
- Ela servirá para facilitar futuras implementações que atendam a mesma finalidade.
- O parâmetro _TipoTabela indicará o formato dos dados que serão fornecidos.

Parâmetros

[out]	String	Qtd	Retorna a quantidade de elementos da lista.
[out]	String	ListaBuffer	Retorna os registros da lista.
[in]	WORD	TamLista	Tamanho, em bytes, do campo “ListaBuffer”

Retorno

Ver [tabela de código de retorno](#).

Parâmetro TipoTabela:

ID layout	Usar estrutura	Descrição
BUF_TAB_FORNECEDORES_SA VS (ID 1)	stREGISTRO_FORNECEDORE S_SAVS	Fornecedores SAVS. Contém os campos, conforme a especificação do Autorizador Voucher.

Parâmetro ListaBuffer:

A lista é formada por registros stREGISTRO_FORNECEDORES_SAVS com os seguintes campos:

Posição	Formato	Descrição
---------	---------	-----------

01 a 04	String	Código do Fornecedor.
05 a 25	String	Nome do Fornecedor.
26 a 27	String	Código do Segmento de Mercado.
27 a 27	String	Quantidade máxima de produtos diferentes.

Exemplo

```

char sQtdFornec[3] = {0};
LONG lRet;
stREGISTRO_FORNECEDORES_SAVS lstFornecedores [20] = {0};

lRet = ScopeRecuperaBufTabela (BUF_TAB_FORNECEDORES_SAVS,
                                sQtdFornec, (char*) lstFornecedores,
                                sizeof (lstFornecedores));

if (lRet == RCS_SUCESSO)
{
    /* Sucesso no recebimento da lista de fornecedores,
       os membros das estruturas estarão preenchidos */
}
else
    // Erro ...

```

Lista de Produtos

A lista de produtos pode ser retornada para a automação comercial no novo layout, como descrito a seguir:

Num fluxo de Autorização de Serviços Voucher, recebendo o código de coleta

TC_COLETA_PRODUTO_SAV (0xFCD6), a aplicação deverá obter a lista de produtos disponíveis, através da função ScopeRecuperaBufTabela.

Protótipo

```

LONG EXPORT ScopeRecuperaBufTabela (BYTE _TipoTabela,
char *Qtd,
char *ListaBuffer,
WORD TamLista)

```

Descrição

- Esta rotina servirá para disponibilizar para a aplicação de automação comercial, uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada.
- Ela servirá para facilitar futuras implementações que atendam a mesma finalidade.
- O parâmetro _TipoTabela indicará o formato dos dados que serão fornecidos.

Parâmetros

[out]	String	Qtd	Retorna a quantidade de elementos da lista.
[out]	String	ListaBuffer	Retorna os registros da lista.
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaBuffer"

Retorno

Ver [tabela de código de retorno](#)

ID layout	Usar estrutura	Descrição
BUF_TAB_PRODUTOS_SAVS (ID 2)	stREGISTRO_PRODUTOS_SA VS	Produtos SAVS. Contém os campos: código, descrição e valores ,

		conforme a especificação do Autorizador Voucher.
--	--	--

ParâmetroListaBuffer:

A lista é formada por registros stREGISTRO_PRODUTOS_SAVS com os seguintes campos:

Posição	Formato	Descrição
01 a 13	String	Código do Produto, Código do Serviço ou Código EAN.
14 a 34	String	Descrição.
35 a 47	String	Valor mínimo
47 a 59	String	Valor máximo
59 a 71	String	Valor sugerido

Exemplo

```

char sQtdProd[3] = {0};
LONG lRet;
stREGISTRO_PRODUTOS_SAVS lstProdutos [50] = {0};

lRet = ScopeRecuperaBufTabela (BUF_TAB_PRODUTOS_SAVS, sQtdProd, (char*)
lstProdutos, sizeof (lstProdutos));

if (lRet == RCS_SUCESSO)
{
    /* Sucesso no recebimento da lista de produtos,
       os membros das estruturas estarão preenchidos */
}
else
    // Erro ...

```

Cielo – Transações Sem Contato (Contactless)

São transações realizadas através de leitura sem contato no PINPad. Atualmente a Cielo 4.1 R2014, também conhecida como Cielo Release 2014, permite realizar esse tipo de transação.

Atualmente as bandeiras Visa, Visa Electron, Mastercard, Maestro e Amex são as únicas certificadas a operar através de leitura sem contato.

Requisitos

Para que esse tipo de transação seja aceito, o estabelecimento precisa estar trabalhando com Cielo 4.1 R2014. O PINPad precisa ser no mínimo 1.08^a com módulo Contactless instalado e firmware atualizado. Em caso de dúvida, consulte o fabricante do PINPad para saber se seu PINPad está ou não preparado para aceitar Contactless.

Na inicialização de tabelas do estabelecimento em questão, a Cielo precisa habilitar e enviar na carga AIDs com Contactless habilitado.

O valor da transação precisa estar dentro da faixa permitida para Contactless, dado que também é configurado através da inicialização de tabelas. A carga no PINPad precisa estar atualizada com pelo menos um AID do grupo da transação em questão (Crédito ou Débito) com Contactless habilitado.

Funcionamento

A habilitação ou não da leitora Contactless é feita de forma automática, conforme tópicos descritos acima. Não é necessária nenhuma configuração no SCOPE.

Quando ocorre erro na leitura sem contato, dependendo do tipo de erro ocorrido, a próxima tentativa pode desabilitar a leitura sem contato, de forma a evitar que o PINPad leia novamente por aproximação quando na verdade, o portador do cartão deveria inserir o chip ou passar a tarja.

A leitura sem contato também é desabilitada automaticamente pelo PINPad quando o valor da transação for acima do limite permitido para Contactless.

Existem dois tipos de cartão sem contato: "Chip EMV Sem Contato" e "Tarja Sem Contato".

Transações de "Tarja Sem Contato" são equivalentes a uma transação "Magnética".

Transações de "Chip EMV Sem Contato" são equivalentes a uma transação de "Chip EMV".

A leitura "Sem Contato" é feita de forma única, ou seja, somente é necessário aproximar o cartão do PINPad uma única vez, mesmo quando se tratar do tipo "Chip EMV Sem Contato".

Valecard

A implementação desta rede trata da especificação das transações com o cartão magnético Valecard, nas modalidades Débito e Crédito, com ou sem Gestão de Frota. As transações disponíveis no SCOPE que podem ser realizadas com esta rede são:

- Débito;
- Estorno Débito;
- Débito Voucher;
- Estorno Voucher;
- Crédito;
- Estorno Crédito;

Para trabalhar com Gestão de Frotas é necessária a utilização de cartão parametrizado pela Valecard em Carga de Tabelas. Além disto, é necessário que tanto a transação de Débito quanto a transação de Crédito sejam efetuadas à vista.

Associação dos produtos e as funções do SCOPE

Abaixo está uma tabela onde são relacionados os produtos da rede Valecard e as funções da API do SCOPE que deve ser invocada para realizar a transação.

Produto Ticket Edenred	Função da API do SCOPE
Compra online (Débito)	ScopeCompraCartaoDebito (vide item Compra com cartão de débito)
Estorno online (Débito)	ScopeCancelamento

	(vide item <i>Estornando a transação</i>)
Compra online (Débito Voucher)	ScopeCompraCartaoDebito (vide item <i>Compra com cartão de débito</i>)
Estorno online (Débito Voucher)	ScopeCancelamento (vide item <i>Estornando a transação</i>)
Compra online (Crédito)	ScopeCompraCartaoCredito (vide item <i>Compra com cartão de débito</i>)
Estorno online (Crédito)	ScopeCancelamento (vide item <i>Estornando a transação</i>)

Dados coletados

Abaixo está a relação do que pode ser solicitado em cada transação:

Transação	Coletas
Compra online (Débito/ Débito Voucher)	Cartão Senha Parcelado ou a vista Quantidade de parcelas Número do Cupom Fiscal KM (Hodômetro) Matricula Lista de Produtos (ver seção <i>Dados da Lista de Produtos Consumidos</i>) Imprimir cupons
Compra online (Crédito)	Cartão Parcelado ou a vista Parcelado Lojista ou Adm Quantidade de parcelas Data de vencimento do cartão Número do Cupom Fiscal KM (Hodômetro) Matricula Lista de Produtos (ver seção <i>Dados da Lista de Produtos Consumidos</i>) Imprimir cupons
Estorno online	Número de controle da compra Cartão Imprimir cupons

Dados da Lista de Produtos Consumidos

Na transação de compra online (Débito Voucher e Crédito a Vista) é necessário enviar dados da Lista de Produtos Consumidos pela Aplicação. O SCOPE retorna, no modo coleta, o estado de coleta TC_COLETA_LISTA_MERCADORIAS (FCCBh) e a automação deve invocar a função

ScopeForneceCampo(char TypeField, void *StructField), com os seguintes parâmetros:

- TypeField: tipo de dado:
 - SCOPE_DADOS_LISTA_MERCADORIAS à Fornecer os dados da Lista de Produtos Consumidos. A utilização desta constante deve ser feita nas transações de débito e crédito, no estado de coleta TC_COLETA_LISTA_MERCADORIAS. Informações adicionais na seção “Fornecendo informações extras para a transação”
- StructField: dados fornecidos pela aplicação, cujo formato para a constante utilizada na tabela abaixo.

Posição	Formato	Descrição
001-004	an4	Header ("LP01") - Identificador do layout da lista
005-007	n3	Quantidade de registros de Produtos a seguir
008-033	ans26	Registro de Produto - Conforme descrição do formato.
034-059	ans26	Registro de Produto - Conforme descrição do formato.
...		

Descrição do formato do registro de Produto:

O formato deste registro foi baseado no formato do registro de Produto da especificação Valecard.

Posição	Formato	Descrição
001-002	n2	Código do Produto consumido
003-014	n12	Quantidade de Produtos Formato 15,50 = "000000001550"
015-026	n12	Valor Total para o produto consumido Formato 96,40 = "000000009640"

SGF

Esta seção trata da especificação das transações com a rede SGF (Sistema Gerenciador de Fidelidade).

As transações implementadas no SCOPE que podem ser realizadas com esta rede são:

- Consulta Pontos
- Acúmulo por Valor
- Acúmulo por Pontos
- Resgate de Produtos
- Resgate Monetizado (por valor)
- Cancelamento de Operação Fidelidade

IMPORTANTE: Atualmente somente o fornecedor MULTIPLUS tem operações Fidelidade habilitadas para a rede SGF.

Associação dos produtos e as funções do SCOPE

Segue uma tabela onde são relacionados os produtos da rede SGF e as funções da API do SCOPE que deve ser invocada para realizar a transação.

Produto SGF	Função da API do SCOPE
Consulta de Pontos/Pontuação	ScopeFidelidade
Acúmulo por Valor	ScopeFidelidade
Acúmulo por Pontos	ScopeFidelidade
Resgate de Produtos	ScopeFidelidade
Resgate Monetizado (por valor)	ScopeFidelidade
Cancelamento Operação Fidelidade	ScopeCancelamento

Antes de iniciar uma Operação Fidelidade

A operação de Fidelidade necessidade de dados sobre o estabelecimento e sobre o operador em questão.

Estas informações devem sempre ser fornecidas pela automação comercial antes de qualquer Operação Fidelidade. A automação deve chamar a função [ScopeForneceCampo\(\)](#) passando Dados da Aplicação ([SCOPE_DADOS_APPLIC](#)):

Exemplo

```

...
TIPO_DADOS_APPLIC DadosAplicacao;
DadosAplicacao.TipoTerminal = 'P';
strcpy(DadosAplicacao.Usuario, "0000123456");

ScopeForneceCampo(SCOPE_DADOS_APPLIC, &DadosAplicacao);

// Chamar a operação Fidelidade desejada
...

```

OBSERVAÇÃO: O tamanho máximo do número/código do operador (campo 'Usuario' na estrutura TIPO_DADOS_APPLIC) é definido pelo Fornecedor na carga de tabelas. Mais detalhes sobre o tamanho máximo a ser informado deve ser verificado com a Multiplus ou outro Fornecedor em questão.

Consulta de Pontos

Para a transação de Consulta de Pontos a automação deverá chamar a função ScopeFidelidade, passando como parâmetro o serviço **S_CONSULTA_PONTOS** e a bandeira representando o Fornecedor Fidelidade.

Durante a transação, como a rede SGF trabalha com Carga de Tabelas, algumas coletas podem ocorrer ou não de acordo com a configuração enviada ao SCOPE na Inicialização de Tabelas.

O SCOPE seleciona internamente o fluxo de coleta, de acordo com a configuração enviada pela rede Autorizadora. Para facilitar o entendimento, seguem as coletas afetadas pela parametrização da Carga de Tabelas e a descrição dos Estados de Coleta que serão desviados pelo SCOPE:

- Parametrização de Fornecedor

De acordo com a parametrização do Fornecedor selecionado, as coletas solicitadas estão descritas na tabela a seguir:

- Coleta de Fornecedor:

Parâmetro da Carga		Coleta
Quantidade de Fornecedores Disponíveis		
	1	Seleciona automaticamente o único Fornecedor disponível e pula para: Coleta da identificação
	>1	<u>Coleta de Fornecedor</u> - Estado TC_COLETA_FORNECEDOR_SGF (0xFCD5)

OBSERVAÇÃO: Se Bandeira/Fornecedor já foi fornecido na chamada da função, ele não será coletado.

IMPORTANTE: A lista de Fornecedores enviada pela autorizadora deve ter equivalência no Scope. Somente os fornecedores (bandeiras) habilitados serão considerados nas transações.

- Coleta da identificação:

Parâmetro da Carga		Coleta
Tipos de identificação		
	1	<ul style="list-style-type: none"> Seleciona automaticamente o único tipo de coleta de identificação disponível e pula para a Coleta da Identificação habilitada (um dos 3 tipos: CPF, cartão ou código de barras)
	>1	<ul style="list-style-type: none"> <u>Identificação CPF</u> - Estado TC_CPF (0xFC66)
	>1	<ul style="list-style-type: none"> <u>Identificação Cartão</u> - Estado TC_CARTAO (0xFC00)
	>1	<ul style="list-style-type: none"> <u>Identificação Código Barras</u> - Estado TC_COLETA_COD_BARRAS (0xFC48)

- Parametrização de Operações.

De acordo com a parametrização da Operação selecionada podem ser habilitados:

- Coleta de dados dinâmicos (Numérico, alfabético, alfanumérico, menu de opções, Data, etc.)
- Estado TC_COLETA_DADOS_ADICIONAIS (0xFC7C)
- Exibição de valor sugerido - Estado TC_COLETA_VALOR (0xFC34)
- Confirmação do Valor Assumido por Parametrização – Estado TC_INFO_AGU_CONF_OP (0xFCFF)
- Senha do Participante – Estado TC_SENHA (0xFC11)
- Exibição de menu - Estado TC_EXIBE_MENU (0xFC87)

Cancelamento de Operações Fidelidade

Para cancelamento da transação de Operações Fidelidade a automação deverá chamar a função ScopeCancelamento() passando como parâmetro o valor total da transação original.

Esta operação só pode ser efetuada no estabelecimento que realizou a Operação Fidelidade sendo cancelada.

IMPORTANTE: A automação comercial deve comandar o cancelamento de uma operação Fidelidade associada a uma compra sempre que esta original for cancelada. Isso evita, por exemplo, que um participante acumule pontos em uma compra que foi efetivamente cancelada.

Lista de Fornecedores

A lista de fornecedores pode ser retornada para a automação comercial no novo layout, como descrito a seguir.

Num fluxo de Operação Fidelidade, recebendo o código de coleta TC_COLETA_FORNECEDOR_SGF (0xFCDF), a aplicação deverá obter a lista de fornecedores disponíveis, através da função ScopeRecuperaBufTabela () .

Protótipo

```
LONG EXPORT ScopeRecuperaBufTabela (BYTE TipoTabela,
char *Qtd,
char *ListaBuffer,
WORD TamLista)
```

Descrição

- Esta rotina servirá para disponibilizar para a aplicação de automação comercial, uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada.
- Ela servirá para facilitar futuras implementações que atendam a mesma finalidade.
- O parâmetro _TipoTabela indicará o formato dos dados que serão fornecidos.

Parâmetros

[out]	String	Qtd	Retorna a quantidade de elementos da lista.
[out]	String	ListaBuffer	Retorna os registros da lista.
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaBuffer"

Retorno

Ver [tabela de código de retorno](#).

Parâmetro TipoTabela:

ID Layout / Tabela	Usar estrutura	Descrição
BUF_TAB_FORNECEDORES_SF F (ID 4)	stREGISTRO_FORNECEDORES_SGF	Fornecedores Fidelidade. Contém os campos, conforme a especificação do SGF.

Parâmetro ListaBuffer:

A lista é formada por registros stREGISTRO_FORNECEDORES_SGF com os seguintes campos:

Posição	Formato	Descrição
01 a 03	String	Código da Bandeira – relativo ao Fornecedor Fidelidade
04 a 34	String	Nome do Fornecedor.

Exemplo

```
| char sQtdFornec[3] = {0};
```

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

```

LONG IRet;
stREGISTRO_FORNECEDORES_SGF lstFornecedores [20] = {0};

IRet = ScopeRecuperaBufTabela (BUF_TAB_FORNECEDORES_SGF,
                               sQtdFornec, (char*) lstFornecedores,
                               sizeof (lstFornecedores));

if (IRet == RCS_SUCESSO)
{
    /* Sucesso no recebimento da lista de fornecedores,
       os membros das estruturas estarão preenchidos */
}
else
    // Erro ...

```

Lista de Produtos

A lista de produtos pode ser retornada para a automação comercial no novo layout, como descrito a seguir

Num fluxo de Resgate de Produtos, recebendo o código de coleta TC_RESP_CONS_ESG_PROD (0xFCDD), a aplicação deverá obter a lista de produtos disponíveis, através da função ScopeRecuperaBufTabela.

Protótipo

```

LONG EXPORT ScopeRecuperaBufTabela (BYTE TipoTabela,
                                     char *Qtd,
                                     char *ListaBuffer,
                                     WORD TamLista)

```

Descrição

- Esta rotina servirá para disponibilizar para a aplicação de automação comercial, uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada.
- Ela servirá para facilitar futuras implementações que atendam a mesma finalidade.
- O parâmetro TipoTabela indicará o formato dos dados que serão fornecidos.

Parâmetros

[out]	String	Qtd	Retorna a quantidade de elementos da lista.
[out]	String	ListaBuffer	Retorna os registros da lista.
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaBuffer"

Retorno

Ver [tabela de código de retorno](#).

ID Layout / Tabela	Usar estrutura	Descrição
BUF_TAB_LISTA_PROD_SGF (ID 6)	stREGISTRO_LISTA_PROD_SGF	Produtos SGF. Contém os campos: código e descrição, conforme a especificação do Autorizador SGF.

Parâmetro ListaBuffer:

A lista é formada por registros stREGISTRO_LISTA_PROD_SGF com os seguintes campos:

Posição	Formato	Descrição
01 a 04	String	Código do Produto
05 a 45	String	Descrição

Exemplo

```

char sQtdProd[3] = {0};
LONG IRet;
stREGISTRO_LISTA_PROD_SGF lstProdutos [20] = {0};

IRet = ScopeRecuperaBufTabela (BUF_TAB_LISTA_PROD_SGF, sQtdProd, (char*)
lstProdutos, sizeof (lstProdutos));

if (IRet == RCS_SUCESSO)
{
    /* Sucesso no recebimento da lista de produtos,
       os membros das estruturas estarão preenchidos */
}
else
    // Erro ...

```

Como processar a lista de Produtos?

Esta lista deve sempre ser processada quando presente para que o cliente ou operador possa gerenciar os itens do resgate. A ideia é que, ao receber o estado TC_RESP_CONS_RESG_PROD, o PDV possa chamar a função ScopeRecuperaBufTabela do SCOPE para receber os itens a serem exibidos e em seguida chamar a ScopeResumeParam para indicar ao SCOPE qual item foi selecionado.

Desta forma, o PDV poderá, por exemplo:

- Exibir todos os itens na mesma tela;
- Excluir algum item da lista que eventualmente não deva ser exibido ao operador;
- Alterar a descrição de algum item para melhor entendimento do operador;
- Fazer a seleção automática de um determinado item sem apresentá-lo ao operador;

OBSERVAÇÃO: O índice do item selecionado deve ser equivalente ao item da tabela fornecida pelo Scope, independente de ter sido exibido para o usuário ou não. Por exemplo, se o menu tem 10 itens, alguns deles não foram exibidos e o operador seleciona o último item do menu, o item a ser selecionado continua sendo o índice do item 10.

Lista de Faixas de Pontos

A lista de faixas de pontos pode ser retornada para a automação comercial no novo layout, como descrito a seguir.

Num fluxo de Resgate Monetizado, recebendo o código de coleta TC_RESP_CONS_RESG_PONTOS (0xFCDE), a aplicação deverá obter a lista de faixas disponíveis, através da função ScopeRecuperaBufTabela.

Protótipo

```

LONG EXPORT ScopeRecuperaBufTabela (BYTE TipoTabela,
char *Qtd,
char *ListaBuffer,
WORD TamLista)

```

Descrição

- Esta rotina servirá para disponibilizar para a aplicação de automação comercial, uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada.
- Ela servirá para facilitar futuras implementações que atendam a mesma finalidade.
- O parâmetro TipoTabela indicará o formato dos dados que serão fornecidos.

Parâmetros

[out]	String	Qtd	Retorna a quantidade de elementos da lista.
[out]	String	ListaBuffer	Retorna os registros da lista.
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaBuffer"

Retorno

Ver [tabela de código de retorno](#).

ID Layout / Tabela	Usar estrutura	Descrição
BUF_TAB_FAIXA_PONTOS_SGF (ID 7)	stREGISTRO_FAIXAS_PONTOS_SGF	Faixas de Pontos SGF. Contém os campos: código e descrição, conforme a especificação do Autorizador SGF.

Parâmetro ListaBuffer:

A lista é formada por registros stREGISTRO_FAIXAS_PONTOS_SGF com os seguintes campos:

Posição	Formato	Descrição
01 a 04	String	Código do item do Menu
05 a 45	String	Descrição do item

Exemplo

```

char sQtdItens[3] = {0};
LONG IRet;
stREGISTRO_FAIXAS_PONTOS_SGF lstFaixas [20] = {0};

IRet = ScopeRecuperaBufTabela (BUF_TAB_FAIXA_PONTOS_SGF, sQtdItens,
(char*) lstFaixas, sizeof (lstFaixas));

if (IRet == RCS_SUCESSO)
{
    /* Sucesso no recebimento da lista de faixas,
    os membros das estruturas estarão preenchidos */
}
else
    // Erro ...

```

Esta lista deve sempre ser processada quando presente para que o cliente ou operador possa gerenciar os itens do resgate. A ideia é que, ao receber o estado TC_RESP_CONS_RESG_PONTOS, o PDV possa chamar a função ScopeRecuperaBufTabela do SCOPE para receber os itens a serem exibidos e em seguida chamar a função ScopeResumeParam para indicar ao SCOPE qual item foi selecionado. Veja exemplo similar e detalhes no item [Como processar a lista de Produtos](#).

Dados Retornados em Operação Fidelidade

Após a Operação Fidelidade ser completada com sucesso, ela pode retornar dados adicionais que permitem identificar se o pagamento foi concluído por meio do programa de Fidelidade, além de outros dados para impressão de cupom não fiscal se for o caso, ou armazenamento do status da operação.

Essas informações adicionais sobre a transação podem ser obtidas através da função ObtemCampoExt2() parametrizada com o valor para a máscara 3: *Dados_Retornados_Operacao_Fidelidade* (definido em scopeapi.h).

Exemplo

```

...
// constante Dados_Retornados_Operacao_Fidelidade definida em scopeapi.h
#define Dados_Retornados_Operacao_Fidelidade 0x00020000
...

long h;
char aux[999];
handle = ScopeObtemHandle(0);
if(handle > 0xFFFF)
{
    // obtendo os dados da transação Fidelidade
    memset(aux, '\0', sizeof(aux));
    ScopeObtemCampoExt2(h, 0x00, 0x00, Dados_Retornados_Operacao_Fidelidade, ':', aux);
}
else
{
    // erro! não conseguiu o handle
}
...

```

Onde "aux" contém um buffer que contenha espaço necessário para serem armazenados os dados relativos às Operações Fidelidade. O buffer será preenchido com a seguinte estrutura:

```

typedef struct {
    char BitMap[8];
    char sPontos[10];
    char sSaldoPontos[12];
    char sCodPreCad[30];
    char sPontosResgate[10];
    char sNomePrograma[30];
    char sNumCupomSort[20];
    char cConcedeBonus;
    char sTipoBonus[2];
    char cUtilizaBonusParc;
    char sPercDescConc[6];
    char sValorDescConc[12];
    char sldBrinde[15];
    char sTipoCartao[2];
    char sPercDescUtil[6];
}

```

IMPORTANTE: Na estrutura, todos os campos são tratados como alfanuméricos, porém alguns podem ser tratados de forma específica (por ex. somente numérico). Veja na tabela abaixo o formato do conteúdo quando retornado. Para campos onde o conteúdo é menor que o tamanho do campo, ele é completado com espaços, por exemplo [123]

De acordo com a transação realizada, alguns dados serão preenchidos e não é necessário que todos os dados desta estrutura sejam preenchidos para o Fornecedor escolhido (por ex. Multiplus) ou para a operação em questão, portanto, nesta estrutura o primeiro campo *Bitmap* determina quais informações estão sendo preenchidas.

Formato	Campo	Descrição
Byte(8)	BitMap	Determina quais campos da estrutura foram preenchidos. Representação do valor em Hexa
N10	sPontos	Número de Pontos Acumulados (concedidos)
N12	sSaldoPontos	Saldo do participante em pontos
N30	sCodPreCad	Reservado Uso Futuro - Código de pré-cadastramento - corresponde ao código do primeiro acesso do participante se houve pré-cadastro do mesmo no fornecedor Fidelidade em tratamento.
N10	sPontosResgate	Número de Pontos Resgatados/Transferidos
AN30	sNomePrograma	Reservado Uso Futuro - Nome do programa do parceiro destino em operação de Transferência de Pontos.
N20	sNumCupomSort	Reservado Uso Futuro - Número do cupom de sorteio.
A1	cConcedeBonus	Reservado Uso Futuro - Concede Bônus (S/N) - indica se o cartão fidelidade em tratamento tem ou não direito a bônus.
N2	sTipoBonus	Tipo do Bônus - contém o tipo de bônus concedido Se concedido bônus, Tipo 01 – (<i>uso futuro</i>) Tipo 02 – Desconto fixo Tipo 03 – (<i>uso futuro</i>)
A1	cUtilizaBonusParc	Reservado Uso Futuro - Permite utilização parcial do Bônus (S/N)
N6	sPercDescConc	Reservado Uso Futuro - Porcentual de desconto concedido, com duas casas decimais.
N12	sValorDescConc	Valor do desconto concedido, com duas casas decimais.
AN15	sldBrinde	Reservado Uso Futuro - Identificação do Brinde concedido (Consulta Bônus – bônus tipo “Brinde”)
N2	sTipoCartao	Reservado Uso Futuro - Tipo de cartão
N6	sPercDescUtil	Reservado Uso Futuro - Porcentual do desconto a utilizar (Resgate Bônus), com duas casas decimais.

IMPORTANTE: Todos os campos marcados como "**Reservado uso Futuro**" não são tratados pelo fornecedor **Multiplus**

Como processar o campo Bitmap e a estrutura de dados?

O primeiro campo da estrutura representa um mapa de bits, ou seja, cada bit (da esquerda para a direita) do valor preenchido representa um dos campos da estrutura na ordem apresentada anteriormente.

Como a estrutura stREGISTRO_DADOS_OPERACOES_FIDELIDADE_SGF atualmente prevê somente 14 campos, este bitmap pode ter somente os 4 primeiros bytes preenchidos (4 bytes em hexa = 16 bits). Os demais bytes estarão zerados (valor = '0') e reservados para uso futuro.

Exemplo Geral

O valor do campo Bitmap está em hexadecimal e representado com caracteres ASCII, portanto deve ser convertido para binário para depois ser validado qual bit está ligado. Cada byte do campo representa 4 bits (valor de '0' a 'F') e de acordo com esta representação, o maior valor possível em hexadecimal será: 0xFFFFFFFF (32 bits).

Veja mais detalhes nos exemplos a seguir.

Exemplo 1

Se uma operação deve retornar os dados de Pontos (1º campo) e de Saldo de Pontos (2º campo), esta estrutura estará com o byte Bitmap[0] preenchido com o valor 1100 ('C') e todos os demais (Bitmap[1] a Bitmap[7]) preenchidos com o valor '0'.

Bitmap[0] = 1100 ('C') indica que os 2 primeiros bits estão ligados que correspondem aos 2 primeiros campos da estrutura

Exemplo 2

Se uma operação deve retornar os dados de Pontos (1º campo) e de Número de Pontos Resgatados/Transferidos (4º campo), Tipo do Bônus (8º campo) e Valor Desconto Concedido (11º campo) esta estrutura em binário seria

"10010001001000000000000000000000", e pode ser representada em hexadecimal como: "91200000", portanto:

Bitmap[0] preenchido com o valor 1001 ('9') - indica que o 1º, 4º bits estão ligados

Bitmap[1] preenchido com o valor 0001 ('1') - indica que o 8º bit está ligado

Bitmap[2] preenchido com o valor 0010 ('2') - indica que o 11º bit está ligado

E todos os demais (Bitmap[3] a Bitmap[7]) preenchidos com o valor '0'.

OBSERVAÇÃO: Lembrando que o campo Bitmap tem 8 bytes e que pode representar até 32 campos, porém a estrutura atualmente contém somente os 14 campos descritos.

IMPORTANTE: Os Campos que estiverem desabilitados de acordo com a representação do Bitmap, contêm valor indefinido e não devem ser processados nem utilizados de nenhuma forma. Campo desabilitado indica que o campo não foi retornada para operação em questão.

REDE V06.xx

Transações Sem Contato (Contactless)

São transações realizadas através de leitura sem contato no PINPad. Atualmente a Rede V6.01, permite realizar esse tipo de transação.

Atualmente somente algumas bandeiras são certificadas a operar através de leitura sem contato.

©2025 Conexão Tech. All rights reserved. Conexão Tech – Confidential
Use and Disclose Solely Pursuant to Company Instructions

Requisitos

Para que esse tipo de transação seja aceito, o estabelecimento precisa estar trabalhando com Rede V6.01. O PINPad precisa ser no mínimo 1.08^a com módulo Contactless instalado e firmware atualizado. Em caso de dúvida, consulte o fabricante do PINPad para saber se seu PINPad está ou não preparado para aceitar Contactless.

Na inicialização de tabelas do estabelecimento em questão, a Cielo precisa habilitar e enviar na carga AIDs com Contactless habilitado.

O valor da transação precisa estar dentro da faixa permitida para Contactless, dado que também é configurado através da inicialização de tabelas. A carga no PINPad precisa estar atualizada com pelo menos um AID do grupo da transação em questão (Crédito ou Débito) com Contactless habilitado.

Funcionamento

Mesmo funcionamento descrito no tópico da [Cielo Contactless](#).

Pagamento de Contas - CORBAN

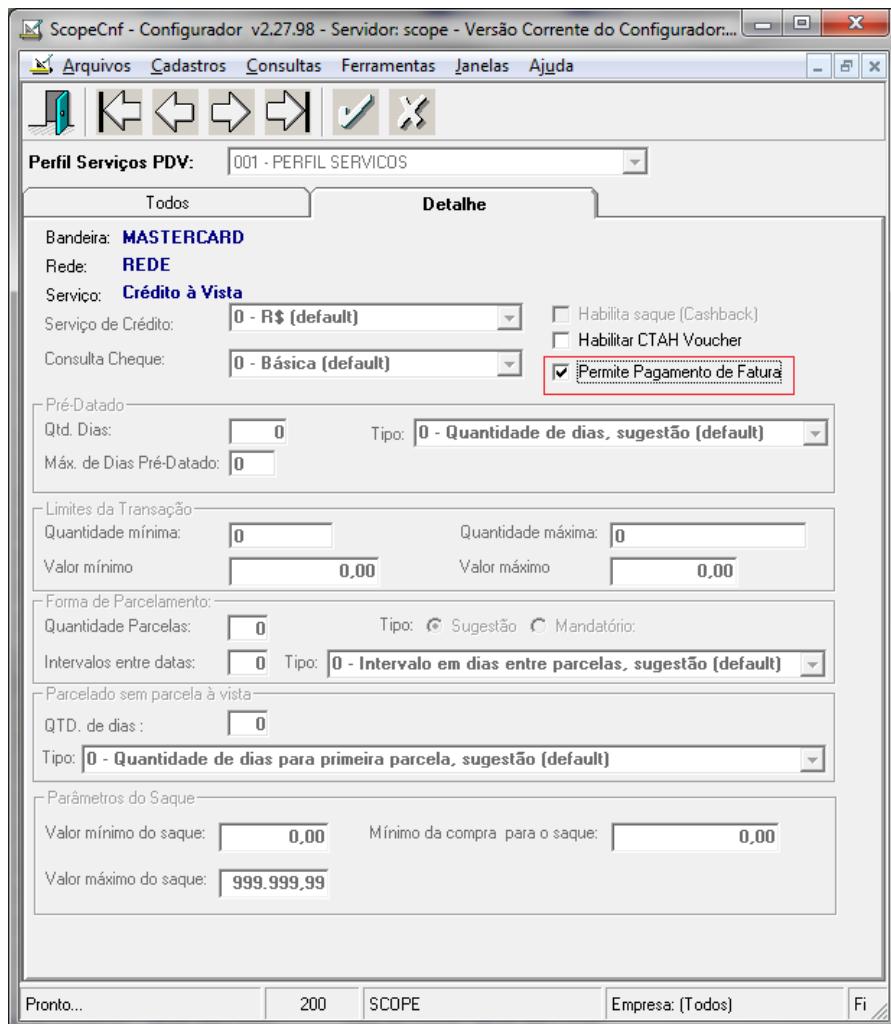
Pagamento de Fatura

São transações de pagamento de Fatura pela Rede V6.01 e estão disponibilizados dentro do fluxo de crédito (ScopeCompraCartaoCredito) e débito (ScopeCompraCartaoDebito). Não há um serviço específico para este tipo de transação.

O Pagamento de fatura não pode ser realizado através da função ScopePagamento.

Esta opção do serviço além de estar previsto na carga de tabelas da REDE para o estabelecimento, também deve ser habilitado no SCOPECNF.

A configuração deve ser feita pelo SCOPECNF – Perfil – Serviços – Parâmetros:



Na figura anterior, vemos que a bandeira MASTERCARD, da REDE, para o serviço de Crédito à Vista está com “Permite Pagamento de Fatura” habilitada. Isso quer dizer que neste serviço a coleta da Pagamento de Fatura será solicitada se na carga de tabelas esta opção para esta bandeira vier também habilitado.

Coleta de Dados do Pagamento de Fatura

A coleta dos dados da Fatura é condicionada às configurações explicadas anteriormente, e também de que a autorizadora respondendo pela transação também tenha a funcionalidade de Pagamento de Fatura.

A transação deve ser iniciada normalmente como uma transação de crédito [ScopeCompraCartaoCredito\(\)](#). Caso o cartão utilizado seja da REDE, no fluxo de crédito pode ser solicitada a coleta de Pagamento de Fatura.

Se a automação decide responder automaticamente a coleta, ela pode responder aos estados extras:

Códigos Retorno		Significado
Hexadecimal	Decimal	
0xFCC1	64705	Coleta opção de pagamento de fatura em transações de débito ou crédito
0xFCDF	64735	Coleta de Código para Identificação pagamento de Fatura (Exemplos de código: CPF, Código de Barras e Cartão)

Para informar dados automaticamente:

1. Tratar o estado TC_DECIDE_PGTO_CARNE e TC_COLETA CPF_CARTAO_OU_CODBARRAS;
2. Chamar a função ScopeResumeParam() informando os dados que a automação já tem;

Exemplo:

```
...
switch (RC)
{
    case TC_DECIDE_PGTO_CARNE:

        // passa se transação é de pagamento de fatura ou não
        ScopeResumeParam(cod_coleta, dados, modo_entrada, acao);
        break;

    case TC_COLETA_CPF_CARTAO_OU_CODBARRAS:

        // passa para Scope os dados do pagamento
        ScopeResumeParam(cod_coleta, dados, modo_entrada, acao);
        break;

    ...
}
```

Dados do Pagamento de Fatura

Para a REDE é necessário informar para qual documento será relacionado o pagamento. Este documento deve ter relação com a Fatura sendo paga, ou seja, uma das seguintes opções:

- CPF do cliente,
- Número do Cartão
- Código de Barras da fatura sendo paga;

Somente se fornecido um destes documentos, a transação será realmente de pagamento de Fatura. A validação dos dados será feita pela autorizadora.

Getnet – Transações Sem Contato (Contactless)

São transações realizadas através de leitura sem contato no PINPad. Atualmente a Getnet LAC V2.94 permite realizar esse tipo de transação.

Atualmente as bandeiras Visa, Visa Electron, Mastercard e Maestro são as únicas certificadas a operar através de leitura sem contato.

Requisitos

Para que esse tipo de transação seja aceito, o estabelecimento precisa estar trabalhando com Getnet LAC V2.94. O PINPad precisa ser no mínimo 1.08^a com módulo Contactless instalado e firmware atualizado. Em caso de dúvida, consulte o fabricante do PINPad para saber se seu PINPad está ou não preparado para aceitar Contactless.

Na inicialização de tabelas do estabelecimento em questão, a Getnet precisa habilitar e enviar na carga AIDs com Contactless habilitado.

O valor da transação precisa estar dentro da faixa permitida para Contactless, dado que também é configurado através da inicialização de tabelas. A carga no PINPad precisa estar atualizada com pelo menos um AID do grupo da transação em questão (Crédito ou Débito) com Contactless habilitado.

Funcionamento

A habilitação ou não da leitora Contactless é feita de forma automática, conforme tópicos descritos acima. Não é necessária nenhuma configuração no SCOPE.

Quando ocorre erro na leitura sem contato, dependendo do tipo de erro ocorrido, a próxima tentativa pode desabilitar a leitura sem contato, de forma a evitar que o PINPad leia novamente por aproximação quando na verdade, o portador do cartão deveria inserir o chip ou passar a tarja.

A leitura sem contato também é desabilitada automaticamente pelo PINPad quando o valor da transação for acima do limite permitido para Contactless.

Existem dois tipos de cartão sem contato: "Chip EMV Sem Contato" e "Tarja Sem Contato".

Transações de "Tarja Sem Contato" são equivalentes a uma transação "Magnética".

Transações de "Chip EMV Sem Contato" são equivalentes a uma transação de "Chip EMV".

A leitura "Sem Contato" é feita de forma única, ou seja, somente é necessário aproximar o cartão do PINPad uma única vez, mesmo quando se tratar do tipo "Chip EMV Sem Contato".

Redes Padrão Lojista Frota

A partir da especificação lojista Scope Padrão ISO8583 - Versão 3.02.003 é possível efetuar transação Voucher com Controle de Frota em que dados adicionais são coletados pela automação comercial para serem enviados para a rede autorizadora.

Atualmente esta modalidade de transação está disponível para as redes, bandeiras e serviços:

Rede	Especificação Lojista
071-ECXCARD	Especificação 1-V3.02(SCOPE Padrão ISO8583 - Versão 3.02.006)
086-VALESHOP	Especificação 1-V08 EMV(SCOPE Padrão ISO8583 EMV-Versão 3.02.003)

Rede	Bandeira	Serviço
071-ECXCARD	416-ECXCARD COMB	50-Débito Voucher
086-VALESHOP	384-VALESHOP COMB	50-Débito Voucher

Para trabalhar com Gestão de Frotas é necessária a utilização de cartão parametrizado pela Rede Lojista em Carga de Tabelas. Além disto, é necessário que a transação de Débito seja efetuada à vista.

Dados coletados

Abaixo está a relação do que pode ser solicitado em cada transação:

Transação	Coletas
Compra online (Débito Voucher)	<ul style="list-style-type: none"> • Cartão • Senha • Lista de Produtos (ver seção <i>Dados da Lista de Produtos Consumidos</i>) • Placa do Veículo • Hodômetro/Horímetro • Matrícula • Imprimir cupons
Estorno online	<ul style="list-style-type: none"> • Número de controle da compra • Cartão • Imprimir cupons

Dados da Lista de Produtos Consumidos

Na transação Débito Voucher é necessário enviar dados da Lista de Produtos Consumidos pela Aplicação. O SCOPE retorna, no modo coleta, o estado de coleta TC_COLETA_LISTA_MERCADORIAS (FCCBh) e a automação deve invocar a função

ScopeForneceCampo(char TypeField, void *StructField), com os seguintes parâmetros:

- TypeField: tipo de dado:
 - SCOPE_DADOS_LISTA_MERCADORIAS à Fornecer os dados da Lista de Produtos Consumidos. A utilização desta constante deve ser feita nas transações de débito, no estado de coleta TC_COLETA_LISTA_MERCADORIAS. Informações adicionais na seção "Fornecendo informações extras para a transação".
- StructField: dados fornecidos pela aplicação, cujo formato para a constante utilizada na tabela abaixo.

Posição	Formato	Descrição
001-002	an2	Identificação do tipo de dado - "VF" (Voucher Frota)
003-004	n2	Versão deste layout - "01"

005-006	n2	Quantidade de Produtos a seguir. (Máximo 38) A quantidade máxima de produtos permitida pode ser verificada por ScopeObtemCampo. Os detalhes estão descritos na seção <i>Quantidade máxima de produtos por venda</i> , a seguir.
007-031	ans25	Registro de Produto - Conforme descrição do formato.
032-056	ans25	Registro de Produto - Conforme descrição do formato.
...		

Descrição do formato do Registro de Produto:

O formato deste registro foi baseado no formato do registro Voucher Frota, definido na especificação Lojista, enviado como Dados Adicionais de Transação.

Posição	Formato	Descrição
001-003	n3	Código do Produto consumido
004-013	n10	Quantidade de Produtos Formato 15,50 = "0000001550"
014-025	n12	Valor Total para o produto consumido Formato 96,40 = "000000009640"

São permitidos até 38 produtos por transação.

A relação dos códigos dos produtos possíveis é de responsabilidade da rede autorizadora e de conhecimento da automação comercial.

A lista de produtos atrelados à transação é fornecida pela automação comercial. O SCOPE valida apenas o formato, mas não tem condições de verificar se os dados fornecidos, incluindo os códigos de produtos, respectivas quantidades e valores estão corretos.

Quantidade máxima de Produtos por venda:

Durante a transação Débito Voucher é possível verificar a quantidade máxima de produtos permitida para a venda. Essa informação pode ser obtida através da função ObtemCampoExt2() parametrizada com o valor para a máscara 3: *Max_Mercadorias_Lista* (definido em scopeapi.h).

Exemplo

```
long h;
char aux[999];
// Solicita um handle para uma transacao que ainda esta em andamento
handle = ScopeObtemHandle(9);
if(handle > 0xFFFF)
{
    // obtendo os dados da transação Fidelidade
    memset(aux, '\0', sizeof(aux));
```

```
    ScopeObtemCampoExt2(h, 0x00, 0x00, Max_Mercadorias_Lista, ':', aux);
}
else
{
    // erro! não conseguiu o handle
}
...
...
```

Onde "aux" contém um buffer que contenha no mínimo 2 caracteres, pois a quantidade máxima é indicada por 2 dígitos.

Somente para bandeira parametrizada para aceitar controle de frota e, caso a lista de produtos ainda não tenha sido fornecida ao Scope durante a transação, será permitido o envio da lista de produtos. Caso contrário o Scope sempre retornará "00" para o obtemcampo Max_Mercadorias_Lista

VeroBanrisul – Transações Sem Contato (Contactless)

São transações realizadas através de leitura sem contato no PINPad. A versão VeroBanrisul Release 2017 (antigo Banrisul EMV) permite realizar esse tipo de transação.

Atualmente as bandeiras Visa, Visa Electron, Mastercard e Maestro são as únicas certificadas a operar através de leitura sem contato.

Requisitos

Para que esse tipo de transação seja aceito, o estabelecimento precisa estar trabalhando com VeroBanrisul R17. O PINPad precisa ser no mínimo 1.08^a com módulo Contactless instalado e firmware atualizado. Em caso de dúvida, consulte o fabricante do PINPad para saber se seu PINPad está ou não preparado para aceitar Contactless.

Na inicialização de tabelas do estabelecimento em questão, a Rede VeroBanrisul precisa habilitar e enviar na carga AIDs com Contactless habilitado.

O valor da transação precisa estar dentro da faixa permitida para Contactless, dado que também é configurado através da inicialização de tabelas. A carga no PINPad precisa estar atualizada com pelo menos um AID do grupo da transação em questão (Crédito ou Débito) com Contactless habilitado.

Funcionamento

A habilitação ou não da leitora Contactless é feita de forma automática, conforme tópicos descritos acima. Não é necessária nenhuma configuração no SCOPE.

Quando ocorre erro na leitura sem contato, dependendo do tipo de erro ocorrido, a próxima tentativa pode desabilitar a leitura sem contato, de forma a evitar que o PINPad leia novamente por aproximação quando na verdade, o portador do cartão deveria inserir o chip ou passar a tarja.

A leitura sem contato também é desabilitada automaticamente pelo PINPad quando o valor da transação for acima do limite permitido para Contactless.

Existem dois tipos de cartão sem contato: "Chip EMV Sem Contato" e "Tarja Sem Contato".

Transações de "Tarja Sem Contato" são equivalentes a uma transação "Magnética".

Transações de "Chip EMV Sem Contato" são equivalentes a uma transação de "Chip EMV".

A leitura "Sem Contato" é feita de forma única, ou seja, somente é necessário aproximar o cartão do PINPad uma única vez, mesmo quando se tratar do tipo "Chip EMV Sem Contato".

Ticket Log – Transações Frota

São transações de Crédito cuja bandeira é B_TICKET_LOG_FROTA = 414. Essas transações podem solicitar "Perguntas Dinâmicas" durante o fluxo de coleta. A Ticket Log usa a nomenclatura "Captura Configurável" para referenciar às transações com "Coleta Dinâmica". Transações que não possuem "Coleta Dinâmica" são chamadas de "Cartão Valor" pela Ticket Log.

Coletas Dinâmicas

Para poder trabalhar com essas transações, a Automação Comercial (AC) precisa necessariamente estar adaptada às regras de integração com o SCOPE Client descritas abaixo:

Tratar o novo estado de coleta abaixo:

- TC_COLETA_DADO_ESPECIAL = 0xFCF4

Ao receber esse estado, a automação deverá chamar a função ScopeGetParamExt ao invés da ScopeGetParam.

Essa função retorna novos tipos de coleta. O tipo TM_SELECAO exige um tratamento adicional. Para receber as opções de seleção use a função ScopeMenuRecuperaltens e para indicar ao SCOPE qual opção foi selecionada use a função ScopeMenuSelecionaltem. Para a função ScopeMenuRecuperaltens há um novo tipo de tabela de dados: MNU_TAB_TIPO_SELECAO_ESPECIAL = 5.

Permissão de Descontos

Os cartões Ticket Log Frota permitem que sejam concedidos descontos. Portanto, a AC precisa ser preparada para tratar eventuais descontos concedidos pela Ticket Log na resposta de uma transação Frota. No SCOPE foi criada uma forma da AC informar se está preparada ou não. Caso não seja usada essa configuração, o SCOPE assume por padrão "não permite desconto". Para essa configuração deve-se usar a função ScopeForneceCampo, com o tipo SCOPE_AUTOMACAO_PERMITE_DESCONTO = 38.

As opções são:

- "0" = Não permite desconto (default);
- "1" = Permite desconto apenas no valor total;
- "2" = Permite desconto "por item";

Essa configuração pode efetuada a qualquer momento após a execução com sucesso da ScopeOpen e ficará válida até a execução do ScopeClose.

Assim, se a automação trabalha de uma única maneira, basta executá-la uma única vez.

Caso a automação queira alterar a configuração, basta executá-la novamente, o que permite trabalhar com configurações diferentes dependendo da transação. Por exemplo, para uma dada transação, o desconto só é possível no "total". Para outra, pode ser "por item";

Tratamento de Desconto Concedido

A AC deverá estar preparada para tratar eventuais descontos concedidos e obter o valor do desconto (total ou por item), além do valor atualizado da transação. Tais informações estão disponíveis através dos campos abaixo, a serem obtidos através de uma das funções do tipo ScopeObtemCampo.

O valor total do desconto poderá ser obtido pelo campo abaixo:

Masc3	Valor_Desconto = 0x00080000
-------	-----------------------------

Caso haja desconto por item, a relação pode ser obtida pelo campo abaixo:

Masc4	TicketLog_Descontos_Por_Item = 0x00000100
-------	---

Esse campo pode ser retornado nas respostas de compra (bit 54 da 0210) e caso preenchido, contém a lista de descontos concedidos pelo autorizador, item a item. Cada item dessa lista possui o seguinte formato:

Formato: CCCCDDDDDDDDDDDDDD

CCCC – Código do produto;

DDDDDDDDDDDD – Valor do desconto (10 inteiros e 2 decimais).

Para obter o valor atualizado, pode-se usar o campo abaixo:

Masc1	Valor_transacao = 0x00000002
-------	------------------------------

Produtos Comprados

Durante o fluxo de uma transação Frota, o SCOPE poderá solicitar a relação de "produtos comprados". O estado de coleta para este caso é:

- TC_COLETA_LISTA_MERCADORIAS = 0xFCFB

A AC deverá fornecer a relação de produtos comprados através da função ScopeForneceCampo com tipo de dado SCOPE_DADOS_LISTA_MERCADORIAS = 27, que usa o formato abaixo:

```
#define QTD_MAX_PRODUTOS_TICKETLOG 37

typedef struct {
    charCodigo[4]; // código do produto
    charValor[12]; // valor unitário do produto (10 inteiros e 2 decimais)
    charQtd[10]; // quantidade do produto (7 inteiros e 3 decimais)
```

```

}stTypeProdutoTicketLog, *LPTYPEPRODUTO TICKETLOG;

typedef struct {
    char IDHeader[4]; // HEADER_LISTA_PRODUTOS_TICKETLOG = "TL01"
    char QtdProdutos[2]; // qtd de elementos (produtos) na lista a seguir
    stTypeProdutoTicketLog sProduto[QTD_MAX_PRODUTOS_TICKETLOG];
}stTypeListaProdutosTicketLog, *LPTYPELISTAPRODUTOSTICKETLOG;

```

A Ticket Log permite trabalhar com uma lista fixa de produtos, que deve ser previamente cadastrada no sistema de frente de loja. Neste caso, a responsabilidade de fazer o “de-para” dos respectivos códigos de produto é da AC. A forma de trabalho deve ser negociada pelo lojista diretamente com a Ticket Log.

Opcionalmente, caso necessário, a AC pode usar a função ScopeObtemProdutosFrota do SCOPE para receber a lista de produtos disponíveis, que foi recebida pelo SCOPE através da transação de Inicialização de Tabelas Online.

Saldo Disponível

Caso retornado um saldo disponível, este pode ser obtido pelo campo abaixo:

Masc1	Saldo_Disponivel = 0x80000000
-------	-------------------------------

O Saldo Disponível eventualmente retornado pela Ticket Log pode estar discriminado em Reais (R\$) ou em Litros (L). Essa informação pode ser obtida pelo campo abaixo:

Masc4	Saldo_Disponivel_Listros_ou_Reais = 0x000000100
-------	---

Esse campo tem o seguinte formato:

Conteúdo	Significado
"0"	Saldo Disponível em Reais (R\$) – Default
"1"	Saldo Disponível em Litros (L)

DICA: Utilize o arquivo fonte demo.c do módulo TstColeta.exe como exemplo e referência de integração.

FIC

Para solicitar senha tem que seguir os seguintes passos:

- Fechar o Scope Server
- Habilitar contrato da 103-Rede Espec 5 - L06.01
- Habilitar contrato da 075-FIC Espec 1 - EMV
- Subir o Scope Server

- Se a carga de tabelas da Rede não for efetuada automaticamente, force pela interface.
- Suspender a 103-REDE
- Tente efetuar transação com o cartão EMV da FIC
- Se aparecer a coleta Cod. Plano de Pagamento: [NN], digite "3" ou "4"
- Deve coletar senha

GetNetLAC v2.99

Split de Pagamento

"Split de Pagamento" é um termo usado para se referir aos dados de uma venda realizada através do conceito de "Marketplace". Possui os dados do "Seller" (loja hospedeira), "Subsellers" (lojistas que vendem produtos através de uma loja hospedeira) e "Produtos" (ou itens), ou seja, a identificação dos produtos e respectivos valores que fazem parte da venda.

Esse conceito é muito difundido em comércio eletrônico (e-commerce), porém neste caso, a ideia é trabalhar de forma semelhante em lojas físicas.

Para habilitar este recurso, o lojista (seller) deverá marcar a checkbox "Marketplace" na tabela "Contrato" através do módulo de configuração do Portal Web do SCOPE.

Se habilitado, o SCOPE irá solicitar os dados de "Split de Pagamento" através do estado de coleta abaixo:

- TC_COLETA_DADOS_SPLIT_PAGAMENTO = 0xFCF7

Caso existam dados, a automação comercial deverá usar a função ScopeForneceCampo com o indicador abaixo para fornecer os dados requeridos:

- SCOPE_DADOS_SPLIT_PAGAMENTO = 39

Caso não existam dados, basta seguir o fluxo normalmente (através da ScopeResumeParam), ou seja, os dados de "Split de Pagamento", mesmo quando habilitado, são opcionais.

Os dados de "Split de Pagamento" deverão ser fornecidos utilizando-se os formatos (estruturas) definidos abaixo:

- stSplitPagamentoF01: Formato 01 - Seller
- stSplitPagamentoF02: Formato 02 - Subsellers
- stSplitPagamentoF03: Formato 03 - Itens dos Subsellers

A estrutura de "Formato 01" (Seller) deverá ser fornecida uma única vez.

A estrutura de "Formato 02" (Subsellers) deverá ser fornecida uma vez para cada subseller participante da venda.

A estrutura de "Formato 03" (Itens dos Subsellers) deverá ser fornecida uma vez para cada item de cada subseller de compõem a venda.

As estruturas estão definidas no módulo "ScopeApi.h", conforme mostrado abaixo:

```
// Estruturas para Split de Pagamento
// Solicitadas em TC_COLETA_DADOS_SPLIT_PAGAMENTO
// Fornecidas por ScopeForneceCampo com SCOPE_DADOS_SPLIT_PAGAMENTO
typedefstruct
{
    char Formato[2]; // Para permitir novos formatos futuros
    char Versao[2]; // Para permitir novas versões futuras
} stHeaderSplitPagamento, *LPHeaderSplitPagamento;

// Formato 01 - Seller
typedefstruct
{
    stHeaderSplitPagamento sHeader;
    char SellerId[36];
    char OrderId[36];
} stSplitPagamentoF01, *LPSplitPagamentoF01;

// Formato 02 - Subsellers
typedefstruct
{
    stHeaderSplitPagamento sHeader;
    char SubSellerId[36];
    char ValorTotal[12];
} stSplitPagamentoF02, *LPSplitPagamentoF02;

// Formato 03 - Itens dos Subsellers
typedefstruct
{
    stHeaderSplitPagamento sHeader;
    char SubSellerId[36];
    char ItemId[15];
    char Descricao[80];
    char ValorItem[12];
    char ValorJuros[12]; // Opcional (uso futuro)
    char ValorTaxa[12]; // Opcional
    char PorcentagemTaxa[9]; // Opcional
} stSplitPagamentoF03, *LPSplitPagamentoF03;
```

NOTA: Um exemplo pode ser obtido no arquivo fonte demo.c do módulo TstColeta.exe.

Roteamento Prioritário para Marketplace

A GetNetLAC permite Split de Pagamento para os produtos de Crédito e Débito. Assim, pode-se usar as funções ScopeCompraCartaoCredito e ScopeCompraCartaoDebito normalmente. Se o serviço selecionado permitir e a checkbox "Habilita Marketplace" estiver marcada para o contrato em questão, o estado TC_COLETA_DADOS_SPLIT_PAGAMENTO irá permitir à aplicação de Automação Comercial fornecer os dados de Split de Pagamento.

Alternativamente, a aplicação de Automação Comercial poderá usar as funções abaixo, indicando ao SCOPE que a rede que fornecer a funcionalidade de Split de Pagamento (necessário, GetNetLAC v2.99) deve ser priorizada.

Exemplo:

ScopeCompraCartaoCredito: Compra sem Marketplace: TEF deve seguir prioridade do Perfil de Serviços;
ScopeCompraCartaoCreditoMarketplace: Compra com Marketplace: TEF deve ser enviada para GetNetLAC;

Protótipo:

```
LONG EXPORT ScopeCompraCartaoCreditoMarketplace (char *Valor, char *TxServico)
```

Parâmetros:

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço

Protótipo:

```
LONG EXPORT ScopeCompraCartaoDebitoMarketplace (char *Valor)
```

Parâmetros:

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
------	-----------------------------------	-------	--

Os códigos de erro possíveis são os mesmos das funções ScopeCompraCartaoCredito e ScopeCompraCartaoDebito.

Bradescard

Esta é a implementação da especificação Interface Transacional Rede Bradescard da rede Bradescard. Com esta implementação, será possível realizar as seguintes transações da Bradescard no SCOPE:

- Compra Crédito à vista, parcelado sem juros e com juros;
- Consulta de Saldos e Limites;
- Consulta Planos de Compra Crédito com Juros;
- Saque;
- Consulta Planos de Saque;
- Recebimento de Fatura à vista ou parcelada;

- Consulta Recebimento de Fatura;
- Consulta de Extrato Resumido;
- Consulta de Fatura Detalhada;
- Pagamento de Contas.

Modos de entrada permitidos

Abaixo uma tabela com os modos de entrada permitidos para cada transação.

Operação	Entrada			
	*CPF	Voucher	Cartão	Barcode
Compra Crédito	X	X	X	-
Consulta de Saldos e Limites	X	-	X	-
Consulta Planos de Compra Crédito com Juros	-	X	X	-
Saque	-	-	X	-
Consulta Planos de Saque	-	-	X	-
Recebimento de Fatura	X	-	X	X
Consulta Recebimento de Fatura	X	-	X	-
Consulta de Extrato Resumido	X	-	X	-
Consulta de Fatura Detalhada	X	-	X	-
Pagamento de Contas	-	-	X	-
Cancelamento	-	-	X	-

*A partir da versão 6.02 o modo de entrada CPF foi descontinuado.

PINPads com suporte à rede

Para a realização das transações com cartões com chip nos PINPads deve-se garantir que o PINPad saiba tratar estes cartões da Bradescard. São suportados PINPads com a versão da especificação da biblioteca compartilha 1.08a ou superior, ou ainda biblioteca ABECS 2.12 ou superior. O PINPad deve conter chave de criptografia (MK) da Software Express. Para eventuais recomendações e maiores detalhes de compatibilidade de PINPad com cartões Bradescard entrar em contato com o banco Bradesco.

Associação das transações Bradescard versus funções do SCOPE

Abaixo está uma tabela onde são listadas as transações da Bradescard e as funções da API do SCOPE que devem ser invocadas para realizar a respectiva transação.

Transação	Função (API SCOPE)	Bandeira
Compra Crédito	ScopeCompraCartaoCredito	-
Compra Crédito por CPF	ScopeCompraCartaoCreditoCPF	402
Consulta de Saldos e Limites	ScopeConsultaSaldoCredito	-
Consulta de Saldos e Limites por CPF	ScopeConsultaSaldoCreditoCPF	402
Consulta Planos de Compra Crédito com Juros	ScopeConsultaCredito	-
Saque	ScopeTransacaoFinanceira (74) ou ScopeSaque	-
Consulta Planos de Saque	ScopeTransacaoFinanceira (73)	-
Recebimento de Fatura	ScopePagamento (91)	402
Recebimento de Fatura por CPF	ScopePagamentoCPF (91)	402

Consulta Recebimento de Fatura	ScopeConsultaPgtoFatura (178)	402
Consulta Recebimento de Fatura por CPF	ScopeConsultaPgtoFaturaCPF (178)	402
Consulta de Extrato Resumido	ScopeTransacaoFinanceira (71)	-
Consulta de Extrato Resumido por CPF	ScopeTransacaoFinanceiraCPF (71)	402
Consulta de Fatura Detalhada	ScopeTransacaoFinanceira (179)	-
Consulta de Fatura Detalhada por CPF	ScopeTransacaoFinanceiraCPF (179)	402
Pagamento de Contas	ScopePagamento (85)	-
Cancelamento	ScopeCancelamento	-

*A partir da versão 6.02 as transações por CPF foram descontinuadas.

Comprovantes

Abaixo estão os comprovantes existentes para cada transação:

Operação	Emissão		
	Loja	Cliente	Reduzido
Compra Crédito	X	X	-
Consulta de Saldos e Limites	-	X	-
Consulta Planos de Compra Crédito com Juros	-	-	-
Saque	X	X	-
Consulta Planos de Saque	-	-	-
Recebimento de Fatura	X	X	-
Consulta Recebimento de Fatura	-	X	-
Consulta de Extrato Resumido	-	X	-
Consulta de Fatura Detalhada	-	X	-
Pagamento de Contas	X	X	-
Cancelamento	X	X	-

Dados coletados

Abaixo está a relação do que pode ser solicitado em cada transação:

Transação	Coletas
Compra Crédito	<ul style="list-style-type: none"> • Cartão • Validade do cartão (se digitado e habilitado) • Consulta? (TC_DECIDE_CONSULTA_PARCELAMENTO = 0xFCF2) • Senha (a partir de v6.02 exigida para consulta) • Exibir consulta (TC_IMPRIME_CONSULTA = 0xFC1B) • Continua? (TC_DECISAO_CONT = 0xFC1C) • Parcelado ou a vista • Parcelado Lojista ou Adm • Quantidade de parcelas • Carência (dias ou ciclos) (TC_QTD_DIAS=0xFC1F ou TC_CICLOS_A_PULAR = 0xFC27) • Categoria de plano (TC_PLANO_PAGAMENTO=0xFC26)

	<ul style="list-style-type: none"> Parcela Grátis (TC_COLETA_PARCELA_GRATIS=0xFCF9) Código de autorização(se cartão digitado) (TC_COLETA_COD_AUTORIZACAO_CREDITO = 0xFC3D) Senha Imprimir cupom
Compra Crédito por CPF	<ul style="list-style-type: none"> CPF (TC_CPF=0xFC66) Escolha de cartão (TC_COLETA_DADO_ESPECIAL=0xFCF4) Consulta? (TC_DECIDE_CONSULTA_PARCELAMENTO = 0xFCF2) Exibir consulta (TC_IMPRIME_CONSULTA = 0xFC1B) Continua? (TC_DECISAO_CONT = 0xFC1C) Parcelado ou a vista Parcelado Lojista ou Adm Quantidade de parcelas Carência (dias ou ciclos) (TC_QTD_DIAS=0xFC1F ou TC_CICLOS_A_PULAR = 0xFC27) Categoria de plano (TC_PLANO_PAGAMENTO=0xFC26) Parcela Grátis (TC_COLETA_PARCELA_GRATIS=0xFCF9) Senha (somente se rede exigir via carga de parâmetros) Dados validação positiva (Pinpad) Imprimir cupom
Consulta de Saldos e Limites	<ul style="list-style-type: none"> Cartão Senha Imprimir cupom
Consulta de Saldos e Limites por CPF	<ul style="list-style-type: none"> CPF (TC_CPF=0xFC66) Escolha de cartão (TC_COLETA_DADO_ESPECIAL=0xFCF4) Dados validação positiva (Pinpad) Imprimir cupom
Consulta Planos de Compra Crédito com Juros	<ul style="list-style-type: none"> Cartão Validade do cartão Valor Código de autorização (se cartão digitado e habilitado) Categoria de plano (TC_PLANO_PAGAMENTO=0xFC26) Senha Imprimir/Exibir consulta (TC_IMPRIME_CONSULTA=0xFC1B)
Saque	<ul style="list-style-type: none"> Cartão Valor Parcelado ou a vista Quantidade de parcelas Adesão ao seguro (TC_COLETA_SEGURO=0xFC4C) Carência (dias ou ciclos) Categoria de plano (TC_PLANO_PAGAMENTO=0xFC26) Parcela Grátis (TC_COLETA_PARCELA_GRATIS=0xFCF9) Senha Imprimir cupom
Transação	Coletas
Consulta Planos de Saque	<ul style="list-style-type: none"> Cartão Validade do cartão Valor Quantidade de parcelas Categoria de plano (TC_PLANO_PAGAMENTO=0xFC26) Senha Imprimir/Exibir consulta (TC_IMPRIME_CONSULTA=0xFC1B)
Recebimento de Fatura	<ul style="list-style-type: none"> Código de Barras ou Cartão Validade do cartão Consulta? (TC_DECIDE_CONSULTA=0xFC5F) Senha (a partir de v6.02) Exibir consulta (TC_IMPRIME_CONSULTA = 0xFC1B)

	<ul style="list-style-type: none"> • Valor • Data de vencimento • Parcelado ou a vista • Quantidade de parcelas • Senha • Imprimir cupom
Recebimento de Fatura por CPF	<ul style="list-style-type: none"> • CPF (TC_CPF=0xFC66) • Escolha de cartão (TC_COLETA_DADO_ESPECIAL=0xFCF4) • Consulta? (TC_DECIDE_CONSULTA=0xFC5F) • Exibir consulta (TC_IMPRIME_CONSULTA = 0xFC1B) • Valor • Data de vencimento • Parcelado ou a vista • Quantidade de parcelas • Dados validação positiva (Pinpad) • Imprimir cupom
Consulta Recebimento de Fatura	<ul style="list-style-type: none"> • Cartão • Validade do cartão • Senha • Imprimir cupom
Consulta de Extrato Resumido	<ul style="list-style-type: none"> • Cartão • Validade do cartão • Senha • Imprimir cupom
Consulta de Extrato Resumido por CPF	<ul style="list-style-type: none"> • CPF (TC_CPF=0xFC66) • Escolha de cartão (TC_COLETA_DADO_ESPECIAL=0xFCF4) • Dados validação positiva (Pinpad) • Imprimir cupom
Consulta de Fatura Detalhada	<ul style="list-style-type: none"> • Cartão • Validade do cartão • Senha • Imprimir cupom
Consulta de Fatura Detalhada por CPF	<ul style="list-style-type: none"> • CPF (TC_CPF=0xFC66) • Escolha de cartão (TC_COLETA_DADO_ESPECIAL=0xFCF4) • Dados validação positiva (Pinpad) • Imprimir cupom
Pagamento de Contas	<ul style="list-style-type: none"> • Cartão • Validade do cartão • Parcelado ou a vista • Parcelado Lojista ou Adm • Quantidade de parcelas • Senha • Imprimir cupom
Cancelamento	<ul style="list-style-type: none"> • Número de controle da compra • Cartão • Validade do cartão • Senha (não exigida a partir de v6.02) • Imprimir cupom

Transação de Consulta Planos de Crédito com Juros

Essa transação pode ser acionada de duas formas:

- Integrada no fluxo da transação de Compra Crédito;
- Isolada através de função própria (ScopeConsultaCredito).

Não há emissão de comprovante pela rede, devendo a automação consumir e/ou exibir os dados retornados através de chamada de função específica ([ScopeObtemDadosAdicionais](#)) dentro do estado de coleta TC_IMPRIME_CONSULTA = 0xFC1B. Para conhecer o leiaute dos dados retornados solicitar e consultar o Anexo I – Bloco de Dados de Operações Bradescard.

Transação de Consulta Planos de Saque

Essa transação pode ser acionada somente de forma isolada.

Não há emissão de comprovante pela rede, devendo a automação consumir e/ou exibir os dados retornados através de chamada de função específica ([ScopeObtemDadosAdicionais](#)) dentro do estado de coleta TC_IMPRIME_CONSULTA = 0xFC1B. Para conhecer o leiaute dos dados retornados solicitar e consultar o Anexo I – Bloco de Dados de Operações Bradescard.

Transação de Consulta Recebimento de Fatura

Essa transação pode ser acionada de duas formas:

- Integrada no fluxo da transação de Recebimento de Fatura;
- Isolada através de função própria (ScopeConsultaPgtoFatura).

Quando chamada de forma integrada, é recomendado o consumo dos retornados através de chamada de função específica ([ScopeObtemDadosAdicionais](#)) dentro do estado de coleta TC_IMPRIME_CONSULTA = 0xFC1B. Quando chamada de forma isolada é recomendada a impressão do comprovante no estado TC_IMPRIME_CUPOM = 0xFC02. Para conhecer o leiaute dos dados retornados solicitar e consultar o Anexo I – Bloco de Dados de Operações Bradescard.

Transações com uso de CPF

São transações similares às respectivas transações originais (sem CPF), com algumas diferenças:

- A Automação Comercial deve informar a bandeira Bradescard Cred (402) e a opção de coleta de CPF do cliente no PINPAD (0) na chamada da função.
- O SCOPE aciona de forma integrada a transação Bradescard Consulta Dados do Cliente a qual retorna um ou mais cartões associados ao CPF do cliente.
- Caso seja retornado apenas um número de cartão (PAN), este é utilizado na sequencia da transação de compra.
- Caso a consulta retorne mais de um PAN , uma lista (dinâmica) será disponibilizada ao sistema de Automação Comercial no estado TC_COLETA_DADO_ESPECIAL (0xFCF4), que por sua vez, deverá apresentar em tela, capturar a escolha do cliente, e selecionar o PAN desejado através das funções abaixo:

Função	Descrição
ScopeMenuRecuperaltens	Recupera os itens disponíveis.
ScopeMenuSelecionaltem	Seleciona o item desejado.

- Serão sorteados 3 (três) dados de validação positiva a serem coletados pelo SCOPE diretamente do cliente através do PINPAD.

- A senha do cartão pode ou não ser coletada no PINPAD condicionada à parametrização da rede Bradescard.

Hub

Saque com Cartão de Crédito

A transação de Saque com cartão de Crédito pode ser executada através de uma das funções abaixo:

- ScopeSaque;
- ScopeTransacaoFinanceira passando o serviço 74 – Saque Crédito no segundo parâmetro;

Para detalhes de uso destas funções vide suas respectivas declarações neste documento.

O modo de entrada do cartão deverá ser preferencialmente por chip. O modo digitado será aceito se habilitado no contrato pelo lojista e pela rede Hub, através de inicialização de tabelas. O modo de leitura por tarja magnética, embora aceito pelo SCOPE, poderá ser negado pela rede Hub, já que não foi certificado pela bandeira Mastercard/Visa.

Para instruções de configuração e particularidades da rede Hub vide tópico específico no Manual de Instalação e Configuração.

RAPPI

Utiliza-se da função Carteira Virtual

O usuário comprador não está presente fisicamente, pois realizou o pedido através do "app Rappi" e já existe uma pré-cobrança externa ao nosso sistema.

O Picker (funcionário RAPPI) está presencialmente no estabelecimento para realizar as compras e solicitar autorização do pagamento no checkout (PDV). Através do app de celular, ele exibirá um qrCode para ser lido no PDV.

Leitura Scanner

Durante o fluxo de Carteira Virtual, a automação comercial deverá passar o qrcode ou código de barras exibido no celular do cliente e lido pelo scanner. O estado de coleta para este caso é:

- **TC_COLETA_LEITURA_SCANNER = 0xFCFA**

A AC deverá fornecer essa informação através da função ScopeForneceCampoEx com tipo de dado SCOPE_LEITURA_SCANNER = 44, que usa o formato abaixo:

```
struct stLeituraScanner {
    char ModoEntrada[1];           // '1'=Scanner, '2'=Digitado...
    char TamConteudo[3];           // Tamanho do dado a seguir (ASCII)
    char Conteudo[40];            // conteúdo com tamanho variável (ASCII)
};
```

ATENÇÃO: Essa informação também deverá ser passada durante o Estorno(Cancelamento) da transação.

Produtos Comprados

Durante o fluxo de Carteira Virtual, o SCOPE solicitará a relação de "produtos comprados". O estado de coleta para este caso é:

- **TC_COLETA_LISTA_MERCADORIAS = 0xFCCB**

A AC deverá fornecer a relação de produtos comprados através da função ScopeForneceCampoEx com tipo de dado SCOPE_DADOS_LISTA_MERCADORIAS_CV= 45, que usa o formato abaixo:

```

#define QTD_MAX_MERCADORIAS_CV    70
#define HEADER_LISTA_MERCADORIAS_CV "CV01"

struct stMercadoriaCV {
    char    EAN[14];           /* Codigo de Venda do produto */
    char    SKU[14];           /* Codigo Interno do produto cadastrado pelo lojista */
    char    Descricao[40];     /* Descrição da mercadoria */
    char    QtdMercadoria[13]; /* Quantidade de Mercadoria com 3 casas decimais (ex: 1000 significa
1,000) */
    char    ValorUnitario[9];   /* Valor unitário da mercadoria */
    char    TipoUnidade[2];    /* Tipo da unidade ('UN': Unidade, 'KG': Peso(kilograma),
'L':Litro, 'PK': Empacotado como Pack/Kit) */
    char    ValorDesconto[9];   /* Valor do desconto */
};

struct stListaMercadoriasCV{
    char    IdLayout[4];        // Identificador do layout da lista (= 'CV01')
    char    Continua[1];        // 'S' = tem proxima lista com mais mercadorias, 'N' = não tem continuação
    char    Sequencial[2];      // numero sequencial incrementado a cada lista enviada (sempre começa com
1)
    char    Moeda[3];          // Moeda utilizada (BRL, USD, EUR, etc.)
    char    ValorTotal[12];
    char    TotalDesconto[12];
    char    QtdReq[3];
    struct stMercadoriaCV sProduto[QTD_MAX_MERCADORIAS_CV];
};

```

ATENÇÃO: Essa informação também deverá ser passada durante o Estorno (Cancelamento) da transação.

Cielo – Parcelado Cliente

Trata-se de um serviço de Crédito Parcelado, semelhante ao Crediário de Crédito, porém com “acréscimo de valor”. Disponível a partir do Release R2024 (Release 3 na nomenclatura Cielo).

A Automação Comercial deve, antes de iniciar a transação de Crédito, chamar a função ScopeForneceCampo com o parâmetro SCOPE_AUTOMACAO_PERMITE_ACRESIMO informando “1” para indicar que está preparada para tratar o acréscimo no valor da transação, conforme exemplo abaixo.

Exemplo

```

...
ScopeForneceCampo(SCOPE_AUTOMACAO_PERMITE_ACRESIMO, "1");
...

```

A CIELO recomenda fortemente que este valor de acréscimo seja incluído na nota ou cupom fiscal.

O cliente deverá pagar o valor total acrescido, porém, este valor do acréscimo será retido pela CIELO no resarcimento da transação, a título de Taxa Administrativa.

O valor do acréscimo não estará presente no comprovante da TEF por opção da CIELO.

No caso da transação aprovada com acréscimo, a função ScopeStatus, ao final da transação, irá retornar 264 (RCS_APROVADO_COM_ACRESCIMO = 0x0108) ao invés de 00 (zero), para indicar que a Automação Comercial deve obter o valor do acréscimo e tratá-lo adequadamente.

A obtenção do Valor do Acréscimo é feita através da função ScopeObtemCampoExt3 com o parâmetro Valor_Acrescimo = 0x00100000 da Máscara 3, conforme exemplo abaixo.

Exemplo

```
...
if (RC == RCS_APROVADO_COM_ACRESCIMO)
{
    char szBuffer[128 + 1];
    ScopeObtemCampoExt3(Handle, 0x00, 0x00, Valor_Acrescimo, 0x00, '#', szBuffer);
}
...
...
```

No que se refere ao fluxo operacional de coleta, a funcionalidade de Parcelado Cliente é semelhante à do Crediário Crédito. Inclusive, deve ser usada as mesmas funções de obtenção de dados (ScopeObtemDadosCrediaroCredito) nos formatos 0 (TLV) e 1 (estrutura).

Para maiores informações, vide tópico ScopeObtemDadosCrediaroCredito.

Simulação Parcelado Cliente

Assim como no Crediário, opcionalmente pode-se fazer uma simulação em fluxo isolado através da função abaixo, que é equivalente à função ScopeSimulacaoCredario.

Protótipo

```
LONG EXPORT ScopeSimulacaoParcCliente (char *Valor, char *TxServico)
```

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String	TxServico	Valor da taxa de serviço (se houver).

Retorno

Ver [tabela de código de retorno](#).

Cielo – DCC Digitado

Normalmente DCC – Dynamic Currency Conversion não permite transações digitadas, porém a Cielo é uma exceção. A expectativa é que o DCC digitado tenha um uso restrito em produção, já que apenas com chip é possível identificar que se trata de um cartão emitido no exterior.

A funcionalidade DCC está disponível a partir do Release R2024 (Release 3 na nomenclatura Cielo).

No caso de cartão lido, a escolha da moeda é feita pelo cliente no PINPad, porém, como no digitado normalmente não há PINPad disponível, a seleção deve ser feita pelo Operador.

Este é o único caso em que DCC afetou a Automação Comercial, pois para cartão lido pelo PINPad, a AC não foi afetada.

Para tratar a seleção da moeda pelo Operador, a AC deverá obter as informações as serem exibidas, pois as duas linhas de 40 caracteres de display não foram suficientes para atender a exigência de display da Cielo.

Portanto, no caso de DCC Digitado o SCOPE Client pode cair no seguinte estado de coleta abaixo:

TC_COLETA_MOEDA = 0xFCEO

Note que este estado de coleta não foi criado exclusivamente para CIELO. Há outras situações no SCOPE em que este estado é usado para tratar a seleção da moeda.

Para obter a lista de moedas deve-se usar a função ScopeRecuperaBufTabela conforme abaixo.

Protótipo

```
LONG EXPORT ScopeRecuperaBufTabela (BYTE _TipoTabela,  
char *Qtd,  
char *ListaBuffer,  
WORD TamLista)
```

Descrição

- Esta rotina servirá para disponibilizar para a aplicação de automação comercial, uma lista de informações a serem exibidas na tela para facilitar a seleção da opção desejada.
- Ela servirá para facilitar futuras implementações que atendam a mesma finalidade.
- O parâmetro _TipoTabela indicará o formato dos dados que serão fornecidos.

Parâmetros

[out]	String [2+1]	Qtd	Retorna a quantidade de elementos da lista.
[out]	String	ListaBuffer	Retorna os registros da lista.
[in]	WORD	TamLista	Tamanho, em bytes, do campo "ListaBuffer"

Retorno

Ver [tabela de código de retorno](#).

Parâmetro TipoTabela:

ID layout	Usar estrutura	Descrição
BUF_TAB_MOEDAS_CIELO (ID 9)	stREGISTRO_MOEDA_CIELO	Contém 1 ou 2 opções de moeda

A estrutura a ser retornada possui o seguinte formato:

```

typedefstruct
{
    char szCodMoeda[3 + 1];
    char szSigla[3 + 1];
    char szValor[12 + 1];
    char szQtdDecValor[2 + 1];
} stMOEDA_CIELO, *LPMOEDA_CIELO;

typedefstruct
{
    char szDisplay[99 + 1];
    char szTaxaMarkup[5 + 1];
    char szCotacaoConversao[8 + 1];
    char szQtdDecCotacao[2 + 1];
    stMOEDA_CIELO sMoeda[2];
} stREGISTRO_MOEDA_CIELO, *LPREGISTRO_MOEDA_CIELO;

```

A quantidade de moedas a ser retornada será sempre um dos dois casos abaixo:

1 Moeda = Somente ocorre em transações de "crédito de captura", quando a seleção já foi feita anteriormente na transação de pré-autorização. Neste caso a informação deve ser exibida para confirmação.

2 Moedas = Devem ser exibidas as duas opções para seleção. Neste caso a seleção '1' sempre será moeda estrangeira (aceita oferta DCC) e seleção '2' sempre será moeda nacional (não aceita oferta DCC).

Exemplo:

```

...
if (RC == TC_COLETA_MOEDA)
{
    char szCodRede[3 + 1];
    memset(szCodRede, '\0', sizeof(szCodRede));
    if (ScopeObtemCampoExt3(scopeHandle, Cod_Redeh, 0x00, 0x00, 0x00, '\0', szCodRede)
== RCS_SUCESSO)
    {
        iCodRede = atoi(szCodRede);
    }

    if (iCodRede == R_CIELO)
    {
        ExibeListaMoedaCielo();
    }
...
...

```

```

void ExibeListaMoedaCielo(void)
{
    int iQtd = 0, i = 0;
    char szQtdReg[2 + 1] = { 0 };
    char szLinha[40 + 1];
    char szValorFormatado[16 + 1];
    stREGISTRO_MOEDA_CIELO      sMoedas;

    memset(&sMoedas, '\0', sizeof(sMoedas));

    if (ScopeRecuperaBufTabela(BUF_TAB_MOEDAS_CIELO, szQtdReg, (char*)&sMoedas,
        sizeof(stREGISTRO_MOEDA_CIELO)) == RCS_SUCESSO)
    {
        iQtd = min(atoi(szQtdReg),2);

        if (iQtd <= 0)
        {
            // Tratamento de erro
            return;
        }

        if (iQtd == 1)
        {
            // Captura DCC
            // Confirmação da moeda
            printf("\n\nCONFIRMACAO DA PRE-AUTORIZACAO COM DCC:\n");
            printf("Cliente ja escolheu pagar em moeda estrangeira\n");
            printf("Estabelecimento recebera o valor em Reais\n");
            printf("\n");

            if (sMoedas.szDisplay[0] != '\0')
            {
                printf(sMoedas.szDisplay);
                printf("\n\n");
            }
            else
            {
                memset(&szValorFormatado, '\0', sizeof(szValorFormatado));
        }
    }
}

```

```

        demo_FormataValor(sMoedas.sMoeda[0].szValor,
(int)strlen(sMoedas.sMoeda[0].szValor), szValorFormatado);
        memset(&szLinha, '\0', sizeof(szLinha));
        sprintf(szLinha, "Valor %s %s\n\n", sMoedas.sMoeda[0].szSigla,
szValorFormatado);

    }

    printf("Confirma? 1-Sim 2-Nao\n");
}

else

{

    // Oferta DCC
    // Seleção da moeda
    // 1-Moeda Estrangeira
    // 2-Moeda Nacional
    printf("\n\nOFERTA DCC:\n");
    printf("Cliente pode escolher moeda de pagamento\n");
    printf("Estabelecimento receberá o valor em Reais\n");
    printf("\n");

    if (sMoedas.szDisplay[0] != '\0')
    {

        printf(sMoedas.szDisplay);
        printf("\n\n");
    }

    printf("Selecione:\n");

    for (i = 0; i < iQtd; i++)
    {

        memset(&szLinha, '\0', sizeof(szLinha));
        memset(&szValorFormatado, '\0', sizeof(szValorFormatado));

        FormataValor(sMoedas.sMoeda[i].szValor,
(int)strlen(sMoedas.sMoeda[i].szValor), szValorFormatado);

        sprintf(szLinha, "\t%d-%s %s\n", i + 1, sMoedas.sMoeda[i].szSigla,
szValorFormatado);
        printf(szLinha);
    }
}

```

}
}
}

STIX – Fidelidade

Para realizar a operação Fidelidade deve-se utilizar a função `ScopeFidelidadeEx()`, como descrito abaixo.

Protótipo

Parâmetros

[in]	String com o máximo de 12 dígitos	Valor	Valor da transação com a vírgula implícita (exemplo: R\$ 123,00 = "12300")
[in]	String com o máximo de 40 dígitos	PedidoDeVenda	Informação obrigatória
[in]	WORD	Servico	Informação opcional, caso não seja especificada, enviar 0
[in]	WORD	Bandeira	Informação opcional, caso não seja especificada, enviar 0
[in]	BYTE	OpcColetaValidacao	Coleta validação via Pinpad (default). No caso de aplicação Mobile. Coleta via teclado Operador. 0 - Pinpad, 1 - Operador

Retorno

[Ver tabela de código de retorno.](#)

Perguntas Frequentes – Geral

O objetivo do capítulo é responder perguntas relacionadas a dúvidas em relação à utilização das funções e transações do SCOPE, no âmbito geral.

1. Como informar a forma de pagamento Vale refeição/alimentação?

Resposta: Faça uma transação de débito.

2. A biblioteca ScopeApi faz a coleta pelo pinpad ou temos que intermediar via PDV?

Resposta: O SCOPE interage com o pinpad, deixando transparente para a automação.

3. No modo coleta está pedindo, número do cartão, validade, informações essas que deveriam ser obtidas do PinPad e não da aplicação PDV.

Resposta: É provável que o seu terminal não tenha um pinpad configurado. É necessário verificar a sua configuração.

4. Teria um exemplo básico na linguagem C de compra com cartão de crédito pelo PinPad com senha informada no próprio PinPad?

Resposta: A interação com o pinpad não é feita pela aplicação, logo não há um exemplo que garanta que irá interagir com o pinpad. Basta configurar no SCOPECNF um pinpad associado ao seu terminal e ter um pinpad conectado ao PDV, e consequentemente, no fluxo será coletada a senha.

5. Como Aplicação PDV informa a forma de pagamento (Crédito, Débito, Vale Refeição/ Alimentação), valor total, valor da parcela, quantidade de parcelas ?

Resposta: A forma de pagamento é acionada por funções específicas (ex. ScopeCompraCartaoCredito(), ScopeCompraCartaoDebito()).

O valor total é um parâmetro das funções de forma de pagamento. A quantidade de parcelas será solicitado no fluxo, mas antes o SCOPE pergunta se é à vista. Caso o operador responda não, o SCOPE poderá perguntar se parcelado pela loja ou pela administradora, antes de perguntar a quantidade de parcelas.

6. Como o PinPad informa o valor e solicita a senha ?

Resposta: É transparente para a aplicação, depende apenas de configuração. (Veja Pergunta 4 anterior).

7. O PDV é notificado sobre evento de solicitação de senha ?

Resposta: Por padrão, não. A aplicação deve informar ao SCOPE Client que ela quer saber quando está acontecendo uma interação do pinpad com o portador do cartão. Para isto basta chamar a função ScopeConfigura(), passando o valor 1 no primeiro e no segundo parâmetro.

Isso fará com que o SCOPE retorne o valor 0xFCFC quando estiver solicitando o cartão e o valor 0xFCFD, quando estiver coletando a senha.

8. O PDV é notificado de compra aprovada ou cancelada ?

Resposta: Sim. No final do fluxo, a função ScopeStatus() encerra com o código 0 (zero) para transações finalizadas com sucesso ou qualquer outro valor para transações com erro.

9. O PDV obtém as informações necessárias para imprimir o cupom e cancelar a compra ?

Resposta: Durante a coleta, o SCOPE irá retornar o código 0xFC02, que indica que a automação deve chamar a função ScopeGetCupomEx() para obter os cupons da transação. O estorno é feito pela função ScopeCancelamento().

10. Iremos utilizar somente as funções de compra com PinPad, Teria como simular essas funções sem um PinPad real, tipo usando apenas um simulador?

Resposta: Não temos nenhum simulador de pinpad. É necessário um pinpad físico.

11. No modo coleta está enviando TC_INFO_RET_FLUXO, mas não sai desse status, mesmo chamando a função ScopeResume, não sabemos como chamar a função ScopeResumeParam para esse estado, teria um exemplo com fluxo completo no modo coleta?

Resposta: Veja o Apêndice C . No estado TC_INFO_RET_FLUXO, devem ser mostradas as informações para o Operador e o fluxo deve ser retornado para o Scope Client, até que o Status retornado pela função ScopeStatus se modifique, por completar a Coleta ou devido algum condição detectada de erro, e assim uma ação diferente seja requerida.

Apêndice A – Tabelas

Códigos de retorno

Coleta de dados

Estes códigos serão retornados pela função [ScopeStatus\(\)](#) quando a aplicação optar pela coleta dos dados através da função [ScopeSetAplColeta\(\)](#). Estes códigos informam para a aplicação de PDV a ação a ser tomada. Como exemplo, coletar um dado, imprimir um cupom e/ou cheque, ou até mesmo mostrar informações no “display” do operador e/ou cliente aguardando a confirmação.

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFC00	64512	Coletar cartão
0xFC01	64513	Coletar validade do cartão
0xFC02	64514	Imprime Cupom
0xFC03	64515	Coletar CGC ou CPF
0xFC04	64516	Coletar banco
0xFC05	64517	Coletar agência
0xFC06	64518	Coletar número do cheque
0xFC07	64519	Coletar data do cheque (bom para)
0xFC08	64520	Imprime cheque
0xFC09	64521	Coletar se a transação será a vista ou não
0xFC0A	64522	Coletar se a transação será parcelada pela administradora ou pelo estabelecimento
0xFC0B	64523	Coletar se a transação será pré-datada
0xFC0C	64524	Coletar se a parcela será à vista
0xFC0D	64525	Coletar quantidade de dias entre parcelas
0xFC0E	64526	Coletar quantidade de parcelas
0xFC0F	64527	Coletar o plano de financiamento
0xFC10	64528	Coletar o dia e o mês (DDMM)
0xFC11	64529	Coletar a senha
0xFC12	64530	Coletar o controle do SCOPE
0xFC13	64531	Coletar a forma de pagamento
0xFC14	64532	Coletar data do primeiro vencimento
0xFC15	64533	Coletar valor de entrada
0xFC16	64534	Coletar a forma de entrada
0xFC17	64535	Coletar conta corrente
0xFC18	64536	Coletar últimos dígitos do cartão
0xFC19	64537	Reimpressão de comprovante
0xFC1A	64538	Coletar se deseja consultar parcelas
0xFC1B	64539	Imprime consulta
0xFC1C	64540	Coletar decisão de continuar
0xFC1D	64541	Coletar decide último
0xFC1E	64542	Coletar número de cheque CDC
0xFC1F	64543	Coletar quantidade de dias
0xFC20	64544	Coletar o número da pré-autorização
0xFC21	64545	Coletar dia do mês fechado
0xFC22	64546	Imprime nota promissória
0xFC23	64547	Coletar CEP

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFC24	64548	Coletar número do endereço
0xFC25	64549	Coletar parte numérica do complemento
0xFC26	64550	Coletar plano de pagamento
0xFC27	64551	Coletar ciclos a pular (Fininvest)
0xFC28	64552	Coletar número do item (Fininvest)
0xFC29	64553	Coletar código de segurança
0xFC2A	64554	Coleta se código de segurança ausente ou ilegível
0xFC2B	64555	Coleta se é com ou sem garantia de pré-datado
0xFC2C	64556	Coleta se aceita ou não risco
0xFC2D	64557	Coleta valor do saque
0xFC2E	64558	Coleta valor da recarga de celular pré-pago
0xFC2F	64559	Coleta código da localidade do telefone
0xFC30	64560	Coleta número do telefone
0xFC31	64561	Coleta dígito verificador do telefone
0xFC32	64562	Coleta data (formato DDMMAA)
0xFC33	64563	Coleta valor da taxa de serviço
0xFC34	64564	Coleta valor
0xFC35	64565	Coleta se quer realizar saque
0xFC36	64566	Coleta se quer realizar simulação de saque
0xFC37	64567	Coleta se quer saldo ou extrato
0xFC38	64568	Coleta se quer o extrato resumido ou a segunda via
0xFC39	64569	Coleta se é consulta investimento ou resgate
0xFC40	64576	Coleta se é resgate avulso
0xFC41	64577	Coleta data (formato DDMMAAAA)
0xFC42	64578	Coleta o código de autorização PBMS
0xFC43	64579	Coleta a lista de medicamentos
0xFC44	64580	Retorna a lista de medicamentos
0xFC45	64581	Exibir mensagem
0xFC46	64582	Imprime cupom parcial
0xFC47	64583	Coleta quantidade de parcelas e aceita 1 parcela
0xFC48	64584	Coleta código de barras
0xFC49	64585	Coleta código de consulta PBM
0xFC4A	64586	Coleta CRM médico
0xFC4B	64587	Coleta código UF CRM médico
0xFC4C	64588	Coleta se cliente deseja aderir ao seguro (IBICred)
0xFC4D	64589	Coleta se é pagamento com cartão
0xFC4E	64590	Coleta dados específicos da rede Tokoro
0xFC4F	64591	Coleta se deseja pagar após vencimento
0xFC50	64592	Coleta se a transação é com senha
0xFC51	64593	Imprime cupom promocional
0xFC52	64594	Coleta se utiliza saldo
0xFC53	64595	Coleta código do material
0xFC54	64596	Coleta código do plano
0xFC55	64597	Coleta se o pagamento é em cheque
0xFC56	64598	Coleta se confirma transação
0xFC57	64599	Coleta se o pagamento é no rotativo
0xFC58	64600	Coleta CMC7
0xFC59	64601	Coleta se o pagamento é em dinheiro ou TEF (cartão)
0xFC5A	64602	Coleta o código do grupo de serviço (para a TEF Externa)
0xFC5B	64603	Coleta o código da rede (para a TEF Externa)
0xFC5C	64604	Coleta o código do estabelecimento (para a TEF Externa)
0xFC5D	64605	Coleta o NSU do Host (para TEF Externa)
0xFC5E	64606	Coleta data (ddmmaaaa) (para TEF Externa)

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFC5F	64607	Coleta se é consulta
0xFC60	64608	Coleta se continua após verificar a conta
0xFC61	64609	Coleta o código da bandeira
0xFC62	64610	Decide se é conta ou fatura
0xFC63	64611	Coleta o valor total
0xFC64	64612	Coleta RG
0xFC65	64613	Coleta se deseja realizar novamente a transação (retentativa)
0xFC66	64614	Coleta somente CPF
0xFC67	64615	Coleta o endereço
0xFC68	64616	Coleta o andar
0xFC69	64617	Coleta o conjunto
0xFC6A	64618	Coleta o bloco
0xFC6B	64619	Coleta o bairro
0xFC6C	64620	Coleta a autorização ou o cartão
0xFC6D	64621	Coleta a data de emissão do cartão
0xFC6E	64622	Coleta o plano Infocards
0xFC6F	64623	Coleta número do cupom fiscal
0xFC70	64624	Coleta a operadora de recarga de celular pré-pago
0xFC71	64625	Coleta dados SAB
0xFC72	64626	Coleta o número do telefone com o dígito verificador
0xFC73	64627	Coleta os dados transação forçada SAB
0xFC74	64628	Coleta o tipo de serviço técnico: baixa técnica, teste comum ou estatística
0xFC75	64629	Coleta o número da OS
0xFC76	64630	Coleta identificação do técnico
0xFC77	64631	Coleta o código de ocorrência
0xFC78	64632	Coleta a EPS credenciada
0xFC79	64633	Decide se coleta valor de entrada (não mais utilizado)
0xFC7A	64634	Decide se coleta valor da primeira parcela (não mais utilizado)
0xFC7B	64635	Coleta o valor da primeira parcela
0xFC7C	64636	Coleta os dados adicionais
0xFC7D	64637	Coleta se cancela ou não a transação
0xFC7E	64638	Go On Chip
0xFC7F	64639	Retira o cartão
0xFC80	64640	Coleta o valor da taxa de embarque
0xFC81	64641	Exibe a mensagem de saldo
0xFC82	64642	Exibir a mensagem e retorna o fluxo (não mais utilizado)
0xFC83	64643	Exibir a mensagem aguarda confirmação do operador (não mais utilizado)
0xFC84	64644	Obtém os serviços
0xFC85	64645	Coleta o cartão digitado
0xFC86	64646	Coleta o código do produto
0xFC87	64647	Exibe o menu
0xFC88	64648	Coleta se é INSS ou cheque
0xFC89	64649	Coleta o contrato
0xFC8A	64650	Coleta a data quando o cliente aderiu ao cartão
0xFC8B	64651	Exibir o valor da consulta Vale Gás Disponibiliza valores da consulta de Cartão Dinheiro.
0xFC8C	64652	Coleta data da transação original no formato DDMMAA
0xFC8D	64653	Coleta o NSU da transação original (número de 6 dígitos)
0xFC8E	64654	Exibir os dados do cancelamento (não mais utilizado)
0xFC8F	64655	Coletar qual é a via da reimpressão (0: todas as vias; 1: apenas da loja; 2: apenas do cliente)
0xFC90	64656	Coleta o DDD no PINPad
0xFC91	64657	Coleta o número de telefone no PINPad

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFC92	64658	Coleta o número de telefone com o dígito verificador no PINPad
0xFC93	64659	Coleta a redigitação da recarga no PINPad
0xFC94	64660	Transação aprovada parcial
0xFC95	64661	Coleta o valor das parcelas
0xFC96	64662	Coleta se a primeira parcela é para 30 ou 60 dias
0xFC97	64663	Uso interno do SCOPE e não retornado para a aplicação
0xFC98	64664	Uso interno do SCOPE e não retornado para a aplicação
0xFC99	64665	Uso interno do SCOPE e não retornado para a aplicação
0xFC9A	64666	Coleta o Número do Cartão (para a TEF Externa)
0xFC9B	64667	Coleta o código de serviço do posto (transação de convênio combustível Banrisul).
0xFC9C	64668	Coleta a matrícula (transação de convênio combustível Banrisul)
0xFC9D	64669	Coleta a quantidade do serviço (transação de convênio combustível Banrisul)
0xFC9E	64670	Coleta o hodômetro (transação de convênio combustível Banrisul)
0xFC9F	64671	Coleta a placa do veículo (transação de convênio combustível Banrisul)
0xFCA1	64673	Coleta o número do resgate de prêmio
0xFCA2	64674	Opção de resgate de prêmio selecionada
0xFCA3	64675	Solicitação a confirmação do resgate
0xFCA4	64676	Coleta o número do resgate prêmio
0xFCA5	64677	Coleta o número do Voucher
0xFCA6	64678	Decide entre DARF e GPS
0xFCA7	64679	Decide entre DARF Preto e Simples
0xFCA8	64680	Coleta o código de receita
0xFCA9	64681	Coleta o número de referência
0xFCAA	64682	Coleta o valor juros
0xFCAB	64683	Coleta o CPF do portador
0XFCAE	64684	Coleta o CNPJ
0XFCAE	64685	Coleta o percentual
0XFCAE	64686	Coleta competência
0xFCAF	64687	Coleta o número identificador
0xFCB0	64688	Coleta o valor INSS
0xFCB1	64689	Coleta a receita bruta
0xFCB2	64690	Coleta a opção do operador para continuar, alterar ou cancelar
0xFCB3	64691	Coleta o valor de acréscimo
0xFCB4	64692	Coleta o valor de dedução
0xFCB5	64693	Coleta redigitação do DDD
0xFCB6	64694	Coleta os 8 primeiros dígitos do cartão no Pin Pad
0xFCB7	64695	Coleta os 8 dígitos finais do cartão no Pin Pad
0xFCB8	64696	Coleta tipo da consulta (transação de Consulta Saldo Credito Infocards, sendo: 1=Saldo, 2=Dados Cadastrais)
0xFCB9	64697	Confirma o número do cartão digitado no Pin Pad
0xFCBA	64698	Coleta validação do saque
0xFCBB	64699	Coleta saque em andamento
0xFCBC	64700	Coleta DDD + Telefone no PIN Pad
0xFCBD	64701	Redigita DDD + Telefone no PIN Pad
0xFCBE	64702	Coleta dados ECF
0XFCC1	64705	Coleta opção de pagamento de carnê (ou fatura) em transações de débito ou crédito
0XFCC6	64710	Coletar dados cartão presente
0xFCCA	64714	Coleta da Lista de Atualização de Preços de Mercadorias
0XFCCB	64715	Coleta da Lista de Mercadorias Consumidas
0XFCCC	64716	Coleta de Horímetro
0XFCCD	64717	Coleta de Cartão Magnético
0XFCD4	64724	Coleta de Segmento de Mercado

Códigos Retorno		Significado
Hexa decimal		Significado
Hexa decimal	Decimal	Significado
0xFCD5	64725	Coleta de Código de Fornecedor de Produto ou Serviço
0xFCD6	64726	Coleta de Código de Produto, Código de Serviço ou Código EAN
0xFCD7	64727	Coleta de Quantidade
0xFCD8	64728	Coleta de Confirmação de Cliente Preferencial
0xFCD9	64729	Coleta Ramo Principal Cielo Auto
0xFCDA	64730	Coleta Código de Mercadoria Cielo Auto
0xFCDB	64731	Coleta tipo cartão Débito ou Crédito
0xFCDC	64732	Coleta número de pontos fidelidade
0xFCDD	64733	Coleta código do produto de resgate
0xFCDE	64734	Coleta código da faixa de pontos fidelidade
0xFCDF	64735	Coleta de Código para Identificação pagamento de Fatura (Exemplos de código: CPF, Código de Barras e Cartão)
0xFCEO	64736	Coleta moeda
0xFCE4	64740	Coleta redigitação da data DDMMAAAA
0xFCE5	64741	Coleta redigitação do valor
0xFCE6	64742	Coleta nome do portador
0xFCE7	64743	Coleta código de segurança no PINPad
0xFCE8	64744	Coleta CPF ou CNPJ do beneficiário
0xFCE9	64745	Coleta CPF ou CNPJ do sacador
0xFCEA	64746	Coleta CPF ou CNPJ do pagador
0xFCEB	64747	Coleta valor do documento
0xFCEC	64748	Recuperar e exibir dados da consulta do BACEN para CORBAN
0xFCED	64749	Coleta terminal origem
0xFCEE	64750	Coleta de Operadora Online
0xFCEF	64751	Coleta de Valor de Recarga Online
0xFCFO	64752	Decide entre Pagamento com Cartão ou com Dinheiro
0xFCF2	64754	Decide se deve consultar parcelamento
0xFCF3	64755	Obtem QRCode
0xFCF4	64756	Coleta de dado "especial". Ao receber esse estado, a automação deverá chamar a função ScopeGetParamExt ao invés da ScopeGetParam.
0xFCF5	64757	Decide Crediário
0xFCF6	64758	Coleta CPF no pin pad
0xFCF7	64759	Coleta dados de Split de Pagamento (para Marketplace)
0xFCF8	64760	Decide nova consulta
0xFCF9	64761	Coleta opção de parcela grátis (1-Sim 0-Nao)
0xFCFB	64763	Coleta de dado "estendido". Ao receber esse estado, a automação deverá chamar a função ScopeGetParamExt ao invés da ScopeGetParam. A identificação do estado de coleta estendido deve ser feita por IdColetaExt, tabela abaixo.
0xFCFC	64764	Coleta cartão em andamento
0xFCFD	64765	Coleta em andamento
0xFCFE	64766	Mostrar informações e retornar fluxo para o cliente SCOPE
0xFCFF	64767	Mostrar Informações e aguardar confirmação do operador

Estados de Coleta Estendidos

Quando o estado de coleta retornado pela função ScopeStatus() for 0xFCFB (64763), deve-se usar a função ScopeGetParamExt para obter o estado de coleta estendido, identificado por IdColetaExt, conforme tabela abaixo.

Códigos Retorno		Significado
Hexa decimal	Decimal	
0x0000	0	Estado de coleta estendido não definido. Nesse caso o estado de coleta não é estendido. Pode ocorrer quando a aplicação de automação comercial usar a função ScopeGetParamExt para obter um estado de coleta “não estendido”.
0x0001	1	Coleta dinâmica solicitada pela rede TICKETLOG.
0x0002	2	Coleta ID do Operador
0x0003	3	Coleta Lista de Medicamentos
0x0004	4	Seleciona Provedor Digital
0x0005	5	Seleciona Produto Digital
0x0006	6	Coleta Produto Digital

Autorizadoras

Abaixo se encontram relacionados os códigos devolvidos pelas redes autorizadoras na mensagem 0210 e entregue para a aplicação através da função ScopeStatus().

Códigos Retorno		Significado
Hexa decimal	Decimal	
0x0000	0	Sucesso
0x0003	3	Estabelecimento comercial inválido
0x0006	6	Erro
0x0009	9	Transação em andamento
0x000C	12	Transação inválida
0x000D	13	Valor da transação inválido
0x000E	14	Cartão inválido
0x000F	15	Instituição não cadastrada
0x0013	19	Refaça transação
0x001E	30	Erro de formato
0x001F	31	Instituição não pertence à rede
0x0026	38	Excedido o número de tentativas do PIN
0x0029	41	Cartão extraviado
0x002B	43	Cartão roubado
0x0033	51	Saldo insuficiente
0x0034	52	Conta corrente não cadastrada
0x0036	54	Cartão vencido
0x0037	55	Senha incorreta
0x0038	56	Cartão sem registro
0x0039	57	Transação não permitida a esse cliente
0x003C	60	Entrar em contato com a instituição
0x003D	61	Excedido o limite de saque
0x0041	65	Excedida a frequência de saque
0x004C	76	Cartão bloqueado
0x004D	77	Pendente de confirmação
0x004E	78	Transação cancelada
0x004F	79	Transação não permitida neste ciclo
0x0050	80	Transação inexistente
0x0051	81	Transação estornada
0x0052	82	Chave de criptografia inválida
0x0053	83	Timeout
0x0054	84	Logon / Desfazimento
0x0055	85	Problema rede local
0x0056	86	Transação desfeita
0x0059	89	Mensagem enviada pelo Host

Códigos Retorno		Significado
Hexa decimal	Decimal	
0x005A	90	Fechamento contábil
0x005B	91	Instituição temporariamente fora de operação
0x0100	256	Retorno genérico para códigos alfanuméricos. Para obter o exato código alfanumérico retornado é necessário chamar a função ScopeObtemCampoExt() passando o bit equivalente ao código de resposta (ver Obtendo os campos).
0x0101	257	Cliente com restrição na lista negra local
0x0102	258	Já consultou sob mesmo banco e agência
0x0103	259	Já consultou sob banco e/ou agência diferente(s)
0x0104	260	Código GAR não autorizado
0x0105	261	Compre Saque REDE aprovado parcial
0x0106	262	Voucher REDE aprovado parcial
0x0107	263	Informação de transação aprovada com prêmio

Códigos de erros do SCOPE

Em qualquer momento, na chamada de qualquer função o SCOPE Client devolverá um código de retorno. O código que não estiver na relação dos códigos de coleta e na das autorizadoras, encontram-se abaixo e geralmente denota um erro.

Códigos Retorno		Significado
Hexa decimal	Decimal	
OxEE01	60929	Timeout Server-NA
OxEE02	60930	Transação não autorizada
OxFA01	64001	Parâmetro 1 inválido
OxFA02	64002	Parâmetro 2 inválido
OxFA03	64003	Parâmetro 3 inválido
OxFA04	64004	Parâmetro 4 inválido
OxFA05	64005	Parâmetro 5 inválido
OxFB01	64257	Não foi possível criar a "thread" na coleta de dados
OxFB02	64258	Erro na montagem do serviço pela API
OxFB03	64259	Erro ao verificar mensagem – mensagem inválida
OxFB04	64260	Erro ao montar mensagem
OxFB05	64261	Erro no arquivo de controle da TEF
OxFB06	64262	Erro no contexto do arquivo de TEF (não mais utilizado)
OxFB07	64263	Erro na totalização de TEF
OxFB08	64264	Erro no arquivo de controle utilizado finalização no ciclo multi-TEF
OxFB09	64265	Estourou o número máximo de TEF numa sessão multi-TEF
OxFB0A	64266	Não salvou a mensagem de confirmação para o SAB
OxFB0B	64267	Não salvou a mensagem de desfazimento para o SAB
OxFB0C	64268	Erro no processo de criptografia entre Client-Server
OxFE00	65024	A transação em andamento – a aplicação deve aguardar
OxFE01	65025	SCOPE API não foi inicializada corretamente
OxFE02	65026	SCOPE API já foi inicializada corretamente
OxFE03	65027	Existe transação suspensa
OxFE04	65028	Não existe transação suspensa

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFE05	65029	API ainda não fez nenhuma transação após a inicialização. A aplicação está tentando obter um <i>handle</i> sem ter feito nenhuma transação desde última conexão com o ScopeSRV
0xFE06	65030	Logon duplicado
0xFE07	65031	Protocolo não suportado – SCOPE Client com versão de protocolo superior ao SCOPE Server
0xFE08	65032	POS não cadastrado
0xFE09	65033	Servidor não configurado no arquivo scope.ini
0xFE0A	65034	Não há mais PDV's disponíveis. Erro retornado quando a aplicação passa o caractere "#" como PDV para que o SCOPE Server lhe dê um número de PDV válido, mas não há mais PDV's disponíveis
0xFE0B	65035	Protocolo incompatível – SCOPE Client com versão de protocolo extremamente antigo (P00, P01 ou P02).
0xFE0C	65036	Não pode mais desfazer uma transação que já iniciou o ciclo de finalização – situação que pode ocorrer apos queda de energia ou na inicialização da aplicação.
0xFE0D	65037	Não há arquivo com dados da transação anterior salvo
0xFF00	65280	ScopeSrv off-line ou o IP configurado no arquivo scope.ini está apontando para a máquina errada
0xFF01	65281	Instituição off-line
0xFF02	65282	Transação cancelada pelo operador ou no caso de um estorno via REDE: estorno fora do prazo permitido, validade não confere.
0xFF03	65283	Serviço ou BIN não configurado
0xFF04	65284	Transação já foi cancelada
0xFF05	65285	Transação não encontrada
0xFF06	65286	Transação não permite cancelamento
0xFF07	65287	Dados informados não conferem com a transação original
0xFF08	65288	Erro no acesso ao banco de dados
0xFF09	65289	Time-out no acesso ao banco de dados
0xFF0A	65290	Banco de dados off-line
0xFF0B	65291	Transação abortada pelo aplicativo
0xFF0C	65292	Transação não implementada
0xFF0D	65293	<i>Handle</i> inválido (ver Obtendo handle)
0xFF0E	65294	Taxa de serviço é inválida
0xFF0F	65295	Taxa de serviço excede limite
0xFF10	65296	Dado inválido
0xFF11	65297	Não existe cupom válido
0xFF12	65298	Área reservada para o buffer é insuficiente para o SCOPE Client preencher com os dados solicitados
0xFF13	65299	Límite inválido – inferior ao permitido
0xFF14	65300	Transação desfeita
0xFF15	65301	Digitação não permitida
0xFF16	65302	Memória insuficiente
0xFF17	65303	"Service Code" inválido
0xFF18	65304	Data inválida
0xFF19	65305	Cartão vencido
0xFF1A	65306	Cartão inválido
0xFF1B	65307	Desfazimento não disponível
0xFF1C	65308	Erro na impressão do cupom
0xFF1D	65309	Sessão em andamento – a transação solicitada deve ser única numa sessão de TEF, ou seja, não pode ser chamada se há transações na sessão de TEF atual
0xFF1E	65310	Transação já efetuada
0xFF1F	65311	Inserir chip do cartão

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFF20	65312	Controle obrigatório
0xFF21	65313	Pré-autorização obrigatória
0xFF22	65314	Serviço não configurado
0xFF23	65315	Serviço não definido
0xFF24	65316	Número de parcelas inválidas
0xFF25	65317	Valor inválido
0xFF26	65318	Serviço ou BIN não configurado para Visanet
0xFF27	65319	Estado de coleta não definido
0xFF28	65320	Operação não permitida
0xFF29	65321	CNPJ/CPF inválido
0xFF2A	65322	Primeiro bloco do código de barras está inválido
0xFF2B	65323	Segundo bloco do código de barras está inválido
0xFF2C	65324	Terceiro bloco do código de barras está inválido
0xFF2D	65325	Quarto bloco do código de barras está inválido
0xFF2E	65326	Dados adicionais AID do chip do cartão inexistente
0xFF30	65328	Autorizador retornou dados inválidos
0xFF31	65329	Conta não permitida
0xFF32	65330	Conta vencida
0xFF33	65331	Não existe resumo
0xFF34	65332	Código de barra inválido
0xFF35	65333	Erro na consistência do DAC
0xFF36	65334	Erro no envio da confirmação ou desfazimento da transação anterior
0xFF37	65335	Serviço invertido – a aplicação está tentando fazer uma transação de débito com um cartão de crédito ou vice-versa
0xFF38	65336	Cartão não permitido
0xFF39	65337	Permitido somente a coleta do CPF – consulta de cheque
0xFF3A	65338	Erro interno na execução da coleta
0xFF3B	65339	Lista (de produtos, de medicamentos, etc.) não está disponível.
0xFF3C	65340	Erro de leitura do cartão
0xFF3D	65341	Controle inválido
0xFF3E	65342	Erro ao enviar mensagem para o servidor
0xFF3F	65343	Interface SAB não inicializada
0xFF40	65344	Erro: dados ainda não disponíveis
0xFF41	65345	Erro: dados indisponíveis
0xFF42	65346	Servidor SAB off-line
0xFF43	65347	Erro de conexão entre SCOPE e SAB
0xFF44	65348	Erro no NSU recebido
0xFF45	65349	Erro no logon do PDV
0xFF46	65350	Erro no processamento do chip
0xFF47	65351	Operadora inválida
0xFF48	65352	Dados, da recarga de celular, não encontrados.
0xFF49	65353	Transação cancelada pelo cliente
0xFF50	65360	Transação aprovada off-line
0xFF51	65361	Versão do banco de dados incompatível
0xFF52	65362	Cancelamento fora do prazo permitido
0xFF53	65363	Mensagem inválida
0xFF54	65364	PIN-Pad não foi aberto
0xFF55	65365	PIN-Pad já foi aberto – a aplicação abriu o PIN-Pad antes de conectar ao ScopeSRV, mas não deveria ter-lo feito, pois no ScopeCNF está configurado com uso exclusivo do SCOPE
0xFF56	65366	Estado inválido – a aplicação está tentando obter os serviços disponíveis num estado de coleta inadequado

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFF57	65367	PIN-Pad compartilhado não está configurado, mas a rede exige que seja compartilhado
0xFF58	65368	PIN-Pad compartilhado não trabalha com a versão 2000 da VISA
0xFF59	65369	Função de uso exclusivo na interface coleta
0xFF5A	65370	Área insuficiente para os atributos do serviço
0xFF5B	65371	O SCOPE está configurado para uso de PIN-Pad compartilhado, mas a aplicação informou que está com o kit Visanet para PIN-Pad
0xFF5C	65372	O SCOPE não está configurado para uso de PIN-Pad compartilhado, mas a aplicação informou que está com PIN-Pad compartilhado
0xFF5D	65373	Erro ao inicializar periféricos
0xFF5E	65374	Erro ao desmontar a estrutura ISO
0xFF5F	65375	Bandeira não está configurada
0xFF60	65376	Função indisponível
0xFF61	65377	Valor mínimo da parcela é inválido
0xFF62	65378	Valor da consulta Vale Gás ainda não está disponível
0xFF63	65379	Número de Telefone inválido
0xFF64	65380	DDD inválido
0xFF65	65381	Erro Rede – Modelo 2
0xFF66	65382	Erro Rede – Modelo 3
0xFF67	65383	Confirmação Positiva Inconsistente
0xFF68	65384	Transação Offline – Permitido apenas reimpressão offline
0xFF69	65385	Contrato Suspenso
0xFF6A	65386	Transação permite somente digitado
0xFF6B	65387	Não Encontrado (para quando não veio menu dinâmico na carga de tabelas).
0xFF6F	65391	Erro ao acessar arquivo de contexto da PBM.
0xFF70	65392	Ticket Car – Placa Inválida
0xFF71	65393	Correspondente Bancário Bradesco – Consulta Não Habilitada – Habilite a Consulta Pagamento no Perfil de Serviços
0xFF72	65394	Código da Bandeira maior que 255.
0xFF73	65395	Rede em processo de inicialização.
0xFF74	65396	PINPad não suporta modo de multi-bandeira extendido
0xFF75	65397	Erro na transação de Estatística Automática da rede Rede
0xFF76	65398	PINPad com tabelas vazias. Verifique se todas as redes desta filial fizeram inicialização de tabelas com sucesso.
0xFF77	65399	Contactless não habilitado na carga de tabelas pela rede em questão para este estabelecimento. No caso da Cielo a habilitação é na tabela 1B – TerminalConfiguration.
0xFF78	65400	Contactless não permitido. AID em questão não possui dados contactless habilitados na carga de tabelas.
0xFF79	65401	Contactless sem produto habilitado pela rede em questão. No caso da Cielo o produto não está habilitado na tabela 7C – GrupoRangeFuncão.
0xFF7A	65402	Excede limite contactless. No caso da Cielo esse limite está estabelecido na tabela 4ª – RiscoContactless.
0xFF7B	65403	Modo Inválido – Insira cartão com chip na leitora.
0xFF7C	65404	Fallback não permitido
0xFF7D	65405	Moeda inválida
0xFF7E	65406	Modo de entrada difere da transação original. Use o mesmo modo de entrada da transação original.
0xFF7F	65407	Documento inválido
0xFF80	65408	Chamada à Função ScopeObtemTransacaold não permitida. È permitida somente durante uma operação de crédito ou débito.
0xFF81	65409	Valor máximo da parcela inválido.
0xFF82	65410	Verificação do Perfil de Pagamento Recorrente negada.

Códigos Retorno		Significado
Hexa decimal	Decimal	
0xFF83	65411	Erro de alocação de memória na função ScopeObtemTransacaold.
0xFF84	65412	Token utilizado nas funções de Pagamento Recorrente inválido.
0xFF85	65413	Cartão utilizado na função ScopeAlteraPreAutorizacaoCredito não corresponde ao utilizado na função ScopePreAutorizacaoCredito.
0xFF86	65414	Coleta do Terminal Origem nas funções ScopeAlteraPreAutorizacaoCredito e ScopeCapturaPreAutorizacaoCredito obrigatória.
0xFF87	65415	Pré-Autorização foi cancelada.
0xFF88	65416	Pré-Autorização já foi confirmada.
0xFF89	65417	Terminal Origem coletado nas funções ScopeAlteraPreAutorizacaoCredito e ScopeCapturaPreAutorizacaoCredito inválido.
0xFF8A	65418	Sem rede disponível para efetuar o débito para pagamento.
0xFF8B	65419	Contactless com valor zerado não permitido
0xFF8C	65420	Desconto não permitido
0xFF8D	65421	Produto não disponível
0xFF8F	65423	Erro na consulta dinâmica
0xFF90	65424	Chamada válida somente para PINpads ABECS
0xFF91	65425	Tamanho inválido, obrigatoriamente deve ser 8 bytes
0xFF92	65426	Não foi possível abrir o arquivo
0xFF93	65427	Nome do arquivo multimídia deve possuir somente caracteres numéricos e letras, sem espaços ou símbolos. Além disso, ele não é case sensitive.
0xFF94	65428	Tipo de arquivo inválido. Deve ser um dos seguintes valores: '1' = PNG '2' = JPG '3' = GIF
0xFF95	65429	Contrato possui Logon com Working Key Dinâmico e a chave não foi inicializada.
0xFF96	65430	É um erro genérico indicando que algum dado importante da rede não está presente no momento da transação.
0xFF97	65431	Pinpad não encontrado automaticamente
-	-	
-	-	
0xFF9A	65434	Erro ao consultar parâmetro online com a rede autorizadora
-	-	
0xFF9E	65438	Transação contactless não permitida devido a alteração do valor da compra
-	-	
0xFFA2	65442	Tamanho do BIN recuperado está com valor zero
0xFFA3	65443	Transação de consulta Bradesco não existe para o documento consultado
0xFFA4	65444	Transação com chip não contém TAGs EMVs obrigatórias
0xFFA5	65445	Dados incompletos, resposta contém apenas parte dos dados esperados
-	-	
0xFFFF	65535	Erro genérico
-	-5000	Erro desmontando o pacote recebido
-	-5001	Não achou a Master Key referente
-	-5002	Ocorreu timeout do pacote
	-5003	Erro configurando o registrador
-	-5004	Parâmetro lib incorreto
-	-5005	Erro Leitura Cartao/Trilha

PIN-Pad compartilhado

Para alguns casos, o SCOPE poderá retornar erros reportados pelo PIN-Pad compartilhado, cujos códigos se encontram descritos na tabela abaixo.

Códigos Retorno Hexa decimal	Decimal	Significado
0x0000	0	Operação OK
0x0001	1	Operação em processamento
0x0002	2	Notificar o operador com a mensagem recebida do PIN-Pad
0x0004	4	Pressionada a tecla de função #1
0x0005	5	Pressionada a tecla de função #2
0x0006	6	Pressionada a tecla de função #3
0x0007	7	Pressionada a tecla de função #4
0x0008	8	Pressionada a tecla "Limpar/Clear"
0x000A	10	Necessária operação previa
0x000B	11	Parâmetro inválido
0x000C	12	Time-out – esgotado o tempo de resposta
0x000D	13	Operação cancelada
0x000E	14	PIN-Pad já foi aberto
0x000F	15	PIN-Pad não foi aberto
0x0010	16	Erro interno de execução
0x0011	17	Função não suportada
0x0012	18	Função não disponível
0x0013	19	Ausência de dado mandatório para o processamento.
0x0014	20	Tabelas expiradas
0x0015	21	Erro ao tentar gravar tabelas (falta de espaço, por exemplo)
0x0016	22	Aplicação da rede adquirente não existe no PIN-Pad.
0x001E	30	Erro de comunicação: porta serial do PIN-Pad provavelmente ocupada.
0x001F	31	Erro de comunicação: PIN-Pad provavelmente desconectado ou problemas com a interface serial. Pode ser que a porta que está passando como parâmetro esteja errada.
0x0020	32	Status informado pelo PIN-Pad não é conhecido.
0x0021	33	Mensagem recebida do PIN-Pad possui formato inválido.
0x0022	34	Tempo esgotado ao esperar pela resposta do PIN-Pad (no caso de comandos não-blocantes).
0x0028	40	Erro interno do PIN-Pad.
0x0029	41	Erro de leitura do cartão magnético.
0x002A	42	Erro na captura do PIN (senha) - Master Key pode não estar presente.
0x002B	43	Não há cartão inteligente presente no acoplador. Cartão removido.
0x002C	44	PIN-Pad não pode processar a captura de PIN temporariamente devido a questões de segurança (como quando é atingido o limite de capturas dentro de um intervalo de tempo).
0x0032	50	Erro genérico no módulo SAM.
0x0033	51	SAM ausente, "mudo", ou com erro de comunicação.
0x0034	52	SAM inválido, desconhecido ou com problemas.
0x003C	60	Cartão não responde ("mudo") ou chip não presente.
0x003D	61	Erro de comunicação do PIN-Pad com o cartão inteligente.
0x003E	62	Cartão do tipo inválido ou desconhecido, não pode ser tratado (não é EMV nem TIBC v1).
0x003F	63	Cartão bloqueado por número excessivo de senhas incorretas (somente para Easy-Entry TIBC v1).
0x0040	64	Cartão TIBC v1 não autenticado pelo módulo SAM (somente para Easy-Entry TIBC v1).
0x0041	65	Cartão TIBC v1 expirado (somente para Easy-Entry TIBC v1).
0x0042	66	Cartão com erro de estrutura - arquivos estão faltando.
0x0043	67	Cartão foi invalidado. Se o cartão for TIBC v1, quando seleção de arquivo ou ATR retornar status '6284'. Se o cartão for EMV, quando seleção de aplicação retornar status '6a81'.

Códigos Retorno Hexa decimal	Decimal	Significado
0x0044	68	Cartão com problemas. Esse status é válido para muitas ocorrências no processamento de cartões TIBC v1 e EMV onde o cartão não se comporta conforme o esperado e a transação deve ser finalizada.
0x0045	69	O cartão, seja TIBC v1 ou EMV, comporta-se corretamente, porém possui dados inválidos ou inconsistentes.
0x0046	70	Cartão sem nenhuma aplicação disponível para as condições pedidas (ou cartão é reconhecido como TIBC v1 ou EMV, mas não possui nenhuma aplicação compatível com a requerida).
0x0047	71	Somente para cartão EMV. A aplicação selecionada não pode ser utilizada neste terminal, pois o "Get Processing Options" retornou status '6985'.
0x0048	72	Somente para aplicação de moedeiro. O saldo do moedeiro é insuficiente para a operação.
0x0049	73	Somente para aplicação de moedeiro. O limite máximo para a operação foi excedido.
0x004A	74	Cartão ainda não efetivo (data de ativação posterior à atual)
0x004B	75	Moeda inválida (cartão moedeiro)
0x004C	76	Erro de alto nível no cartão EMV que é passível de "fallback" para tarja magnética.
0x0050	80	Mais de um cartão sem contato foi apresentado ao leitor (este código de retorno é opcional e depende da capacidade do equipamento em detectar esta situação).
0x0051	81	Erro de comunicação entre o terminal (antena) e o cartão com chip sem contato.
0x0052	82	Cartão foi invalidado (seleção de aplicação retornou status '6ª81').
0x0053	83	Cartão com problemas. Esse status é válido para muitas ocorrências no processamento de cartões sem contato em que o cartão não se comporta conforme o esperado e a transação deve ser finalizada.
0x0054	84	Cartão sem nenhuma aplicação disponível para as condições pedidas (nenhum AID encontrado).
0x0055	85	A aplicação selecionada não pode ser utilizada (o Get Processing Options retornou status '6985' ou houve erro no comando Select final), e não há outra aplicação compatível na lista de candidatas.
0x00C8	200	Transação negada na função PP_GoOnChip()
0x00C9	201	Transação negada na função PP_GoOnChip()
0x00CA	202	Memória não alocada para a estrutura do PIN-Pad compartilhado
0x00CB	203	Erro alocando memória
0x00CC	204	Memória insuficiente para receber os dados
0x00CD	205	PIN-Pad já aberto via SCOPE
0x00CE	206	Não foi possível definir a Master Key a ser utilizada
0x00CF	207	Não foi possível definir o Estado de coleta no PIN-Pad
0x00D0	208	Erro no parâmetro da função GetPIN
0x00D1	209	PIN-Pad não configurado
0x00D2	210	Display não permitido neste momento ou situação
0x00D3	211	PIN-Pad não foi aberto pela aplicação
0x00D4	212	Time-out do cliente / usuário
0x00D5	213	Dado no chip não encontrado
0x00D6	214	Comanda Vazia
0x00D7	215	Comanda Inválida
0x00D8	216	As tabelas do PINPad estão vazias
0x00D9	217	Problema na carga de tabelas do PINPad
0x00DA	218	A operação não é permitida no modo, pois não está disponível
0x00DB	219	Tecla seta para cima
0x00DC	220	Tecla seta para baixo

Códigos de Retorno de Pré-TEF

Os códigos abaixo podem ser retornados pelo ScopeSRV ao ScopeAPI quando ocorre erro em uma Pré-TEF. Embora sejam erros internos ao SCOPE podem ser úteis para identificar e ajudar na análise de logs para identificação de problemas:

Código de Retorno (Decimal)	Descrição
00	Sucesso
01	Rede Offline
02	Erro na consulta ao BD
03	Timeout na consulta ao BD
04	BD não conectado
05	Transação não localizada no BD
06	BIN ou Serviço não configurado
07	PDV já logado
08	PDV não cadastrado
09	Taxa de serviço inválida
10	Taxa de serviço excede limite
11	Digitação não permitida
12	Protocolo não suportado
13	Não há PDVs disponíveis
14	Protocolo incompatível
15	Inserir cartão chip
16	BIN ou serviço não configurado pela rede adquirente
17	Conta não permitida
18	Serviço "invertido" (Crédito por Débito ou vice-versa)
19	Identificação de PDV inconsistente (recebeu um buffer inválido)
20	Erro na leitura do cartão
21	Dados não encontrados
22	Rede inválida ou não suportada pelo PINPad desse PDV
23	Dados de recarga não encontrados
24	Versão do BD incompatível
25	Chip não habilitado pela rede
26	Serviço não permitido para chip
27	Chip sem dados adicionais (pode indicar falha na carga de tabelas)
28	Cartão com Service Code inválido
29	Rede exige uso de PINPad compartilhado
30	PINPad compartilhado não opera com rede VISANET 2000.01
31	Estorno fora do prazo permitido
32	Estorno parcial não permitido (valor deve ser o total)
33	Estorno excede valor máximo permitido
34	Estorno com valor acima do total original não permitido
35	Estorno com valor acima do total original excede limite permitido
36	Estorno com valor zerado não permitido
37	Estorno com valor original zerado não permitido
38	Estorno já realizado
39	Cupom não encontrado
40	Prompts Adicionais não encontrados (pode indicar falha na carga de tabelas)
41	Contrato suspenso
42	Cartão inválido
43	Modo inválido
44	Leitor inativo
45	Erro nas chaves de criptografia
46	Erro na parametrização de trilhas da rede REDE (pode indicar falha na inicialização de tabelas)

47	Erro na leitura da tarja do cartão (Trilhas inconsistentes ou fora do padrão. Pode indicar falha de leitura, cartão corrompido ou fraudado)
48	Falha na configuração da confirmação positiva da rede adquirente GetNetLac
49	Transação por celular não habilitada no contrato
50	Transação por celular exige que a rede habilite transação digitada
51	A rede está em processo de inicialização. Tente novamente.
52	PINPad não suportado em modo “Multi-Bandeira Estendido”
53	Não representa erro. Uso interno em modo “Multi-Bandeira Estendido”.
54	Erro de leitura. PAN excede tamanho máximo.
55	Erro de leitura. Trilha1 excede tamanho máximo.
56	Erro de leitura. Trilha2 excede tamanho máximo.
57	Contactless não habilitado na carga de tabelas
58	Chip Contactless sem dados adicionais (pode indicar falha na carga de tabelas)
59	Contactless sem produto habilitado na carga de tabelas
60	Excede valor limite definido para Contactless
61	Servidor não disponível para receber conexões do PDV. Usado, por exemplo, para permitir a saída controlada do ScopeSRV através do botão “Bloquear PDVs”.
62	Fallback não permitido ou não habilitado (tabelas online) para o cartão e rede em questão.
63	Logon com a rede necessário para realizar TEF. Usado quando a rede trabalha com chave dinâmica, mas uma transação de logon entre o servidor e a rede em questão ainda não foi realizado com sucesso.
64	O cartão lido não é o mesmo usado na transação original. Usado em transações de confirmação/captura quando o cartão da transação original de pré-autorização é diferente.
65	Não necessariamente representa um erro. Usado pela função “Débito para Pagamento de Fatura” quando o cartão com chip lido, que está de acordo com a prioridade de débito, deve ser enviado para outra rede adquirente para efetuar a referida transação. Neste caso o Server está indicando ao Client que nova leitura deve ser efetuada, conforme parâmetros do “modo estendido”.
66	Usado pela função “Débito para Pagamento de Fatura” quando não houver nenhuma rede disponível para enviar a referida transação. Pode ser um problema momentâneo (rede offline) e/ou produto não configurado.
67	Valor 0 (zero) não é permitido quando a transação é realizada via contactless. Para a Rede GETNETLAC, essa condição vem indicada na tabela AID.

Formatos dos dados

Utilizado na coleta do dado, a aplicação valida a entrada de dado que o usuário forneceu de acordo com o código do formato do dado recebido.

Códigos Format Hexadeci mal	Formato Decimal	Significado
0x0000	00	String representando uma data no formato “DDMMMAA”
0x0001	01	String representando uma data no formato “DDMM”
0x0002	02	String representando uma data no formato “MMAA”
0x0003	03	String representando uma hora no formato “HHMMSS”
0x0004	04	String representando um número
0x0005	05	String representando uma senha que é numérica
0x0006	06	String representando um número com 4 dígitos
0x0007	07	String representando um dado alfanumérico

Códigos Format mal	Formato	Significado
Hexadeci mal	Decimal	
0x0008	08	String representando uma data no formato "DDMMAAAA"
0x0009	09	String de display para exibição (não há coleta)
0x0010	10	String representando uma data no formato "MMAAAA"
0x0011	11	Exibir '*' na tela, mas enviar em claro
0x0012	12	String representando uma hora no formato "HHMM"
0x0013	13	Booleano, a resposta deve ser 0=Não ou 1=Sim
0x0014	14	String representando "valor monetário" com tamanho total de 12, sendo os dois últimos dígitos os centavos
0x0015	15	String representando número não inteiro, com casas decimais
0x0016	16	Seleção de Opção (Menu)
0x0017	17	String representando o PAN do cartão
0x0018	18	Reservado para uso interno do SCOPE
0x0019	19	Formato desconhecido

LEMBRETE:utilizado apenas com a interface coleta.

Códigos das Teclas

Durante o processamento da transação, a aplicação deverá disponibilizar um meio que permita que usuário prossiga, retorne ou cancele o processamento. No entanto, de acordo com o momento do processamento, ou seja, conforme o estado da coleta de dados em que o SCOPE Client se encontra, nem sempre estas três opções estarão disponíveis. Para que a aplicação saiba qual(is) ação(ões) ela deve disponibilizar ao usuário, ela deve utilizar o membro HabTeclas da estrutura stPARAM_COLETA, após a chamada à função ScopeGetParam(), ou stPARAM_COLETA_EXT, após a chamada à função ScopegetParamExt(). O valor deste campo é a combinação binária (bitwise) de:

Códigos Format mal	Formato	Significado
Hexadeci mal	Decimal	
0x0001	01	Tecla cancela habilitada
0x0002	02	Tecla próximo habilitada
0x0004	04	Tecla retorna habilitada

Ele deverá ser verificado a cada iteração, logo após a chamada à função (ver [Obtendo os parâmetros da transação](#))

LEMBRETE:utilizado apenas com a interface coleta.

Códigos de Fluxo

Num determinado momento, coletado o dado ou não, a aplicação deverá informar ao SCOPE qual é ação que ele deverá tomar (ver [Passando o dado da coleta ao SCOPE Client](#)). Esta ação está associada às opções que o usuário pode tomar: avançar para o próximo estado, retornar para o estado anterior ou cancelar a transação. Cada ação tem um código amarrado a ela que se encontra na tabela que segue.

Códigos Formato		
Hexadecim al	Decimal	Significado
0x0000	00	Próximo estado
0x0001	01	Estado anterior
0x0002	02	Cancelar
0x0003	03	Erro na coleta de dados

LEMBRETE:utilizado apenas com a interface coleta.

Código das redes

Abaixo se encontram as redes suportadas pelo SCOPE e seus respectivos códigos.

Códigos Rede Decimal	Significado	Código SAT Decimal
0	SCOPE	999
1	Tecban	31
2	Itaú	999
3	Visanet – especificação 2000.01	999
4	Bradesco	999
5	REDE (Redecard)	25
6	Fininvest	999
7	Serasa	999
8	Teledata	999
9	Banrisul	999
10	Ticket	32
11	Associação Comercial de São Paulo - ACSP	999
12	BrasilCard (Antiga CNS)	999
13	Sysdata	999
14	REDE (Redecard) – especificação L0102	25
15	Visanet – especificação 1998.10	999
16	CBD	999
17	Lojista	999
18	CSU	999
19	Parati	999
20	Bem	999
21	Tokoró	999
22	MaxiCred	999
23	Zogbi	999
24	ACC Card	999
25	Sorocred	30
26	Coopercred	999
27	Telesp	999
28	Policard	23
29	Via Varejo	999
30	SAB	999
31	E-Pharma	999
32	Vidalink	999
33	Orizon	999
34	Hipercard	19
35	Interchange	999
36	Tecban – especificação 2.0	31

Códigos Rede Decimal	Significado	Código SAT Decimal
37	Nutricash	999
38	Losango	999
39	Sem uso	999
40	Cetelem	11
41	REDECOMPRAS	999
42	American Express (AMEX) – especificação 01	3
43	Banco do Brasil GCB	999
44	Sonae	999
45	Incomm (reutilização do código da rede BankBoston)	999
46	Portal Card	999
47	Valecard	999
48	TELENET-OBSOLETA	999
49	Evangélico	999
50	Funcional Card	18
51	Ediguay	999
52	CheckCheck	999
53	Banktec	999
54	Big	999
55	Big Card	8
56	SuperCard	999
57	Banese	999
58	TR Centre	999
59	TRN Card	999
60	Infocards	999
61	Valecash	999
62	Premium	999
63	CredSystem	999
64	REDE (Redecard) – especificação L02.05	25
65	E-Capture	999
66	Check Express	999
67	Conductor	999
68	ChequePre	999
69	Visanet 4.1	999
70	American Express – especificação 03.00	3
71	ECX Card	14
72	Ultragaz	999
73	GetNet	999
74	CentralCard	999
75	Orbitall	999
76	Recarga de Celular – especificação OKI/Itautec	999
77	IBI	999
78	DATASUS	999
79	Comprocards	999
80	Framaseg	999
81	Intellisys	999
82	Somar	999
83	Solocard	999
84	Oboé	999
85	Da Casa	999
86	Valeshop	999
87	Fidelize	999
88	UtilCard	999
89	RV Tec	999
90	GW Cel	999

Códigos Rede Decimal	Significado	Código SAT Decimal
91	U-Paid	999
92	REDE (Redecard) – especificação L0401	25
93	Banpará	999
94	Neus	999
95	CrediShop	999
96	Banco HSBC	999
97	Banco do Brasil ISO-GCB	999
98	Pharma Link	999
99	VEROBANRISUL	999
100	Banestes	999
102	Cielo	12
103	REDE (Redecard) – especificação L05.00	25
104	Diamante	999
105	RedeSoftnex	999
108	Getnetlac	999
109	Usecred	999
112	DM (anteriormente denominada DMCARD)	999
113	Siscred	999
114	Epay	999
115	Algorix	999
116	Orgcard	999
117	SAVS	999
118	DBR	999
121	RVTECNOLOGIA	999
123	GIVEX	999
124	CREDITITEM	999
125	MURY	999
126	BIN	999
127	TOPCARD	999
128	ELAVON	999
129	SMARTNET (VR)	31
133	PBM PADRAO	999
134	CONDUCTOR-PL – especificação V02.05	999
135	Stone	999
139	SGF	999
140	Safra	33
141	Adianti	999
142	GetCard	999
143	UNISYS	999
144	ITSPAY	999
145	VINDI	999
146	TELENET	999
149	GPB (Global Payments Brasil)	999
150	MAXISCARD	999
152	COOPERCARD	999
153	BRADESCARD	999
154	MERCADO PAGO	
156	BANCO INTER	
157	TICKETLOG	999
158	TODO-CARTÕES	999
159	PAGSEGURO	
160	PAGBANK	
161	CONNECT	999
163	AME DIGITAL	

Códigos Rede Decimal	Significado	Código SAT Decimal
164	PICPAY	
165	RAPPI	
166	BANCO DO BRASIL (PIX)	
167	BRADESCO-PIX	
170	SEICON	999
173	ALGORIX-EMV	999
174	BKBANK	999
175	BSCASH-EMV	999
176	SOLUCARD	999
177	STIX	

Código de especificação das redes

Abaixo se encontram as redes que possuem mais que uma especificação suportada pelo SCOPE.

Código Rede	Código Especificação	Significado	
13	1	SYSDATA	V1.XX
	2		V2.00
25	0	SOROCRED	V01
	1		V13
28	0	POLICARD	V1.05
	1		V3.00 - EMV
31	0	EPHARMA	V4.0.0
	1		V5.1.0
41	0	REDECOMPRAS	V1.00
	1		V3.01
57	1	BANESE	V8.04 e inferior (magnético)
	2		V9.07 – EMV
66	1	CHECKEXPRESS	V1.XX
	2		V2.00
63	0	CREDSYSTEM	V1.14
	1		V2.50
	2		V2.60
71	0	ECXCARD	V2.50
	1		V3.02
73	1	GETNET	V01.00 (telefone 8 dígitos)
	2		V06.xx (telefone 9 dígitos)
	3		V07.xx EMV (9 dígitos)
77	1	IBI	V1.XX
	2		XSAB 1.28a
86	0	VALESHOP	V1.00
	1		V8.00 EMV
89	2	RVTEC	V2.00
90	1	GWCEL	V0003
	2		V0005
91	1	UPAID	V1.XX
	2		V2.00
95	0	CREDI-SHOP	V2.05
	1		V3.03
99	0	VEROBANRISUL	R01
	1		R17
	2		R18 (R17 com Subadquirência)
102	0	CIELO	4.1.0
	1		4.1R2014
	2		4.1R2014b (Circular Ticket)
	3		4.1R2014c (Debit Mastercard)
103	1	REDE/REDECARD	V5.00
	2		V5.01
	3		V5.02
	5		V6.01
	6		V6.02
	7		V6.03
	8		V6.03b (Debit Mastercard)
	9		V7.01
105	0	REDESOFNTEX	V2.05.004

Código Rede	Código Especificação	Significado	
	1		V3.03.002 EMV
108	0	GETNETLAC	V2.10
	1		V2.90
	2		V2.92
	3		V2.94
	4		V2.96
	5		V2.99 (Revisão A)
109	0	USECRED	V2.05
	1		V3.03
112	0	DM	V2.05
	1		V3.03
114	0	EPAY	V1.07
	1		V1.18
118	1	DBR	V1.XX
	2		V2.00
125	0	MURY	V2.05
	1		V3.03
129	0	SMARTNET	V2.00.008 - EMV
	1		V2.00.008 - EMV e ABECS
135	0	STONE	V1.06
	1		V2.17
	2		V2.17.1
	3		V4.00
139	1	SGF	V1.00 ed.009
143	1	UNISYS	V1.1
140	0	SAFRA	V1.05
	1		V1.13
144	1	ITSPAY	V3.01
	2		V3.02 - EMV
146	0	TELENET	V3.02
149	0	GPB	V1.03
150	0	MAXISCARD	V3.02
152	0	COOPERCARD	V3.02
153	0	BRADESCARD	V0516
	1		V0518
	2		V06.02
157	0	TICKETLOG	V1.19
158	0	TODO-CARTÕES	V3.02
159	0	PAGSEGURO	V1.1.1
	1		V2.0.4
161	0	CONNECT	V3.02
	1		V3.03
170	0	SEICON	V3.03
173	0	ALGORIX-EMV	V3.03
174	0	BKBANK	V3.03
175	0	BSCASH-EMV	V3.03
176	0	SOLUCARD	V3.03
177	0	STIX	SNC V2.02

Código das bandeiras

Como algumas transações exigem, o operador deve escolher uma bandeira e passar para o SCOPE o seu código cuja relação se encontra abaixo. Para que o operador possa escolher, é necessário que ele tenha uma lista com os códigos de cada bandeira. Uma opção é ter uma lista impressa disponível para o operador. No entanto, como geralmente nem todas as bandeiras são utilizadas para uma determinada loja, pode ser viável a aplicação, por meio de configuração, exibir na tela uma lista de opções, somente com as bandeiras utilizadas, no momento que for necessário.

Códigos Bandeira		Significado
Hexadecim al	Decimal	
0x0000	000	SCOPE
0x0001	001	Visa
0x0002	002	Mastercard
0x0003	003	Amex
0x0004	004	Lojista
0x0005	005	Diners
0x0006	006	Sollo
0x0007	007	Cheque Eletrônico
0x0008	008	Mastercard Maestro (conhecido também como Redeshop)
0x0009	009	Itaú
0x000A	010	Bradesco
0x000B	011	Fidelidade
0x000C	012	Serasa
0x000D	013	Telecheque
0x000E	014	Sodexo Alimentacao
0x000F	015	BrasilCard (antiga RVA)
0x0010	016	Ticket Alimentação
0x0011	017	Hipercard
0x0012	018	CardCo (antiga CNS)
0x0013	019	MaxiCred
0x0014	020	Banrisul
0x0015	021	Visa Electron
0x0016	022	REDE (Redecard) (bandeira para operações internas, independentes de cartão. Exemplo: Pagamento de Contas, Resumo de Vendas)
0x0017	023	JCB
0x0018	024	Quality Card
0x0019	025	Unnisa
0x001A	026	Fininvest
0x001B	027	Multi-Cheque
0x001C	028	VR (cartão voucher da CSU)
0x001D	029	TransCheck
0x001E	030	TecBan (bandeira genérica para serviços TECBAN)
0x001F	031	ACC Card
0x0020	032	Sorocred
0x0021	033	Parati
0x0022	034	BEM – Banco do Estado do Maranhão
0x0023	035	Tokoro

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x0024	036	Zogbi
0x0025	037	TopPremium
0x0026	038	Carrefour (cartão voucher da CooperCred)
0x0027	039	Rainbow (cartão voucher da TecBan)
0x0028	040	Telesp Celular
0x0029	041	Policard
0x002A	042	Via Financeira
0x002B	043	IBI
0x002C	044	e-Pharma (PBM)
0x002D	045	Vidalink (PBM)
0x002E	046	Orizon (PBM)
0x002F	047	e-Pharma balcão (solicitação de autorização via conexão direta com ScopeGW.)
0x0030	048	Aura
0x0031	049	Correspondente bancário – Unibanco
0x0032	050	Correspondente bancário – Citibank
0x0033	051	Recebimento Fininvest
0x0034	052	Multi-Benefícios
0x0035	053	VA Eletrônico (cartão alimentação)
0x0036	054	Valetik
0x0037	055	Losango
0x0038	056	Goodcard
0x0039	057	FIC
0x003A	058	Banestik (cartão voucher da TecBan)
0x003B	059	Cabal
0x003C	060	TMS
0x003D	061	Bonus (cartão voucher)
0x003E	062	Visa Vale
0x003F	063	Correspondente bancário – Banco do Brasil
0x0040	064	Colaborador (cartão de crédito do SONAE)
0x0041	065	Incomm (reutilização do código da bandeira CB-BankBoston)
0x0042	066	Portal Card
0x0043	067	Vale Card
0x0044	068	PERSONAL CARD CRED
0x0045	069	Green Card (não mais utilizada)
0x0046	070	AsCard
0x0047	071	Evangelico
0x0048	072	Funcional card (PBM)
0x0049	073	ACSP – Associação Comercial de São Paulo
0x004A	074	ExtraBom/ABN
0x004B	075	Ediguay
0x004C	076	GoodMed (PBM)
0x004D	077	Bônus Eletrônico (cartão voucher do SONAE)
0x004E	078	RefeiSul
0x004F	079	Multi-Alimentação
0x0050	080	Multi-Cheque (Novo)
0x0051	081	CheckCheck
0x0052	082	EcxCard
0x0053	083	BigCard
0x0054	084	SuperCard
0x0055	085	Banese Crédito – Banco do Estado de Sergipe
0x0056	086	Novartis (PBM)
0x0057	087	FlexMed (PBM)

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x0058	088	TrnCentre – Transaction Centre
0x0059	089	InfoCards
0x005A	090	BaseCard
0x005B	091	SysData
0x005C	092	Correspondente bancário – Bradesco
0x005D	093	Ticket Restaurante
0x005E	094	ValeCash
0x005F	095	BankCard
0x0060	096	MedCheque
0x0061	097	Premium (cartão de crédito do SONAE)
0x0062	098	Vale Gás
0x0063	099	Datasus (PBM)
0x0064	100	BankTec
0x0065	101	PoupCard
0x0066	102	ChequePre
0x0067	103	Banquet (cartão voucher alimentação)
0x0068	104	Private Label da REDE
0x0069	105	Cielo
0x006A	106	Rancho Convênio – PortalCard
0x006B	107	Rancho Alimentação – PortalCard
0x006C	108	PlanVale
0x006D	109	IBICard
0x006E	110	IBI PL,
0x006F	111	Pague-Conta Visanet
0x0070	112	Cartão Fácil (EPA)
0x0071	113	Cartão Fácil Losango (EPA)
0x0072	114	CPF – IBI PL deposito CDB
0x0073	115	ComproCard
0x0074	116	FarmaSeg (PBM)
0x0075	117	Unik
0x0076	118	Verde Card
0x0077	119	Rossi
0x0078	120	Somar
0x0079	121	Solocard
0x007A	122	Banquet Smart
0x007B	123	VEGAS CARD CRED
0x007C	124	Planvale REDE
0x007D	125	Oboé Card
0x007E	126	DaCasa
0x007F	127	CABAL Vale (TECBAN e GETNET 6.0)
0x0080	128	CABAL Débito (TECBAN e GETNET 6.0)
0x0081	129	BRTelecom Telefonia FIXA
0x0082	130	Brasil Telecom
0x0083	131	GoodVale (GETNET 6.0 Voucher)
0x0084	132	RedeSoftnex – Com senha
0x0085	133	PratiCard
0x0086	134	PrestaServ
0x0087	135	Safra Amanco
0x0088	136	SimCred
0x0089	137	Premiação SONAE
0x008A	138	Presente SONAE
0x008B	139	ValeShop
0x008C	140	ZUUM

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x008D	141	FAI/Orbital
0x008E	142	ConvCard
0x008F	143	Operadora Claro
0x0090	144	Operadora OI
0x0091	145	Operadora TIM
0x0092	146	Operadora Telemig
0x0093	147	Operadora Amazônia Celular
0x0094	148	Operadora Embratel
0x0095	149	Operadora Telefonica
0x0096	150	Operadora TELEMAR
0x0097	151	Operadora CTBC-Celular
0x0098	152	Operadora CTBC Fixo
0x0099	153	Operadora SERCOMTEL Celular
0x009A	154	Operadora SERCOMTEL Fixo
0x009B	155	Operadora Telefonica Família
0x009C	156	Operadora NEXTEL
0x009D	157	Operadora VIVO
0x009E	158	Cartão Parcele Mais (REDE L0401)
0x009F	159	BANPARA
0x00A0	160	NEUS
0x00A1	161	BANCRED SENHA
0x00A2	162	Credi-Shop
0x00A3	163	Sapore
0x00A4	164	CB-HSBC
0x00A5	165	Verocheque
0x00A6	166	Getnet
0x00A7	167	Pharmalink PBM
0x00A8	168	Aura FNAC
0x00A9	169	Aura BERGAMAIS
0x00AA	170	Com Você
0x00AB	171	BANESCARD Crédito
0x00AC	172	Nokia
0x00AD	173	Fala Fácil (Nexus)
0x00B0	176	BRASIL CONVENIOS
0x00B1	177	AMCARD
0x00B2	178	Rede MED
0x00B3	179	GPA
0x00B4	180	PAT – sem senha
0x00B5	181	PAT – com senha
0x00B6	182	FAN CARD
0x00B7	183	BANRICOMPRAIS
0x00B9	185	Banrisul – Refeisul Alimentação / Refeição
0x00BA	186	Banrisul – Refeisul Combustível
0x00BB	187	TIM ON-LINE
0x00BC	188	BNB CLUBE VOUCHER
0x00BD	189	NEUS Senha
0x00BE	190	Diamante
0x00BF	191	MinasCred
0x00C0	192	Nutricash
0x00C2	194	ELO Debito
0x00C3	195	ELO Credito
0x00C4	196	Goodcard Senha
0x00C7	199	USECRED CRED

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x00C8	200	USECRED com senha (descontinuada para espec. v3.03 ou sup.)
0x00C9	201	SOROCRED Crédito
0x00CA	202	SICREDI Crédito
0x00CB	203	SICREDI Débito
0x00CC	204	BrasilCard – com senha
0x00CF	207	PL GETNET VISA
0x00D0	208	DMCARD
0x00D1	209	PLANVALE com senha
0x00D2	210	SISCRED sem senha
0x00D3	211	SISCRED com senha
0x00D4	212	DMCARD com senha
0x00D5	213	RedeSoftnex – Convênio
0x00D6	214	RedeSoftnex
0x00D7	215	Claro Off-line
0x00D8	216	Oi Fixa
0x00D9	217	Oi Off-line
0x00DA	218	Vivo off-line
0x00DB	219	Telesp Super 15
0x00DC	220	Embratel Livre Online
0x00DD	221	Epay
0x00DE	222	ALGORIX CRED
0x00DF	223	ALGORIX_S
0x00EO	224	Orgcard débito
0x00E1	225	Orgcard crédito
0x00E2	226	Orgcard crédito DV,CS
0x00E4	228	Ticket Car
0x00E5	229	PL GETNET MASTER
0x00E6	230	SAVS
0x00E9	233	POLICARD SENHA
0x00F1	241	GIVEX
0x00F2	242	CREDITITEM SENHA
0x00F4	244	MURY
0x00F5	245	ABRAPETITE
0x00F8	248	Sodexo Refeicao
0x00FA	250	TOPCARD com senha
0x00FB	251	TOPCARD sem senha
0x00FC	252	Elavon
0x00FD	253	VR ALIMENTACAO
0x00FE	254	VR AUTO
0x00FF	255	VR CULTURA
0x0100	256	VR REFEICAO
0x0101	257	MULTIBENEFICIO
0x0102	258	MULTIALIMENTACAO BEM
0x0103	259	MULTICESTABASICA
0x0104	260	CARTAO MAMAE
0x0105	261	MULTICASH
0x0106	262	CARTAO BRINQUEDO
0x0107	263	CARTAO NATAL
0x0108	264	MULTICHEQUE BEM
0x0109	265	MULTIEMPRESARIAL
0x010A	266	MULTICOMBUSTIVEL
0x010B	267	MULTICULTURA
0x010C	268	MULTIFARMA

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x010D	269	MULTIREFEICAO
0x0112	274	FORTBRASIL
0x0116	278	ELO AUTO
0x0125	293	BAHAMAS CRÉDITO
0x0126	294	BAHAMAS ALIMENTAÇÃO
0x0127	295	BANESCARD Débito
0x012C	300	ValeCard Deb
0x0131	305	ALGORIX VOUCHER
0x0132	306	Banese Débito – Banco do Estado de Sergipe
0x0136	310	MULTIPLUS
0x0137	311	POLICARD CREDITO
0x0138	312	POLICARD DEBITO
0x0139	313	POLICARD ALIMENTAÇÃO
0x013A	314	POLICARD REFEIÇÃO
0x013B	315	GOODCARD ALIMENTAÇÃO
0x013C	316	GOODCARD REFEIÇÃO
0x013F	319	GETNET CRÉDITO
0x0140	320	GETNET DÉBITO
0x0141	321	GETNET VOUCHER
0x0142	322	VALECARD CULTURA
0x0143	323	VALECARD ALIMENTACAO
0x0144	324	VALECARD REFEICAO
0x0145	325	VALECARD COMBUSTIVEL
0x0146	326	VALECARD VOUCHER
0x0147	327	ADIANTI CRED
0x0148	328	ADIANTI DEB
0x0149	329	ACENTO
0x014A	330	GETCARD DEB
0x014B	331	GETCARD VOUCHER
0x014C	332	ECO FROTAS
0x014D	333	ZAFFARI
0x014E	334	GETCARD CRÉD
0x014F	335	SANTANDER
0x0150	336	ITSPAY DEB
0x0152	338	BIQ VOUCHER
0x0153	339	VEGAS CARD VOUCHER
0x0154	340	REDECOMPRAS CRED
0x0155	341	CETELEM
0x0156	342	TICKET CULTURA
0x0157	343	SOROCRED DEBITO
0x0158	344	CREDSYSTEM PL
0x0159	345	MAIS CREDITO
0x015A	346	VINDI
0x015B	347	ITSPAY
0x015C	348	BAHAMAS REFEIÇÃO
0x015D	349	VEGAS CARD ALIMENTACAO
0x015E	350	VALE REAL
0x015F	351	BANESE ALIMENTACAO
0x0160	352	PERSONAL CARD VOUCHER
0x0161	353	BNB CLUBE CRED
0x0162	354	CABAL REFEICAO
0x0163	355	CABAL ALIMENTACAO
0x0164	356	GREENCARD REFEICAO

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x0165	357	GREENCARD ALIMENTACAO
0x0166	358	VEROCHEQUE REFEICAO
0x0167	359	VEROCHEQUE ALIMENTACAO
0x0168	360	VEROCHEQUE CULTURA
0x0169	361	VEROCHEQUE FROTA
0x016A	362	NUTRICASH FROTA
0x016B	363	UP VOUCHER
0x016C	364	UP FROTA
0x016D	365	UP CULTURA
0x016E	366	PERSONALCARD DEB
0x016F	367	BNBCLUBE DEB
0x0170	368	TELENET VOUCHER
0x0171	369	SENFF ALIMENTACAO
0x0172	370	SENFF REFEICAO
0x0173	371	SENFF VOUCHER
0x0174	372	COOPER CREDITO
0x0175	373	COOPER ALIMENTACAO
0x0176	374	COOPER REFEICAO
0x0177	375	COOPER VOUCHER
0x0178	376	SOROCRED PL
0x0179	377	SOROCRED INST
0x017C	380	BANRISUL VERO
0x017D	381	VALESHOP ALIMENTACAO
0x017E	382	VALESHOP REFEICAO
0x017F	383	VALESHOP BENEFICIO
0x0180	384	VALESHOP COMBUSTIVEL
0x0181	385	VALESHOP DEBITO
0x0182	386	VALESHOP CREDITO
0x0183	387	SODEXO CULTURA
0x0184	388	SODEXO PREMIUM PASS
0x0185	389	SODEXO COMBUSTIVEL PASS
0x0186	390	SODEXO GIFT PASS
0x0187	391	BENEFÍCIOS VISA VALE REFEIÇÃO
0x0188	392	BENEFÍCIOS VISA VALE ALIMENTAÇÃO
0x018A	394	MAXISCARD CRED
0x018D	397	COOPERCARD CRED
0x018E	398	COOPERCARD ALIM
0x018F	399	COOPERCARD REF
0x0190	400	COOPERCARD BEM
0x0191	401	MERCADO PAGO
0x0192	402	BRADESCARD CREDITO
0x019C	412	TICKET LOG CREDITO
0x019D	413	TICKET LOG DEBITO
0x019E	414	TICKET LOG FROTA
0x019F	415	ECXCARD CRED
0x01A0	416	ECXCARD COMB
0x01A1	417	TODO VOUCHER
0x01A3	419	INVESTCRED
0x01A4	420	MAXXCARD DEB
0x01A5	421	MAXXCARD REFEICAO
0x01A6	422	MAXXCARD ALIMENTACAO
0x01A7	423	MAXXCARD AUTO
0x01A8	424	MAXXCARD CULTURA

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x01A9	425	MAXXCARD BENEFICIO
0x01AA	426	REDECOMPRAS DEB
0x01AB	427	REDECOMPRAS REFEICAO
0x01AC	428	REDECOMPRAS ALIMENTACAO
0x01AD	429	REDECOMPRAS AUTO
0x01AE	430	REDECOMPRAS CULTURA
0x01AF	431	REDECOMPRAS BENEFICIO
0x01B0	432	PIX
0x01B1	433	PAGBANK
0x01C2	450	BAHAMAS ELO CRED
0x01CA	458	CONNECT CRED
0x01CB	459	AME DIGITAL
0x01CC	460	PICPAY
0x01D2	466	RAPPI
0x01D8	472	ALELO CULTURA
0x01D9	473	CABAL CULTURA
0x01DA	474	VR BENEFICIO
0x01DB	475	COOPER CULTURA
0x01DC	476	COOPER GIFT
0x01DD	477	GREENCARD BENEFICIO
0x01DE	478	SENFF AUTO
0x01DF	479	SENFF BENEFICIO
0x01EO	480	UP BRASIL DEBITO
0x01E1	481	UP BRASIL AUTO
0x01E2	482	UP BRASIL BENEFICIO
0x01E3	483	UP BRASIL PRESENTE
0x01E4	484	VALECARD BENEFICIO
0x01E5	485	VALECARD AUTO
0x01E6	486	VEROCARD BENEFICIO
0x01E7	487	VEROCARD AUTO
0x01E8	488	LECARD VOUCHER
0x01E9	489	LECARD CRED
0x01EA	490	REDESOFTNEX DEB
0x01EB	491	REDESOFTNEX VOUCHER
0x01EC	492	VEGAS CARD DEB
0x01ED	493	ABRAPETITE VOUCHER
0x01EE	494	ROMCARD CRED
0x01EF	495	ROMCARD DEB
0x01FO	496	CEAPAY
0x01F7	503	SEICON CRED
0x01F8	504	SEICON VOUCHER
0x0204	516	UZE CRED
0x0205	517	UZE VOUCHER
0x020E	526	BKBANK BEN
0x0210	528	BSCASH CRED
0x0211	529	STIX
0x0215	533	SOLUCARD CRED
0x0216	534	SOLUCARD BEN
0x0217	535	MURY CRED
0x0218	536	MURY VOUCHER
0x0219	537	DINERS DEB
0x021A	538	USECRED ALIM
0x021B	539	USECRED REF

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x021C	540	USECRED COMB
0x021D	541	USECRED BEN
0x21E	542	LIVELO
0x021F	543	BL-BRADESCO
0x0220	544	MEGAVALECARD CREDITO
0x0221	545	ELO VOUCHER PAT
0x0222	546	MEGAVALECARD DEBITO

Dados disponíveis das transações

Abaixo está a relação dos campos disponíveis que podem ser obtidos das transações pelas funções [ScopeObtemCampoExt\(\)](#) e [ScopeObtemCampoExt2\(\)](#).

	CAMPO DE DADOS	VALOR DO BIT
Máscara 1	PAN – Personal Account Number (número do cartão)	0x00000001
	Valor da transação	0x00000002
	NSU (Número Sequencial Único) da transação	0x00000004
	Hora local da transação	0x00000008
	Data local da transação	0x00000010
	Data de validade do cartão	0x00000020
	Data contábil da transação	0x00000040
	Número do cheque	0x00000080
	Código de autorização	0x00000100
	Código de resposta	0x00000200
	Identificação do terminal	0x00000400
	Código do estabelecimento (contrato)	0x00000800
	Número de parcelas	0x00001000
	Taxa de serviço (gorjeta)	0x00002000
	NSU do Host	0x00004000
	Número do banco	0x00008000
	Número da agência	0x00010000
	Nota: Para obter o "Num. Agencia Receptora" no caso de Correspondente Bancário Banco do Brasil, utilizar o campo de dados "Código do Estabelecimento(contrato)", extraíndo a partir da 5º posição com tamanho 4.	
	Data de agendamento	0x00020000
	Código da bandeira	0x00040000
	Código do Serviço (*)	0x00080000
	Conteúdo do BIT 62	0x00100000
	Número do controle	0x00200000
	Código de rede	0x00400000
	Nome da bandeira	0x00800000
	Nome da Rede	0x01000000
	Trilha 02 do cartão	0x02000000
	Número de notas promissórias	0x04000000
	Código de estabelecimento Visanet	0x08000000

	Código CMC7	0x10000000
	CGC do convênio PBMS	0x20000000
	Mensagem de autenticação do cheque	0x40000000
	Saldo disponível (cartão convênio)	0x80000000
Máscara 2	NSU da transação original	0x00000001
	Cliente aderente ao seguro (IBICred)	0x00000002
	Dados do parcelado da rede Cetelem	0x00000004
	Data do movimento (Interchange, BBGCB)	0x00000008
	Nome do cedente ou empresa de convênio (Interchange, BBGCB)	0x00000010
	Lista das formas de pagamento em TEF permitidas (Interchange)	0x00000020
	Linha de autenticação (Interchange – Fininvest)	0x00000040
	Dados da consulta de fatura (Interchange – Fininvest) (Vide ScopeApi.h)	0x00000080
	Formas de financiamento (A: Administradora – E: Estabelecimento)	0x00000100
	Código específico da consulta AVS	0x00000200
	Pontos adquiridos ou resgatados	0x00000400
	Fator de compra	0x00000800
	NSU do Host da transação original (estornada)	0x00010000
	Identificação do cliente PBM junto à autorizadora (apenas Vidalink)	0x00020000
	Código da operadora de celular	0x00040000
	Código de área (DDD)	0x00080000
	Número do telefone	0x00100000
	ULTRAGAZ: dados do ValeGás	0x00200000
	Código IF (Instituição Financeira)	0x00400000
	Número do item da Fininvest ou Cetelem, ou número do contrato (CPCHEQUE/INSS) do IBI	0x00800000
	Valor da taxa de embarque	0x00100000
	Uso exclusivo do SONAE	0x00200000
	Informação contida no bit 124 - CDC Orbitall	0x00400000
	Código de serviço da transação original (estorno)	0x00800000
	Código de barras	0x01000000
	Permite desfazimento	0x02000000
	Logo do PAN	0x04000000
	Código da Empresa	0x08000000
	Código de Autenticação	0x10000000
Máscara 3	Dados do pagamento	0x20000000
	UsoRes_63	0x40000000
	Número do PDV	0x80000000
	Informações sobre a quantidade e os e-cupons disponíveis ao cliente	0x00000001
	Informação do desconto do resgate monetário	0x00000002
	Informações sobre a coleta de dados realizada na transação (BIT 48).	0x00000004
	Modo de Entrada (Entry Mode)	0x00000008
	Valor do Saque	0x00000010
	Resposta da consulta Infocards (bit 62 da 0110)	0x00000020
	Dados da resposta de Consulta da EPAY. Os dados retornados consistem em 3 valores de concatenados: 1. Valor Mínimo (12 dígitos) 2. Valor Máximo (12 dígitos) 3. Saldo Disponível (12 dígitos)	0x00000040

	Dados da resposta de Consulta valor Gift Card (INCOMM) Os dados retornados consistem em 3 valores de concatenados: 1. Valor Mínimo (12 dígitos) 2. Valor Máximo (12 dígitos) 3. Saldo Disponível (12 dígitos)	0x00000040
	Máximo de mercadorias permitidas para uma transação (Cartão TicketCar, Valecard, entre outras) O dado retornado é um campo de 2 dígitos.	0x00000100
	Código SAT (ver códigos em Código das redes)	0x00000200
	Versão corrente de Carga de Tabelas do Host Formato: 10 dígitos (Preenchido com zeros a esquerda, caso necessário). Disponível em transações com as seguintes Redes: • SAVS	0x00000400
	CNPJ da rede credenciadora - SAT	0x00002000
	Dados do Correspondente Bancário	0x00008000
	Dados Adicionais Gift Card: • Para Incomm - código UPC11, • Para BlackHawk - código EAN12, • Para a EPAY - código do produto com 8 dígitos, com brancos à direita quando o campo for menor.	0x0010000
	Dados retornados da Operação Fidelidade (SGF)	0x00020000
	Valor Total do Pagamento	0x00040000
	Valor de Descontos do Pagamento	0x00080000
	Valor de Entrada (IATA)	0x00800000
	Valor de Acréscimos do Pagamento	0x00100000
	Dados do Perfil de Pagamento Recorrente	0x01000000
	Dados da Assinatura do Pagamento Recorrente	0x02000000
	Dados Consulta BACEN – para títulos registrados	0x04000000
	Valor Documento	0x08000000
	Resposta Consulta BACEN – comprovante	0x10000000
	Modo Pagamento: '00' – não informado '01': cheque (Utilizado para pagamento de contas) '02': dinheiro (Utilizado para pagamento de contas) '03': "debito em conta" (Utilizado para carteiras virtuais e/ou pagamento de contas) '04': "cartao credito" (Utilizado para carteiras virtuais) '05': "pix" (Utilizado para carteiras virtuais) '06': "cartao de debito" (Utilizado para carteiras virtuais) '07': "saldo + cartao" (Utilizado para carteiras virtuais)	0x20000000
	Consulta Cedente - Dados da Consulta BACEN BRADESCO	0x40000000
	Data Vencimento CORBAN – do título/conta	0x80000000
Máscara 4	Nome do Portador do Cartão (Informação com até 26 caracteres)	0x00000001
	Data de Validade do Cartão (YYMMDD)	0x00000002
	Merchant ID (informação com até 32 caracteres)	0x00000004
	Código do Estabelecimento Externo (informação com até 15 caracteres)	0x00000008
	String para gerar o QRCode	0x00000020
	Relação de Descontos por Item, recebidos da Ticket Log no bit 54 da 0210	0x00000080
	Informa se o Saldo_Disponivel está em [0] = Reais (default) ou [1] = Litros	0x00000100
	Número do CPF	0x00000200
	ARQC do chip, se disponibilizado pelo cartão	0x00000400
	AID do chip, se disponibilizado pelo cartão	0x00000800

	Indicação se a transação foi autorizada mediante uso de senha pessoal [1] = Sim, [0] = Não	0x00001000																		
	campo TID da tabela Mensagem (do Banco de Dados) - transacoes pix (txid)	0x00002000																		
	campo Referencia da tabela Mensagem (do Banco de Dados) - transacoes pix (end2endld)	0x00004000																		
	Tamanho do BIN	0x00008000																		
	Estrutura de strings finalizadas com null, conforme abaixo:	0x00010000																		
	<table border="1"> <thead> <tr> <th>Dado</th> <th>Tamanho + null</th> <th>Obs</th> </tr> </thead> <tbody> <tr> <td>Valor Convertido</td> <td>12 + 1</td> <td></td> </tr> <tr> <td>Cotação de Conversão</td> <td>8 + 1</td> <td></td> </tr> <tr> <td>Taxa Markup</td> <td>5 + 1</td> <td>%</td> </tr> <tr> <td>Sigla da Moeda Estrangeira</td> <td>3 + 1</td> <td>ISO 4217</td> </tr> <tr> <td>Código da Moeda Estrangeira</td> <td>3 + 1</td> <td>ISO 4217</td> </tr> </tbody> </table>	Dado	Tamanho + null	Obs	Valor Convertido	12 + 1		Cotação de Conversão	8 + 1		Taxa Markup	5 + 1	%	Sigla da Moeda Estrangeira	3 + 1	ISO 4217	Código da Moeda Estrangeira	3 + 1	ISO 4217	0x00020000
Dado	Tamanho + null	Obs																		
Valor Convertido	12 + 1																			
Cotação de Conversão	8 + 1																			
Taxa Markup	5 + 1	%																		
Sigla da Moeda Estrangeira	3 + 1	ISO 4217																		
Código da Moeda Estrangeira	3 + 1	ISO 4217																		
	Pode ser usada a estrutura stDadosDCC definida em scopeapi.h																			
	Status DCC: '0' = Não Realizado '1' = Cliente Não Aceitou '2' = Cliente Aceitou '3' = Não Elegível '4' = Erro de Comunicação Qualquer outro valor = Desconhecido	0x00040000																		
	Dados de Consulta de Pré-Autorização Estrutura de strings finalizadas com null, conforme abaixo:	0x00080000																		
	<table border="1"> <thead> <tr> <th>Dado</th> <th>Tamanho + null</th> </tr> </thead> <tbody> <tr> <td>Status indicativo da Pré-autorização</td> <td>2 + 1</td> </tr> <tr> <td>Data da Pré-Autorização</td> <td>8 + 1</td> </tr> <tr> <td>Data da Expiração da Pré-Autorização</td> <td>8 + 1</td> </tr> <tr> <td>NSU do Host</td> <td>12 + 1</td> </tr> <tr> <td>Valor da transação</td> <td>12 + 1</td> </tr> </tbody> </table>	Dado	Tamanho + null	Status indicativo da Pré-autorização	2 + 1	Data da Pré-Autorização	8 + 1	Data da Expiração da Pré-Autorização	8 + 1	NSU do Host	12 + 1	Valor da transação	12 + 1							
Dado	Tamanho + null																			
Status indicativo da Pré-autorização	2 + 1																			
Data da Pré-Autorização	8 + 1																			
Data da Expiração da Pré-Autorização	8 + 1																			
NSU do Host	12 + 1																			
Valor da transação	12 + 1																			
	Status indicativo da Pré-autorização pode ser: '00' - Consulta não retornou dados válidos '01' - Pré-Autorização pendente '99' - Pré-autorização já cancelada ou inexistente Se diferente de '01' os demais dados serão preenchidos com zeros.																			
	Pode ser usada a estrutura stDadosConsultaPreAut definida em scopeapi.h.																			
	-	-																		
	Código do Produto Digital (formato EAN13 ou UPC12) Pode ser obtido ao final da transação de Consulta Conteúdo Digital Pode ser usada a constante TAM_COD_PROD_DIGITAL para o tamanho da variável. Exemplo: char szCodProdDigital[TAM_COD_PROD_DIGITAL + 1];	0x00800000																		

Grupo de Serviços

Nesta tabela encontram-se os grupos de serviços que o SCOPE trabalha.

Códigos Bandeira		Significado
Hexadecimal	Decimal	
0x0001	001	Cartão de débito
0x0002	002	Cartão de crédito
0x0003	003	Consulta a cheques
0x0004	004	Controle SCOPE/Autorizador
0x0005	005	Controle API/Server
0x0006	006	PAT
0x0007	007	CDC
0x0008	008	Garantia de desconto de cheques
0x0009	009	Resumo de vendas
0x000A	010	Uso futuro
0x000C	012	Reimpressão de comprovante
0x000E	014	Fidelidade
0x000F	015	Recarga de celular
0x0010	016	Transação financeira
0x0011	017	Investimento
0x0012	018	Medicamento
0x0013	019	Pagamento de conta
0x0014	020	Técnico
0x0015	021	Cartão Dinheiro
0x0017	023	Administrativo
0x0018	024	Pagamento Mobile
0x0019	025	Recorrência
0x001A	026	Débito para Pagamento
0x001B	027	Carteira Virtual
0x001C	028	Crédito para Pagamento

Códigos dos Serviços

Além dos grupos de serviços, o SCOPE trabalha com o conceito de serviços. Estes serviços estão na tabela abaixo.

Código de serviço	Descrição
006	Compra com cartão de débito á vista
009	Compra com cartão de crédito á vista
013	Pré-Autorização com cartão de crédito
017	Consulta de cheques – a vista
018	Consulta de cheques – pré-datados
020	Compra com cartão de débito à vista forçada
021	Compra com cartão de débito pré-datada

Código de serviço	Descrição
022	Compra com cartão de débito parcelada sem parcela à vista
023	Compra com cartão de débito parcelada – parcela à vista
024	Compra com cartão de débito parcelada – parcela à vista forçada
025	Compra Vale Gás
027	Compra com cartão de crédito parcelado pela administradora
028	Compra com cartão de crédito parcelado pelo estabelecimento
031	Cancelamento de compra de débito
032	Cancelamento de compra de crédito
034	Compra CDC (CNS)
035	Garantia de cheques
036	Desconto de cheques
037	Solicitação de resumo de vendas
039	Compra com cartão de crédito IATA
040	Compra com cartão de crédito IATA parcelado com juros
041	Compra com cartão de crédito IATA parcelado sem juros
042	Cancelamento de compra de credito IATA
043	Cancelamento compra com cartão CDC
044	Consulta planos de pagamento para cartão CDC
045	Compra com cartão CDC
047	Consulta parcelas de crédito
048	Consulta parcelas de débito
050	Compra com cartão de débito Voucher (Alimentação)
051	Cancelamento de compra com cartão de débito Voucher (Alimentação)
056	Cancelamento de garantia de cheque
058	Consulta AVS
059	Cash
060	Cancelamento de Cash
061	Confirmação de Pré-Autorização
062	Estorno de Pré-Autorização
063	Consulta de Pontos - <u>Fidelidade</u>
064	Consulta saldo de crédito
065	Consulta Cash
066	uso interno
067	uso interno
068	Consulta valores possíveis de recarga de celular
069	Recarga de celular
070	Consulta saldo
071	Consulta extrato resumido
072	Consulta extrato
073	Simulação de saque
074	Saque – Crédito
075	Consulta saldo de investimento
076	Consulta extrato de investimento
077	Resgate avulso
078	Resgate
079	Cancelamento de saque
080	Cancelamento de resgate
081	Obtém cartão de investimento
082	Consulta medicamento
083	Compra medicamento
084	Estorno compra medicamento
085	Pagamento de conta com cartão
086	Solicitação de autorização
087	Pagamento de conta sem cartão

Código de serviço	Descrição
088	Débito Voucher parcelado
089	Consulta pagamento de conta
090	Estorno de pagamento de conta
091	Pagamento de fatura
092	Consulta Saldo Débito
093	Resumo de Pagamentos
094	Baixa de O.S.
095	Teste de Comunicação
096	Estatística
097	Moedeiro
098	Compra com Cartão Dinheiro
099	Estorno da compra com Cartão Dinheiro
100	Consulta Contrato
101	Saque INSS
102	Saque Cpcheque
103	Consulta Vale Gás
104	Compra de Cheque Pré-Datada
105	Depósito CDB
106	Resgate CDB
107	Estorno de Depósito CDB
108	Estorno de Resgate CDB
109	Compra com Cartão de Crédito com juros
110	Carga de Cartão Dinheiro
111	Consulta Saldo do Cartão Dinheiro
112	Estorno da Carga de Cartão Dinheiro
113	Parcele Mais
114	Estorno do Parcele Mais
115	Transação Off-line
116	Reservado
117	Reservado
118	Troco Surpresa (Chance Legal)
119	Captura Pré-autorização parcelado estabelecimento
120	Captura Pré-autorização parcelado administradora
121	Consulta Resgate de Prêmios
122	Pagamento de DARF
123	Pagamento de GPS
124	Injeção de chaves
125	Débito parcelado pela administradora
126	Envio de off-line
127	Atualização de chip
128	Autorização de Voucher
129	Compra Crédito parcelado plano
130	Empréstimo
131	Consulta Genérica
132	Carga Cartão Presente
133	Simulação Crediário (<i>Simulação Crediário Débito – obsoleto – substituído pelo 174 – Simulação Crediário Crédito</i>)
134	Crediário (<i>Crediário Débito – obsoleto – substituído pelo 173 – Crediário Crédito</i>)
135	Pagamento Mobile
136	Atualização de preços
137	Elegibilidade Cartão PBM
138	Pré-autorização Medicamento – PBM
139	Cancelamento Pré-autorização Medicamento – PBM
140	Troca de senha do chip

Código de serviço	Descrição
141	Reservado
142	Cancelamento de Autorização de Voucher
143	Pré-autorização parcelado estabelecimento
144	Pré-autorização parcelado administradora
145	Acúmulo Por Valor – <u>Fidelidade</u>
146	Acúmulo Por Pontos – <u>Fidelidade</u>
147	Resgate de Produto – <u>Fidelidade</u>
148	Resgate Monetizado – <u>Fidelidade</u>
149	Cancelamento de Operação <u>Fidelidade</u>
150	Logon Manual
154	Verifica Cartão de Pagamento Recorrente
155	Cadastra Pagamento Recorrente
156	Inclui Cartão de Pagamento Recorrente
157	Estorno de Cadastro de Pagamento Recorrente
158	Saque – Débito
159	Débito para Pagamento de Fatura
161	Consulta Cartão Online
167	Baixa Técnica PDV
168	QR Code Comprador
169	QR Code Vendedor
170	Estorno Carteira Virtual
172	Consulta Perguntas Dinâmicas
173	Crediário Crédito
174	Simulação Crediário Crédito
161	Consulta Cartão Online
167	Baixa Técnica PDV
168	QR Code Comprador
169	QR Code Vendedor
170	Estorno Carteira Virtual
172	Consulta Perguntas Dinâmicas
173	Crediário Crédito
174	Simulação Crediário Crédito
175	Consulta Cartões por CPF
178	Consulta Pagamento de Fatura
179	Consulta Fatura Detalhada
180	Consulta Parâmetros Débito
181	Consulta Parâmetros Crédito
183	Consulta Conversão de Moeda para Crédito
184	Consulta Conversão de Moeda para Débito
185	Verifica Integridade PDV
186	Pagamento CDC
187	Crédito Parc. Cliente
188	Simulação Crédito Parc. Cliente
189	Crédito para Pagamento de Fatura
190	Consulta Pré-Autorização
-	-
192	Solicitação de PIN Impresso

Convênios

As transações relacionadas às transações de PBM exigem o código das redes, as quais são listadas abaixo.

Códigos do Convênio		Significado
Hexadecim al	Decimal	
0x0001	0001	E-Pharma
0x0002	0002	Vidalink
0x0003	0003	Orizon
0x0004	0004	Funcional Card
0x0005	0005	GoodCard
0x0006	0006	Novartis
0x0007	0007	FlexMed
0x0008	0008	DataSUS
0x0009	0009	FarmaSeg
0x000A	0010	Pharmalink
0x000B	0011	PBM Padrão

Modo de entrada

Valores possíveis para o modo de entrada de uma transação retornado pela máscara Modo_Estrada (0x00000008).

Códigos de retorno		Significado
Caracter ASCII	Decimal	
C	0067	CHIP
D	0068	DIGITADO
E	0068	E-COMMERCE
F	0070	FALLBACK-CHIP-MAGNETICO
H	0072	CHEQUE
L	0076	CODIGO-BARRAS
M	0077	MAGNETICO
P	0080	CPF
T	0084	MOBILE
V	0086	CARTEIRA-VIRTUAL
c	0099	CONTACTLESS-CHIP
f	0102	FALLBACK-CHIP-DIGITADO
g	0103	FALLBACK-MAGNETICO-DIGITADO
m	0109	CONTACTLESS-MAGNETICO
o	0111	CHIP-OFFLINE

Apêndice B – Especificação Visanet 4.1

Com a nova especificação da Visanet, denominada Visanet 4.1, as automações já integradas ao SCOPE necessitarão se readequar e se certificar perante a Visanet. Este apêndice descreve as alterações necessárias.

Adequação

Para suportar a Visanet 4.1, os PIN-Pads terão seu *firmware* atualizado. Os PIN-Pads atualizados são denominados *PIN-Pads compartilhados*. Este novo *firmware* é compatível com as versões atuais, ou seja, também suporta as funções antigas de acesso ao PIN-Pads realizada pela biblioteca PPVISA da Visanet, e que eram utilizadas pelas aplicações de PDV. No entanto, quando o SCOPE estiver configurado para trabalhar com o PIN-Pads compartilhado, o acesso ao PIN-Pads será feito pelo SCOPE e não mais pelas aplicações de PDV.

Abaixo seguem algumas funcionalidades da Visanet 4.1 que exigirão adequação por parte da aplicação de PDV:

- Transação de venda crédito à vista ou parcelada.
- Transação de venda débito à vista, parcelada e pré-datada. Esta funcionalidade também contempla o Vale Alimentação e o Vale Refeição. Assim atende as operações da Visanet como prestadora de serviços de Acquirer para os produtos Visa Vale e Valetik. Vinculado à transação de débito à vista, o produto Compre & Saque possibilita ao portador a realização de saque em dinheiro. Durante o fluxo de coleta, desde que habilitado pela rede, o SCOPE solicitará o valor do saque através do estado TC_COLETA_VALOR_SAQUE (0xFC2D).
- O cancelamento de venda pode ser feito quando ocorrerem erros na digitação (valor, data de agenda, número de parcelas etc.) ou desistência da compra por parte do cliente, sendo necessário informar os dados da transação original a ser cancelada.
- Reimpressão de comprovante. Esta função possibilita ao lojista a impressão de uma nova cópia do comprovante da última transação realizada ou alguma específica, desde que tenha sido aprovada e se encontre no log do Concentrador TEF, funcionando como solução para o caso da existência de algum problema com a impressão original.
- Pagamento de contas. Possibilidade de pagamento de contas de concessionárias com cartão Visa Electron em estabelecimentos comerciais afiliados a Visanet (rede de farmácias, supermercados e estabelecimentos de varejo). Somente serão permitidas transações de débito à vista.
- Pré-autorização de crédito. No momento que esta transação é aprovada, a administradora de crédito sensibiliza o limite do cartão, de acordo com o valor informado, como forma de reserva e garantia ao Comércio Usuário que no momento de receber efetivamente o pagamento da despesa. Atualmente somente hotéis e locadoras são autorizados para utilização da Pré-Autorização.
- Consulta CDC. Consulta direta ao banco emissor, possibilita o acesso a taxas aplicadas no momento da intenção de compra.
- AVS. Transação utilizada para verificação do endereço de envio da fatura do portador do cartão. Estes dados são informados pelo mesmo em uma compra feita através de cartão não presente.
- Pagamento de Fatura cartões Private Label.

Certificação

Uma vez que a aplicação de PDV cumpra as adequações mencionadas anteriormente, ela deve ser certificada pela Visanet.

A certificação deve ser agendada com a NCR e consiste em:

- Pré-homologação: realizada pela NCR;
- Homologação: realizada pela Visanet ou por empresa designada por ela.

Apêndice C – PIN-Pad Compartilhado

IMPORTANTE: Este tópico deve ser considerado apenas pela automação que utiliza a interface coleta do SCOPE.

Abaixo segue alguns aspectos para integração com SCOPE na interface coleta:

- O SCOPE será responsável por toda a interação com o PIN-Pad Compartilhado num processo de TEF;
- Quando configurado o PIN-Pad Compartilhado no ScopeCNF, o SCOPE não retornará mais para a aplicação de PDV alguns estados de coleta, pois estes estados serão executados internamente pelo SCOPE. Os estados são:
 - TC_CARTAO
 - TC_COLETA_AUT_OU_CARTAO
 - TC_SENHA
 - TC_DECISAO_CONT
- A aplicação terá a possibilidade de interromper a interação do SCOPE com o PIN-Pad. Para isto a Aplicação PDV deverá:
 - Executar a função **ScopeConfigura** sinalizando que a aplicação PDV utilizará tal recurso (ver [Configurações gerais](#));
 - Quando configurado, o SCOPE na leitura do cartão irá devolver o código **TC_COLETA_CARTAO_EM_ANDAMENTO (0xFCFC)** e durante as demais interações com o PIN-Pad devolverá o código **TC_COLETA_EM_ANDAMENTO (0xFCFD)** através da função **ScopeStatus**. Neste momento, a aplicação de PDV poderá decidir se continua ou interrompe o processo;
 - Para continuar o processo, a aplicação de PDV deverá executar a função `ScopeResumeParam()` com o parâmetro **PROXIMO_ESTADO (0x00)**, e para cancelar, **CANCELAR (0x02)**.
 - Observação: O SCOPE devolverá os códigos **TC_COLETA_EM_ANDAMENTO (0xFCFD)** e **TC_COLETA_CARTAO_EM_ANDAMENTO (0xFCFC)** ao menos uma vez por segundo.
- Para a aplicação de PDV cancelar uma leitura de cartão no PIN-Pad e realizar uma transação digitada, nos casos permitidos, serão oferecidas as seguintes opções:
 - Durante a coleta do cartão, se for acionado a tecla **Cancela** no PIN-Pad;
 - E se no estado **TC_COLETA_CARTAO_EM_ANDAMENTO (0xFCFC)** for executada a função **ScopeResumeParam**, com o parâmetro **CANCELA**.

- Se ocorrer uma das opções acima, o SCOPE devolverá o estado de **coleta TC_CARTAO_DIGITADO (0xFC85)** para receber da aplicação de PDV o número do cartão digitado.
- No arquivo scope.ini poderá ser desabilitado as opções acima - digitação do cartão (ver [Sessão \[PPCOMP\]](#)). Opcionalmente, a aplicação de PDV também poderá utilizar a função [ScopeConfigura](#) para o mesmo fim.
- O SCOPE também disponibilizará a função ScopeValidInterfacePP que possibilitará a aplicação de PDV validar se está utilizando a mesma interface de acesso ao PIN-Pad que o que está configurado no SCOPE.

Apêndice D – Conjunto de bibliotecas do SCOPE Client

Neste apêndice, relacionamos o conjunto de bibliotecas e arquivos que compõe o ambiente em que se localiza o SCOPE Client para cada sistema operacional e linguagem.

MS-WINDOWS®

Para a maioria dos ambientes de programação, as bibliotecas necessárias para a execução do SCOPE Client em MS-Windows® são:

- CMC7.DLL
- COMVERIFONE.DLL
- CSMSG.DLL
- ECF4000.DLL
- ECNF.DLL
- PINPAD.DLL
- PPDIOW32.DLL
- PPGERW32.DLL
- PPINGW32.DLL
- PPSLBW32.DLL
- PPVFNW32.DLL
- PPW32.DLL
- SCOPEAPI.DLL
- SCOPECLT.DLL
- SCOPECNX.DLL
- SCOPECOM.DLL
- SCOPEECF.DLL
- SCOPEISO.DLL
- SCOPELIB.DLL
- SCOPEPRF.DLL
- SCOPEREG.DLL

- SCOPETCP.DLL
- scope.ini

Linguagem Java

Aplicativos de PDV escritos em Java, além das bibliotecas citadas acima, também precisam das seguintes:

- PINPADJAVA.DLL
- PINPADJAVA.JAR
- SCOPEJAVA.DLL
- SCOPEJAVA.JAR

Linux

Os ambientes executados no sistema operacional Linux precisam das seguintes bibliotecas:

- libScopeApi.so
- libScopeCom.so
- libScopeSerial.so
- libSenha.so
- scope.ini

Linguagem Java

Como acontece em MS-Windows®, o aplicativo de PDV escrito em Java também precisa das seguintes bibliotecas:

- libPinpadJava.so
- pinpadjava.jar
- libScopeJava.so
- scopejava.jar

Android

Os ambientes executados com sistema operacional Android precisam das seguintes bibliotecas e arquivos:

- libscopeapi.so
- libScopeJava.so
- clientjavaandroid.jar
- scope.ini

IMPORTANTE:

Para dispositivos com Android 10 ou inferior

Uma pasta com o nome “Scope” deve ser criada na raiz da memória interna dos dispositivos Android.
Por exemplo:

\Phone\scope

Após a criação da pasta, o scope.ini deve ser adicionado à ela.

Para dispositivos com Android 11 ou superior

Uma pasta com o nome “Scope” deve ser criada no diretório de escopo da aplicação do dispositivo Android.
Por exemplo, se o diretório de escopo da aplicação é:

\Phone\Android\data\com.empresaplataforma\files\

Criar a pasta scope dentro de files:

\Phone\Android\data\com.empresaplataforma\files\scope\

Após a criação da pasta scope, o scope.ini deve ser adicionado à ela.

Além disso, para o Android 11 ou superior, é necessária a chamada da função setConfig pela aplicação de automação, para informar ao Scope Client o diretório de escopo da aplicação.

OBSERVAÇÃO: Todos os arquivos de controle utilizados pelo Scope, e os arquivos de trace (quando habilitados) serão gerados na pasta “Scope”.

Apêndice E – Android

Configuração do projeto no Android Studio

Após a criação do projeto no Android Studio, no Windows Explorer, navegar até a pasta \app\src\main. Dentro dessa pasta main, criar uma pasta jniLibs.

Verifique se o seu pacote de arquivos possui o arquivo clientjavaandroid.jar.

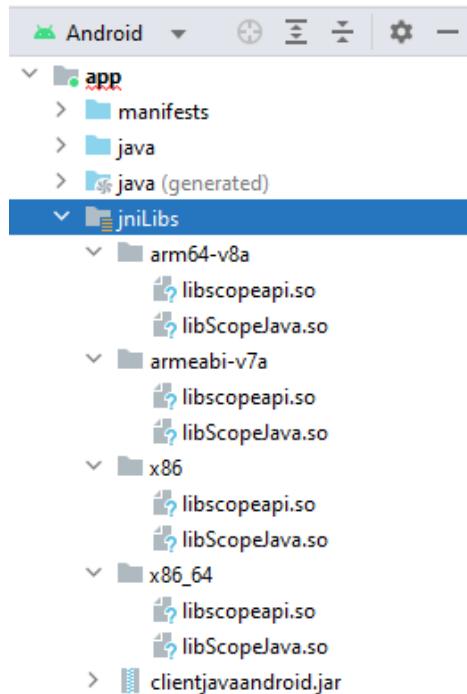
Caso possua, siga para o Passo A. Senão, siga para o passo B.

Passo A: Para pacotes do Scope Client Android sem suporte a terminais POS:

Pacotes da biblioteca sem suporte a terminais POS possuem o arquivo clientjavaandroid.jar.

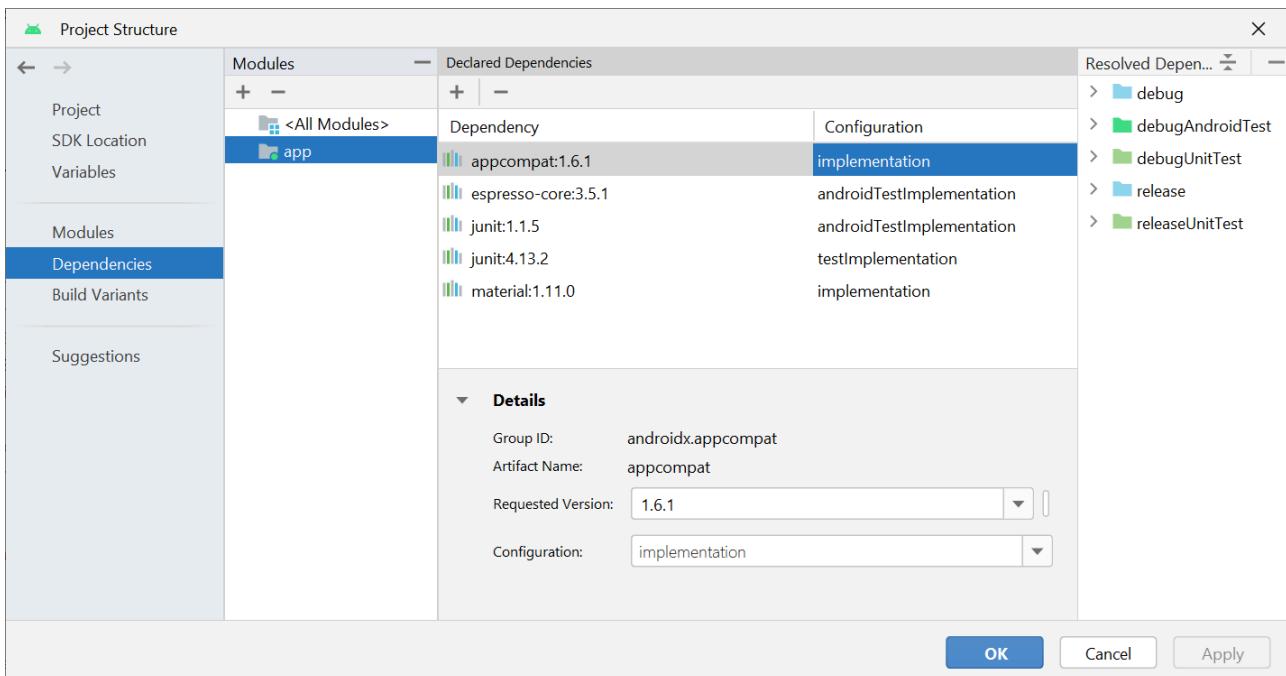
Copiar o conteúdo da pasta libs-Android-xyyzzz-Cwww para a pasta jniLibs.

Onde xyyzzzwww é a versão do Scope.

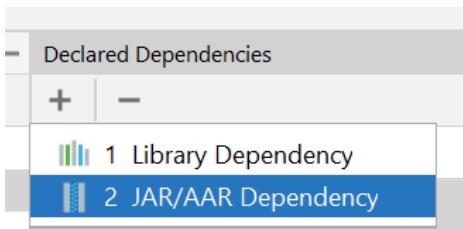


Clicar com o botão direto sobre o projeto-> Open Module Settings -> Dependences.

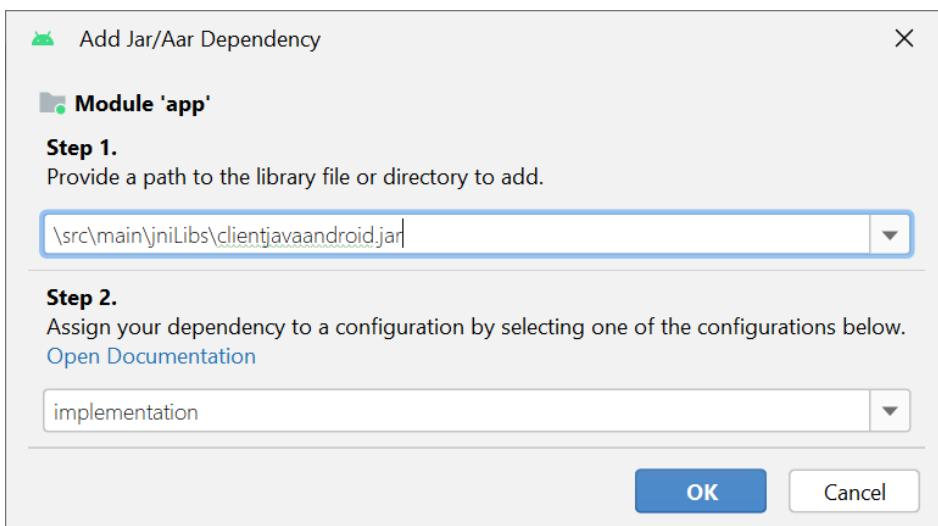
Selecionar o módulo app:



Em Declared Dependencies, adicionar uma JAR/AAR Dependency:



Adicionar clientjavaandroid.jar como dependência do projeto.



Passo B: Para pacotes do Scope Client Android com suporte a terminais POS:

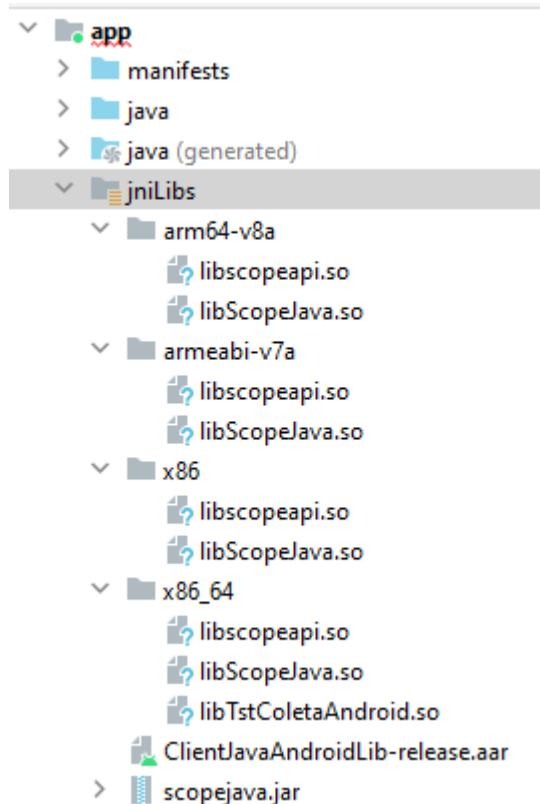
Pacotes da biblioteca com suporte a terminais POS possuem o arquivo ClientJavaAndroidLib-release.aar e scopejava.jar.

Para uma aplicação que **será executada em um POS**, copie o conteúdo da pasta **libs-Android-xyyzzz-Cwww-pos\jniLibs** para a pasta jniLibs.

Para uma aplicação que **será executada em celulares, tablets, totens**, copie o conteúdo da pasta **libs-Android-xyyzzz-Cwww\jniLibs** para a pasta jniLibs.

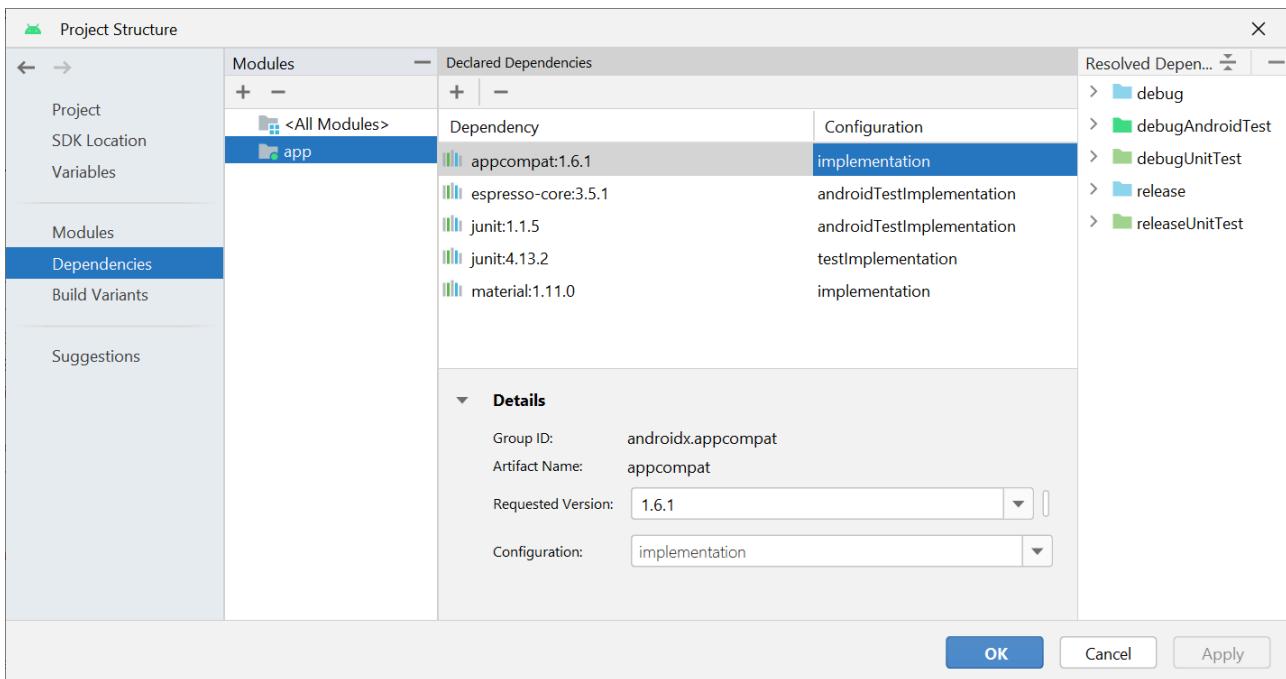
Onde xyyzzzwww é a versão do Scope.

Como para ambos os casos (POS ou outros dispositivos) o nome das bibliotecas é o mesmo, teremos:

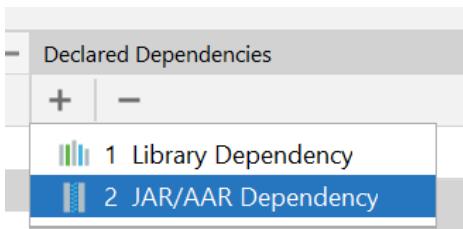


Clicar com o botão direto sobre o projeto-> Open Module Settings -> Dependences.

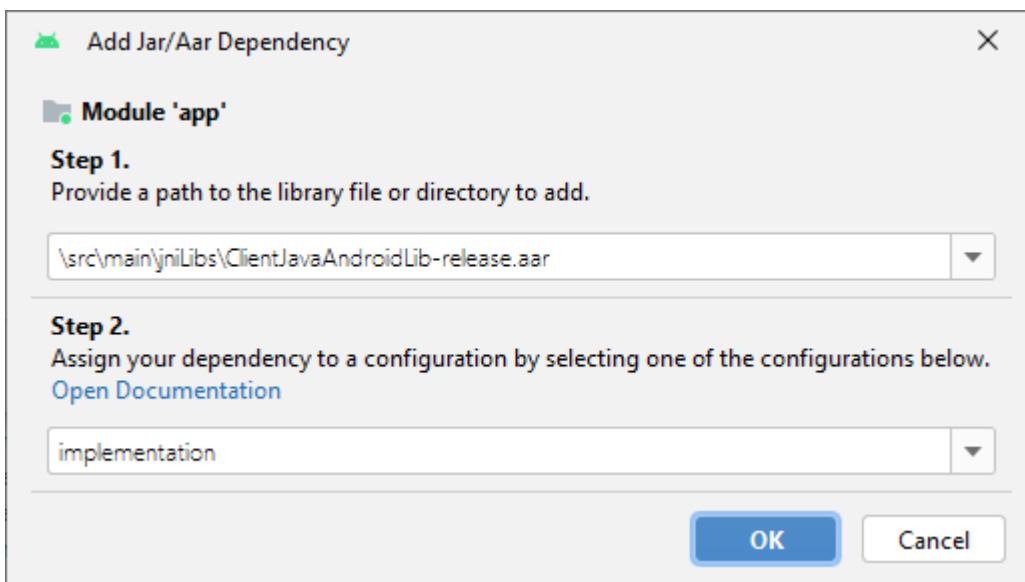
Selecionar o módulo app:



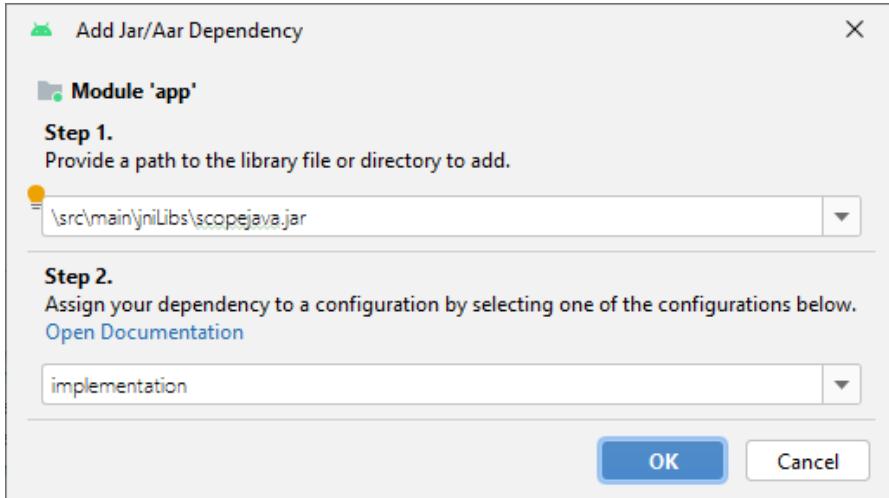
Em Declared Dependencies, adicionar uma JAR/AAR Dependency:



Adicionar ClientJavaAndroidLib-release.aar como dependência do projeto.



Repita o mesmo processo para a lib scopejava.jar:



Configurando o AndroidManifest

O Scope faz uso de alguns recursos do dispositivo, para o bom funcionamento do Scope Client Android, é necessário adicionar as seguintes permissões:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
```

Para comunicação com pinpads Bluetooth:

Além das permissões acima, para comunicação com pinpads Bluetooth, adicionar:

```
<uses-permission
    android:name="android.permission.BLUETOOTH"
    android:maxSdkVersion="30" />
<uses-permission
    android:name="android.permission.BLUETOOTH_ADMIN"
    android:maxSdkVersion="30" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
<uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>

<uses-feature
    android:name="android.hardware.bluetooth"
    android:required="true" />
```

Para comunicação com pinpads USB:

Para comunicação com pinpads USB, adicione o elemento <uses-feature> que declare que seu aplicativo usa o recurso android.hardware.usb.host.

```
<uses-feature android:name="android.hardware.usb.host" />
```

Diretório /scope

Descrição

Para dispositivos com Android 10 ou inferior, a pasta para os arquivos do scope é:

/sdcard/scope

Já para dispositivos com Android 11 ou superior, existe uma restrição de acesso dessa versão do sistema, e os aplicativos possuem acesso apenas à pasta de escopo da aplicação.

Deve-se criar uma pasta “scope” dentro do diretório de escopo da aplicação, por exemplo:

/sdcard/Android/data/com.ncr.scope.exampleclientandroid/files/scope

Independentemente da versão do Android, todos os arquivos de controle utilizados pelo Scope, e os arquivos de trace (quando habilitados) serão gerados na pasta “scope”.

Também o arquivo scope.ini deve estar localizado nessa pasta.

setConfig

Para dispositivos com Android 11 ou superior, o aplicativo deve chamar a função setConfig para informar a pasta de escopo para o Scope Client.

Essa função deve ser chamada logo após criar o objeto para a classe Scope.

Classe	Retorno	Método
Scope	Int	setConfig(String path)

Parâmetros

[in]	String	path	Diretório de escopo da aplicação
------	--------	------	----------------------------------

Exemplo

```

...
long rc;
Scope scope = new Scope();

String mConfigFilePath = "";
if (android.os.Build.VERSION.SDK_INT > Build.VERSION_CODES.Q) {
    mConfigFilePath = getExternalFilesDir(null).getAbsolutePath() + "/scope/";
    rc = scope.setConfig(mConfigFilePath);
    if (rc != Scope.PC_OK){
        // Falha ao configurar o diretório
    }
}

```

Arquivo Scope.ini

Como comentado no tópico anterior, o arquivo scope.ini deve estar na pasta /scope.
Segue exemplo de um arquivo scope.ini:

```
[00010001]
Name=127.0.0.1
Port=2050
TimeOutLogon=30
TimeOutAdm=30
TimeOutSrvJava = 40
ThinClient=s
ExibeQRcode=S

[PINPAD]
TamMinDados=4
```

Neste exemplo, temos Empresa=0001 e Filial= 0001.

Além disso, neste exemplo, considera-se que estão instalados os apps Scope Connector e Scope Client NA no dispositivo.

Para ativar a gravação de arquivos de log para análise, inserir a seguinte chave com os parâmetros:

```
[SCOPEAPI]
TraceApi=s
TracePin=s2
TraceSrl=s
```

Esses parâmetros para gravação de arquivos de log não devem estar ativados em produção.

Captura de arquivos de log do POS

Em ambiente de produção, o acesso aos arquivos de um terminal POS pode ser restrito, desta forma, recomendamos que seja implementado pela automação um serviço que capture e envie o logcat do terminal para uma máquina remota, para possibilitar a análise de eventuais incidentes em campo.

PIN-Pad Bluetooth

Scope PSW

No Scope PSW, o terminal PDV deve estar associado a um perfil de Hardware configurado da seguinte forma:

Fabricante: Genérico

Equipamento: Pinpad Biblioteca ABECS

Porta: COM1

Uso exclusivo do Scope = Não

Comunicação com o PIN-PAD

Para realizar a conexão com o PINPad é necessário criar um objeto do tipo BTCommConnector e setar o MAC Address pelo método setAddress.

Classe	Retorno	Método
BTCommConnector	void	setAddress(String endereço)

Parâmetros

[in]	String	endereço	informa ao SCOPE o endereço MAC Adress do PIN-PAD
------	--------	----------	---

Exemplo

```

if (!Util.isPosHardware()) {
    Log.d(TAG, "Não é POS()");
    //Inicializa classes bluetooth e solicita permissão
    if (getTipoComunicao() == ScopeConstantes.PPCanalComunicacao.PC_COMM_BLUETOOTH) {
        Log.d(TAG, "Iniciando Bluetooth...");
        application.setBluetooth(new BTCommConnector());
        btConnector = application.getBluetooth();
        application.setMacAddress(pref.getString(BLUETOOTH_ADDRESS_KEY, null));
        btConnector.setAddress(application.getMacAddress());
        requestBluetooth();
    }
}

```

Iniciando comunicação

Antes do início de qualquer transação, a aplicação deve abrir o PIN-Pad, ou seja, iniciar o canal de comunicação com o PIN-Pad. O método ppOpenSecure da classe Scope é utilizado para abrir a comunicação.

Classe	Retorno	Método
--------	---------	--------

Scope	int	ppOpenSecure(int tipoCanal, String endereco)
-------	-----	--

Parâmetros

[in]	int	tipoCanal	Informa o tipo de canal que pode ser:2 – USB; 3 – Bluetooth
[in]	String	endereco	Endereço bluetooth

Exemplo

```

...
long rc;
BTCommConnector btCommConnector = new BTCommConnector();
...
Scope scope = new Scope();
...
rc      =      scope.ppOpenSecure(ScopeConstantes.PPCanalComunicacao.PC_COMM_BLUETOOTH,
"54:7F:54:60:7B:90");
if (rc != Scope.PC_OK){
// Falha ao abrir o PIN-PAD
}

```

Encerrando a comunicação

Uma vez que o PIN-Pad não será mais utilizado, a aplicação pode encerrar a comunicação com o PIN-Pad, deixando uma mensagem no visor (*display*) do PIN-Pad.

Classe	Retorno	Método
Scope	int	ppClose(String mensagem)

Parâmetros

[in]	String	mensagem	Mensagem de 32 caracteres a ser deixada no display do PIN-PAD.
------	--------	----------	--

Exemplo

```

...
long rc;
rc = scope.ppClose("OKI Brasil");

```

IMPORTANTE:

- A criação do objeto do tipo BTCommConnector deve ser feita antes do objeto Scope.

OBSERVAÇÃO:

- Todos os métodos da Classe Scope podem ser consultados pelo JavaDoc do Scope.

PIN-Pad USB

Scope PSW

No Scope PSW, o terminal PDV deve estar associado a um perfil de Hardware configurado da seguinte forma:

Fabricante: Genérico

Equipamento: Pinpad Biblioteca ABEC5

Porta: COM1

Uso exclusivo do Scope = Não

Pinpad Compatível

Gertec PPC 930 versão de especificação 2.12

Definindo variáveis

Adicionar os seguintes imports e atributos no seu projeto.

```
import android.app.PendingIntent;
import android.hardware.usb.UsbDevice;
import android.hardware.usb.UsbManager;
import android.content.BroadcastReceiver;
import com.itautec.scope.client.usb.UsbCommConnector;
...
private UsbCommConnector usbConnector = null;
private UsbDevice device = null;
private PendingIntent mPermissionIntent;
private UsbManager manager;
```

BroadcastReceiver

Adicione um BroadcastReceiver para a permissão de acesso ao dispositivo USB conectado.

```
private final static BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
    UsbDevice deviceTmp = null;

    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (UsbCommConnector.ACTION_USB_PERMISSION.equals(action)) {
            synchronized (this) {
                deviceTmp = (UsbDevice)
            intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
                if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED,
false)) {
                    if (deviceTmp == null) {
```

```
Toast.makeText(context, "Não permitiu",  
Toast.LENGTH_LONG).show();  
}  
}  
}  
}  
};
```

Iniciando comunicação USB e recuperando o dispositivo conectado

Adicione o código a seguir no seu projeto para iniciar a comunicação USB, recuperar o dispositivo conectado e solicitar permissão do usuário.

```

if (!Util.isPosHardware()) {
    //Pinpad USB
    if (getTipoComunicao() == ScopeConstantes.PPCanalComunicacao.PC_COMM_USB) {
        //inicia o usb
        usbConnector = new UsbCommConnector();
        manager = (UsbManager) getSystemService(Context.USB_SERVICE);

        //Configura path para arquivo de log usb.txt
        usbConnector.setPathScope(getScopeIniPath());

        //Recupera o dispositivo conectado à USB do celular
        device = usbConnector.getDevice(this);

        //Se existe dispositivo conectado, solicita permissão do usuário para conexão
        if (device != null) {

            mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(UsbCommConnector.ACTION_USB_PERMISSION),
                PendingIntent.FLAG_IMMUTABLE);
            IntentFilter filter = new IntentFilter(UsbCommConnector.ACTION_USB_PERMISSION);
            registerReceiver(mUsbReceiver, filter);

            // Utilizado para exibir um dialog pedindo permissão para o respectivo device.
            manager.requestPermission(device, mPermissionIntent);
        }
    }
}
}(...)

```

Abrindo a comunicacão com o pinpad

Antes do início de qualquer transação, a aplicação deve abrir o PIN-Pad, ou seja, iniciar o canal de comunicação com o PIN-Pad. O método `ppOpenSecure` da classe `Scope` é utilizado para abrir a comunicação.

Classe	Retorno	Método
Scope	int	ppOpenSecure(int tipoCanal, String endereco)

Parâmetros

[in]	int	tipoCanal	Informa o tipo de canal que pode ser:2 – USB; 3 – Bluetooth
[in]	String	endereco	Endereço bluetooth

Exemplo

```

import com.itautec.scope.Scope;
import com.itautec.scope.ScopeConstantes;
import com.itautec.scope.StConsultaPP;
...
private StConsultaPP cPP;
public Scope scope;
protected int tipoDeComunicacao;
...
scope = new Scope()
...
// Para comunicação USB
tipoDeComunicacao = ScopeConstantes.PPCanalComunicacao.PC_COMM_USB;
...

int rc = scope.consultaPP(cPP);
if (rc == Scope.PC_OK) {
    if (cPP.usoExclusivoScope == 0) {
        if (cPP.configurado == ScopeConstantes.PPModoOperacao.PC_MODO_ABEC5) {
            if (tipoDeComunicacao ==
ScopeConstantes.PPCanalComunicacao.PC_COMM_BLUETOOTH) {
                RC = scope.ppOpenSecure(tipoDeComunicacao, macAddress);
            } else {
                // Abrindo comunicação com pinpad USB
                RC = scope.ppOpenSecure(tipoDeComunicacao, String.format("%d",
cPP.porta));
            }
        }
    }
}

```

Encerrando a comunicação

Uma vez que o PIN-Pad não será mais utilizado, a aplicação pode encerrar a comunicação com o PIN-Pad, deixando uma mensagem no visor (*display*) do PIN-Pad.

Classe	Retorno	Método
Scope	int	ppClose(String mensagem)

Parâmetros

[in]	String	mensagem	Mensagem de 32 caracteres a ser deixada no display do PIN-PAD.
------	--------	----------	--

Exemplo

```
...
long rc;
rc = scope.ppClose("OKI Brasil");
```

OBSERVAÇÃO:

- Todos os métodos da Classe Scope podem ser consultados pelo JavaDoc do Scope.

POS

Scope PSW

No Scope PSW, o terminal PDV deve estar associado a um perfil de Hardware configurado da seguinte forma:

Fabricante: Genérico

Equipamento: Pinpad Biblioteca Compartilhada

Porta: COM1

Uso exclusivo do Scope = Não

POS Compatíveis

Gertec GPOS720

Positivo L400

Adicionando libs da Gertec para o GPOS720

Para integração com o POS Gertec GPOS720, é necessária a inclusão de algumas libs da Gertec.

Acesse a pasta **libs-Android-xyyzzz-Cwww-pos\libsGPOS720**.

Onde xyyzzzwww é a versão do Scope.

Copie as libs da Gertec, por exemplo:

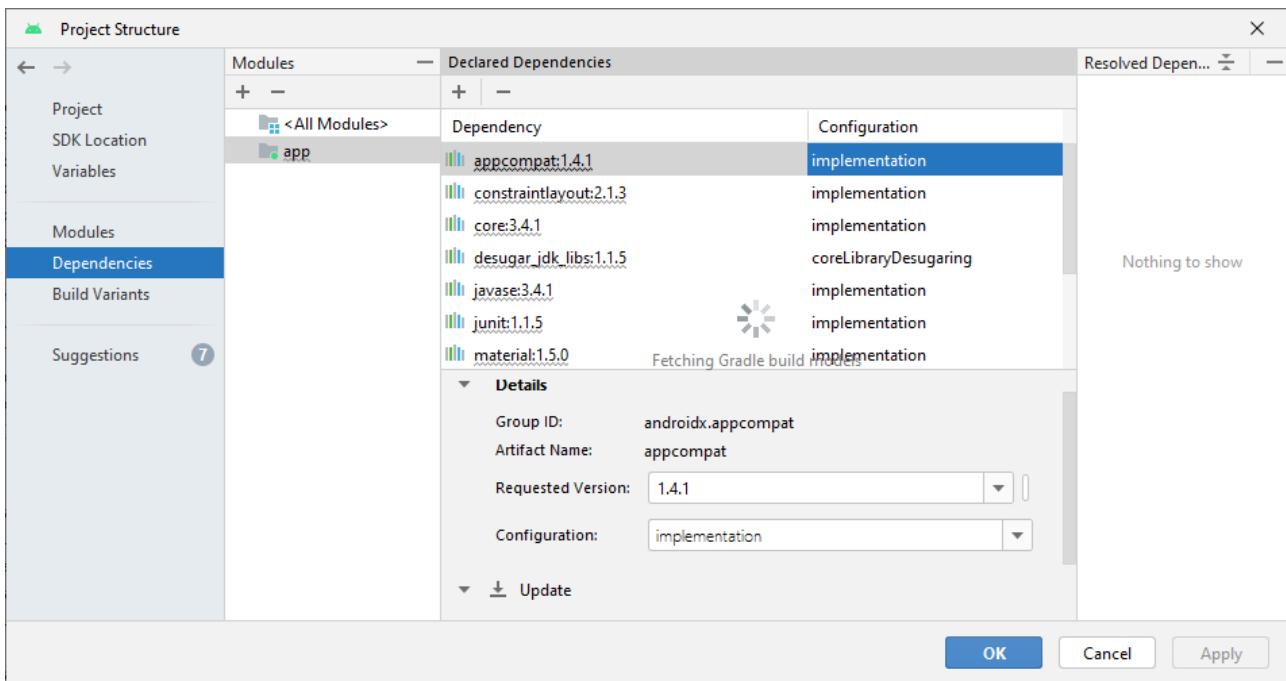
- libgandi-0.1.20-gpos720-payment-release.aar
- libgedi-0.16.21-gpos720-payment-release.aar
- PPCompGPOS720-v1.37.aar

Para a seguinte pasta da sua aplicação:

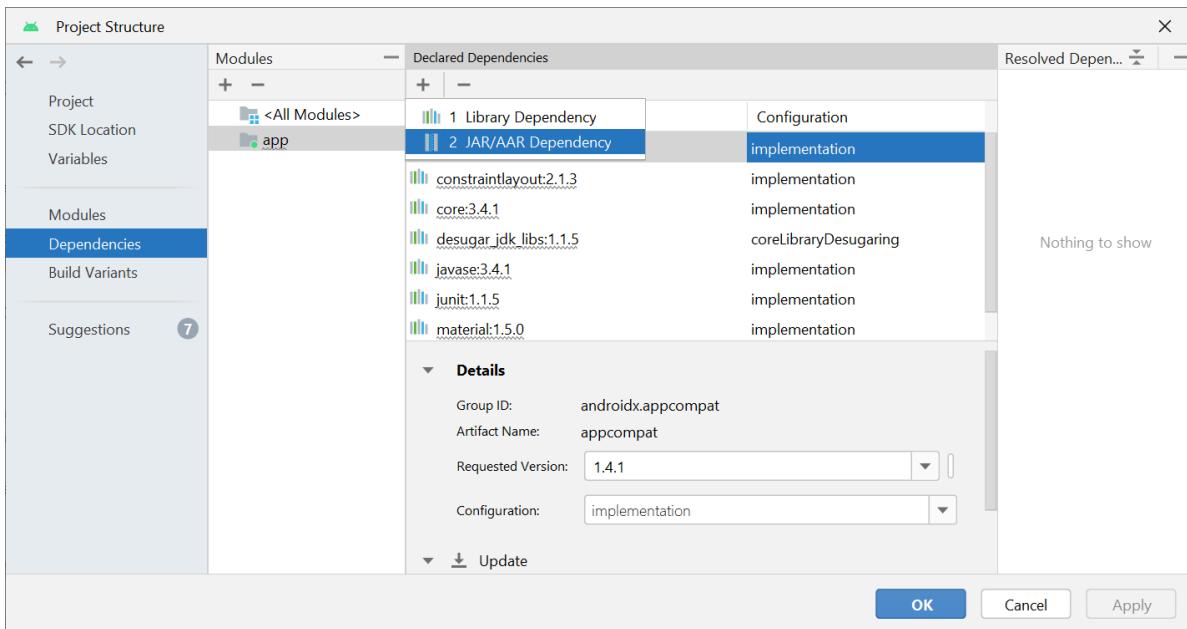
\app\libs

No Android Studio, selecionar o projeto com o botão direito, e então Open Module Settings.

Selecione à esquerda a opção Dependencies:

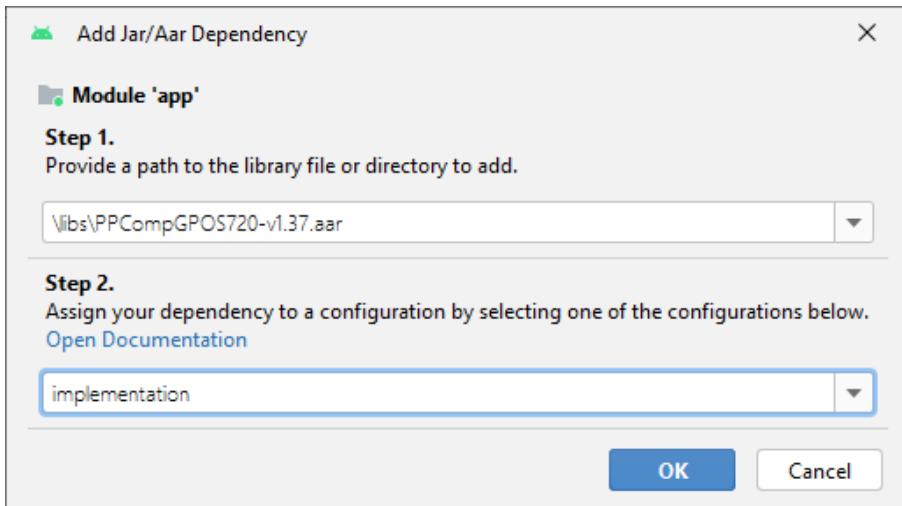


Selecione em Modules a opção “app”. Em seguida, em All Dependencies, selecione o sinal de “+” e JAR/AAR Dependency:



Informe o caminho do arquivo a ser atualizado.

Por exemplo, se o arquivo a ser adicionado ao projeto é PPCompGPOS720-v1.37.aar:



Selecionar OK.

Conferir no arquivo build.gradle (app) que foi inserida a dependencia:

```
implementation files(""\libs\"\PPCompGPOS720-v1.37.aar")
```

Apague no build.gradle a referência à dependência do arquivo anterior.

Repetir o processo para os demais arquivos da Gertec a serem adicionados ao projeto.

Adicionando libs da Positivo para o L400

Para integração com o POS Positivo L400, é necessária a inclusão de algumas libs da Positivo.

Acesse a pasta **libs-Android-xyyzzz-Cwww-pos\libsL400**.

Onde xyyzzzwww é a versão do Scope.

Copie as libs da Positivo, por exemplo:

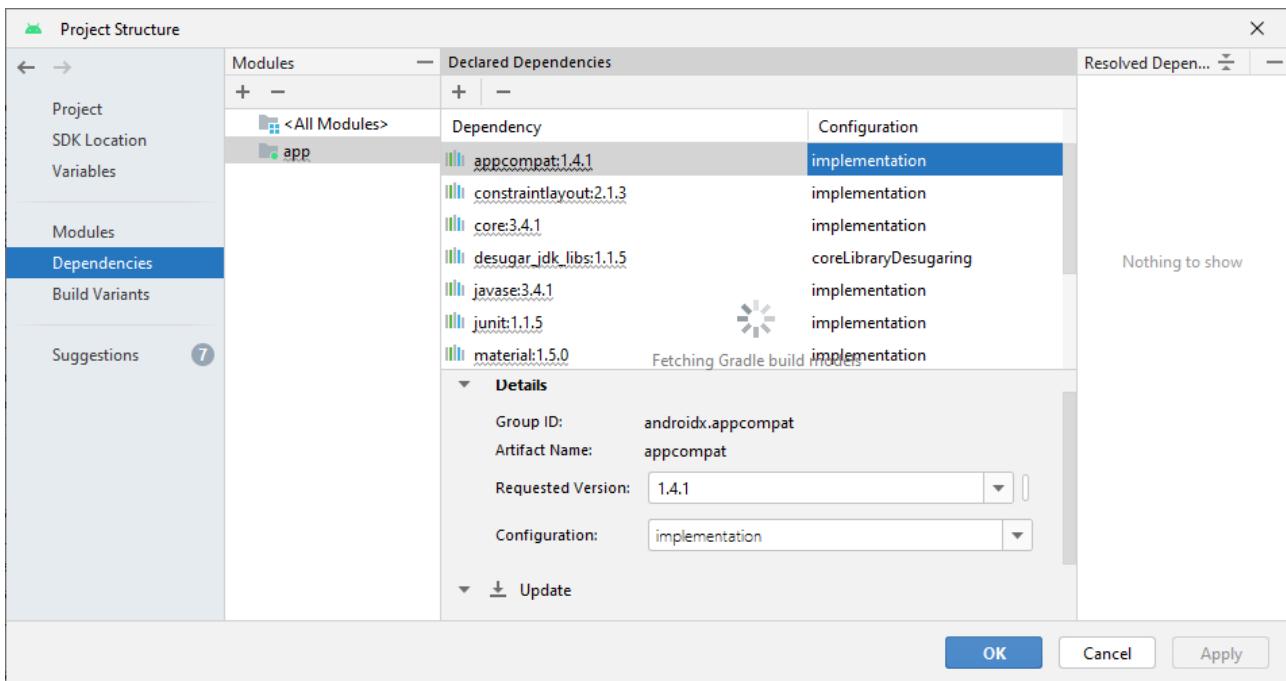
- PositivoAPI-15.8.5-release.aar
- PositivoLIB-15.9.38-ncr-release.aar

Para a seguinte pasta da sua aplicação:

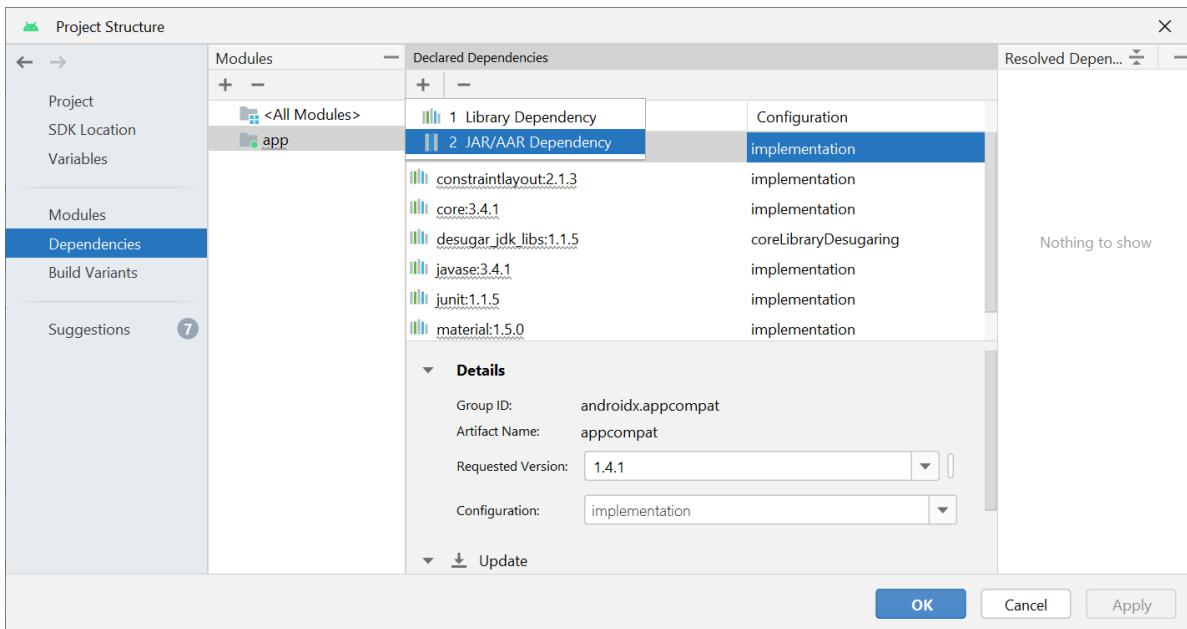
\app\libs

No Android Studio, selecionar o projeto com o botão direito, e então Open Module Settings.

Selecione à esquerda a opção Dependencies:

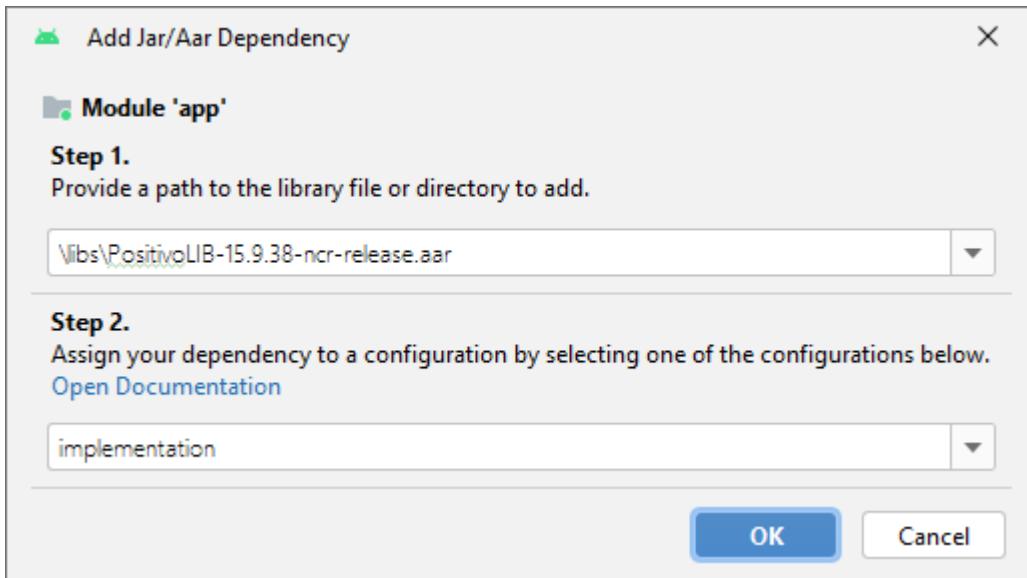


Selecione em Modules a opção “app”. Em seguida, em All Dependencies, selecione o sinal de “+” e JAR/AAR Dependency:



Informe o caminho do arquivo a ser atualizado.

Por exemplo, se o arquivo a ser adicionado ao projeto é PositivoLIB-15.9.38-ncr-release.aar:



Selecionar OK.

Conferir no arquivo build.gradle (app) que foi inserida a dependencia:

```
implementation files("'\libs'\PositivoLIB-15.9.38-ngr-release.aar")
```

Apague no build.gradle a referência à dependência do arquivo anterior.

Repetir o processo para os demais arquivos da Positivo a serem adicionados ao projeto.

Inicializando as classes de comunicação com o POS

No início da aplicação, criar uma instância da classe InitPOS, conforme exemplo a seguir.

Exemplo:

```
private InitPos initPos = null;
private static Impressora impressora;

//Inicializacao para comunicacao com POS
initPos = new InitPos(this);

//Inicializa classe para impressao
impressora = getImpressora(ctx);
```

BroadcastReceiver

A biblioteca Scope Client Android pode devolver para a automação mensagens originárias da biblioteca do POS.

Essas mensagens são enviadas para a automação através de um BroadcastReceiver.

Essas mensagens devem ser exibidas na tela da aplicação.

Exemplos de mensagens que podem ser recebidas pelo BroadcastReceiver:

Tipo de mensagem	Exemplo de mensagem
Atualizando tabelas	ATUALIZANDO TABELAS
Cartão selecionado	SELECIONADO: MasterCard
DCC	SELECT CURRENCY IN NEXT SCREEN PRESS GREEN: USD PRESS RED: BRL
DCC	PRESS GREEN/RED USD 2,00 BRL 10,10 1BRL=0,197851USD
Saldo voucher insuficiente	SALDO VALOR 250,50
Última tentativa de senha	CUIDADO! O PRÓXIMO ERRO BLOQUEARÁ A SENHA
Senha bloqueada	SENHA BLOQUEADA
null	Ação: <Limpar mensagens>

A seguir, exemplo de registro do BroadcastReceiver.

Exemplo:

```

BroadcastReceiver scopeBroadcast;

public void registerScopeBroadcast() {
    Log.d(TAG, "registerScopeBroadcast()");
    scopeBroadcast = new BroadcastReceiver() {
        @Override
        public void onReceive(Context arg0, Intent i) {
            Log.d(TAG, "onReceive:");
            String message = i.getStringExtra("message");

            Log.d(TAG, "onReceive: ScopeReceiver: message: " + message);

            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if (message != null) {
                        displayMsgPOS.setText(message);
                    } else {
                        //Limpas mensagens
                        displayMsgPOS.setText(null);
                    }
                }
            });
        }
    };
    context.registerReceiver(scopeBroadcast, new IntentFilter("br.com.scope.intent.ScopeReceiver"));
}

public void unregisterScopeBroadcast() {
    Log.d(TAG, "unregisterScopeBroadcast()");
}

```

```

try{
    if(scopeBroadcast!=null)
        context.unregisterReceiver(scopeBroadcast);
}catch(Exception e){}
}

```

Neste exemplo, a mensagem será exibida em um TextView na tela.

A seguir, exemplo de chamada do método para registrar o BroadcastReceiver, sendo realizado no onCreate de uma activity.

Exemplo:

```

protected void onCreate(Bundle savedInstanceState) {
(...)

//BroadcastReceiver para mensagens de call-back da biblioteca do POS
if (Util.isPosHardware()) {
    registerScopeBroadcast();
}

```

A seguir, exemplo de chamada do método para remover o BroadcastReceiver, sendo realizado no onDestroy de uma activity.

Exemplo:

```

public void onDestroy() {
    Log.d(TAG, "onDestroy");
    super.onDestroy();
    unregisterScopeBroadcast();
}

```

Abrindo a comunicação com o pinpad

Antes do início de qualquer transação, a aplicação deve abrir o PIN-Pad, ou seja, iniciar o canal de comunicação com o PIN-Pad. O método ppOpen da classe Scope é utilizado para abrir a comunicação.

Classe	Retorno	Método
Scope	int	ppOpen(int porta)

Parâmetros

[in]	int	Porta	Porta COM. No caso de POS, informar 1.
------	-----	-------	--

Exemplo

```

import com.itautec.scope.Scope;
import com.itautec.scope.ScopeConstantes;
import com.itautec.scope.StConsultaPP;

...
private StConsultaPP cPP;
public Scope scope;
protected int tipoDeComunicacao;

...
scope = new Scope()
...

int rc = scope.consultaPP(cPP);
if (rc == Scope.PC_OK) {
    if (cPP.usoExclusivoScope == 0) {
        if (cPP.configurado == ScopeConstantes.PPModoOperacao. PC_MODO_COMPART ) {
            RC = scope.ppOpen(cPP.porta);
        }
    }
}
}

```

Encerrando a comunicação

Uma vez que o PIN-Pad não será mais utilizado, a aplicação pode encerrar a comunicação com o PIN-Pad, deixando uma mensagem no visor (*display*) do PIN-Pad.

Classe	Retorno	Método
Scope	int	ppClose(String mensagem)

Parâmetros

[in]	String	mensagem	Mensagem de 32 caracteres a ser deixada no display do PIN-PAD.
------	--------	----------	--

Exemplo

```

...
long rc;
rc = scope.ppClose("OKI Brasil");

```

Cancelando a coleta de cartão: GPOS720 e L400

Para o GPOS720 e L400, para cancelar a coleta de cartão, é necessário chamar o método `scope.ppAbort()`. Veja esse trecho de código sugerido, extraído do projeto exemplo.

Neste exemplo, a tela de interface para o usuário é uma activity que implementa `OnClickListener`, e há um botão Cancelar nessa tela.

Ao clicar no botão Cancelar, é chamado o método `acaoCancelar`:

```

public void acaoCancelar() {
    Log.d(TAG, "acaoCancelar()");
}

```

```

if (retorno == Scope.TC_COLETA_CARTAO_EM_ANDAMENTO ||
    retorno == Scope.TC_COLETA_EM_ANDAMENTO ||
    retorno == Scope.TC_INFO_RET_FLUXO ) {
    if (isGPos() || isL400()) {
        Log.d(TAG, "cancelarColeta ");
        scope.ppAbort();
        return;
    }
}
(...)
```

Além disso, no estado de coleta TC_INFO_RET_FLUXO (64766), desabilitar o botão Próximo.

```

// apenas mostra informacao e deve retornar ao scope
case Scope.TC_INFO_RET_FLUXO:
    if (isPosHardware()) {
        if (isGPos() || isL400()) {
            proximoHabilitado = false;
            //Desabilita botao Voltar para Estorno na coleta de cartao
            if ((this instanceof ServicoCancelamento) && (paramColeta.MsgOp2.contains("Passe o Cartao"))) {
                retornaHabilitado = false;
            }
        } else {
            cancelaHabilitado = false;
            retornaHabilitado = false;
        }
        habilitaBotoesTela(retornaHabilitado, cancelaHabilitado, proximoHabilitado);
        //Deve chamar ScopeResumeParam, que só retornará resposta após cartão lido.
        //Para cancelar a coleta, chamar scope.ppAbort()
    } else {
        habilitaBotoesTela(false, false, false);
    }
    break;
```

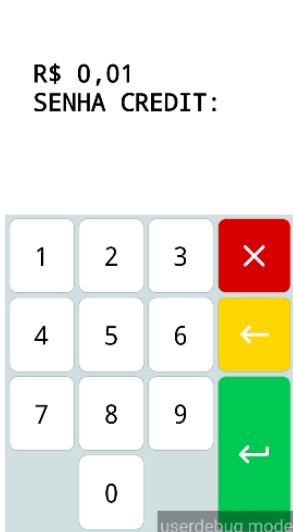
Telas exibidas pela biblioteca Scope Client Android

A biblioteca Scope Client Android poderá exibir telas para o usuário em algumas situações.
Alguns exemplos a seguir.

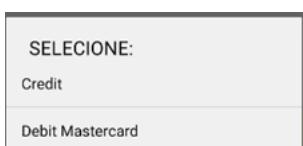
Tela para coleta de senha, para GPOS720:



Tela para coleta de senha, para L400:



Tela para seleção do AID (Application Identifier), para cartões com múltiplos AIDs:



Tela para opção do DCC:



Nesse exemplo de tela, além dos botões para seleção da opção DCC (ANULA, LIMPA e ENTRA), são exibidas as mensagens informativas de DCC, que são enviadas para a automação via BroadcastReceiver, como comentado em tópico anterior.

OBSERVAÇÃO:

- Todos os métodos da Classe Scope podem ser consultados pelo JavaDoc do Scope.

Métodos básicos

Neste tópico são apresentados apenas os métodos principais da biblioteca Scope Client Android. Todos os métodos da biblioteca estão detalhados ao longo desse manual, sendo que nesse tópico, são apresentados apenas os métodos principais, com descrição e exemplo de chamada na plataforma Android. Para verificar algum outro método, consulte o tópico correspondente nesse manual.

open

Abre comunicação com o Scope Server.

Classe	Retorno	Método
Scope	Int	open(String Modo, String Empresa, String Filial, String PDV)

Exemplo

```

...
long rc;
String empresa = "0001";
String filial = "0001";
String pdv = "001";

// comunicação com o server
rc = scope.open("2", empresa, filial, pdv);

```

close

Encerra comunicação com o Scope Server.

Classe	Retorno	Método
Scope	Int	close()

Exemplo

```

...
long rc;

// encerra comunicação com o server
rc = scope.close();

```

setApiColeta

Informa que toda a coleta de dados será feita pelo aplicativo.

Classe	Retorno	Método
Scope	Int	setApiColeta()

Exemplo

```

...
long rc;
// comunicação com o server
...
// Informa que a aplicação deseja coletar os dados
rc = scope.setApIColeta();

```

consultaPP

Retorna as informações configuradas para o pinpad no Scope.

Classe	Retorno	Método
Scope	Int	consultaPP(StConsultaPP consulta)

Parâmetros

[in]	StConsultaPP	consulta	Dados do PIN-PAD conforme configuração do ScopeCNF.
------	--------------	----------	---

Exemplo

```

...
long rc;
StConsultaPP CPP = new StConsultaPP();
rc = scope.consultaPP(CPP);

if (CPP.configurado != 0 && CPP.usoExclusivoScope == 0){
    // Abre o PIN-PAD
}

```

abreSessaoTEF

Informa ao Scope para iniciar uma nova sessão TEF, que poderá ser formada por uma ou mais transações.

Classe	Retorno	Método
Scope	Int	abreSessaoTEF()

Exemplo

```

...
long rc;
rc = scope.abreSessaoTEF();
if(rc != Scope.RCS_SUCESSO){
    // falha ao abrir a sessão
}

```

fechaSessaoTEF

Aciona o SCOPE para finalizar uma sessão de TEF (ciclo com uma ou mais transações de TEF), ou seja, confirmar ou desfazer as transações da sessão em aberto, após encerrar o processamento da transação.

Classe	Retorno	Método
Scope	Int	fechaSessaoTEF(stSessaoTef sessao)

Parâmetros

[in]	stSessaoTef	sessao	Dados para confirmação ou desfazimento das transações.
------	-------------	--------	--

Exemplo

```

...
long rc;
stSessaoTef stSessao = new stSessaoTef();
if (Scope.PROXIMO_ESTADO){
    stSessao.confirma();
}
else{
    stSessao.desfaz();
}
rc = scope.fechaSesaoTEF (stSessao);

```

status

A consulta do status da transação deve ocorrer enquanto uma transação (cartão de crédito, cartão de débito, recarga de celular, etc.) está em processamento.

Classe	Retorno	Método
Scope	Int	status()

Exemplo

```

...
long rc;
while((rc = scope.status())== Scope.RCS_TRN_EM_ANDAMENTO){
    try{
        Thread.sleep(500);
    }catch(InterruptedException e){}
}
// deve tomar alguma atitude
if((rc >= Scope.TC_PRIMEIRO_TIPO_COLETA) &&
(rc <= Scope.RCS_ULTIMO_COLETA_DADOS){
    Scope.getParam(rc,stColeta);
}

```

getParam

Durante a iteração do processamento de uma transação, a aplicação deve obter informações necessárias para solicitar a coleta do dado para o operador e/ou cliente. A função ScopeGetParam(), responsável em descrever os dados a coletar, deve ser chamada sempre que o SCOPE Client retornar um estado de coleta correspondente a uma coleta de dados (ver Status de transação). Com o retorno do SCOPE Client, a aplicação de PDV deverá atualizar as mensagens das telas do operador e/ou cliente, disponibilizar um meio

de entrada de dado, validar a entrada, entre outras ações. A aplicação deve prover o operador de algum meio que ele possa cancelar, retornar e continuar a coleta de dados da transação.

As mensagens que o SCOPE Client disponibiliza para serem exibidas são fundamentadas no fato de que no PDV haja dois visores (display): um que é apresentado ao cliente da loja e outro, apresentado ao operador do PDV. Além disso, por padrão, considera-se que cada um desses visores possua 2 linhas de exibição de mensagens com largura de 40 caracteres. É possível configurar o tamanho máximo de colunas para até 20 ou 16 (ver o capítulo Configuração).

Classe	Retorno	Método
Scope	Int	getParam(int param, StParamColeta lpparam)

Parâmetros

[in]	int	param	Informa que tipo de parâmetro retornado pelo status(), deseja-se obter.
[in/out]	StParamColeta	lpparam	Objeto que será preenchido pelo Scope Client.

Exemplo

```

...
long rc;
StparamColeta stColeta = new StParamColey();
(rc <= Scope.RCS_ULTIMO_COLETA_DADOS){
    scope.getParam(rc,stColeta);
}
if (rc != Scope.TC_COLETS_CARTAO_EM_ANDAMENTO){
    // exibe mensagens operador
}

```

resumeParam

Retoma o fluxo de execução para o cliente Scope passando os dados coletados pela aplicação, no caso de uma coleta, ou simplesmente retornando o fluxo no caso de uma exibição de mensagem, impressão de cupom ou cheque.

O aplicativo deve informar através do parâmetro acao se o fluxo de prosseguir, retornar para estado anterior, cancelar o fluxo ou mesmo informar que ocorreu um erro na coleta de dados ou impressão do cupom ou cheque.

Classe	Retorno	Método
Scope	Int	resumeParam(int codColeta, stDadosdados dados,int dadosParam, int acao)

Parâmetros

[in]	int	codColeta	Código do estado para qual o dado sói coletado, informado pela aplicação.
[in]	Stdados	dados	Dado coletado pela aplicação
[in]	int	dadosParam	Informa o tipo de captura do dado. Captura pelo teclado (0x0004), pelo chip(0x0080),pela tarja magnética(0x0020)
[in]	Int	acao	Determina a sequência do fluxo de execução do cliente.

Exemplo

```

...
long rc;
stDados dados = new stDados();
dados.Work = "";
rc = scope.resumeParam(rc,dados,Scope.SCO_TECLADO,Scope.Proximo_estado)
if(rc !=Scope.RCS_SUCESSO){
    StColetaMsg smg = new StColetaMsg();
    Scope.getLastMsg(smg);
    // exibe as mensagens
}
else if(rc == Scope.RCS_SUCESSO){
    // confirma ou desfaz a transação
}

```

getLastMsg

Obtém as últimas mensagens a serem mostradas para o operador e/ou cliente.

Classe	Retorno	Método
Scope	Int	getLastMsg(StColetaMsg msg)

Parâmetros

[in/out]	StColetaMsg	msg	Objeto contendo tanto a linha 1 como a linha 2 das mensagens do operador e cliente.
----------	-------------	-----	---

Exemplo

```

...
long rc;
if(rc !=Scope.RCS_SUCESSO){
    StColetaMsg msg = new StColetaMsg();
    Scope.getLastMsg(msg);
    // exibe as mensagens
    //msg.Cl1, msg.Cl2, msg.Op1, msg.Op2
}

```

getCupomEx

Devolve a via do cupom Cliente e do Estabelecimento separadamente. Além disso, devolve o cupom reduzido se houver.

Classe	Retorno	Método
Scope	Int	getCupomEx(StCupomEx cupom)

Parâmetros

[in/out]	StCupomEx	cupom	Todos os cupons disponíveis.
----------	-----------	-------	------------------------------

Exemplo

```

...
long rc;
String cupom;
StCupomEx cupom = new StCupomEx();
rc = scope.getCupomEx(cupom);
cupom = "Cabecalho:\n"+cupom.cabec
      +"\nCupom Cliente:\n"+cupom.cpCliente
      +"\nCupom Loja:\n"+ cupom.cpLoja;

```

compraCartaoCredito

Aciona o Scope para efetuar uma transação de compra com cartão de crédito.

Classe	Retorno	Método
Scope	Int	compraCartaoCredito(String valor, String txservico)

Parâmetros

[in]	String	valor	Valor da trasacao com máximo de 12 dígitos.
[in]	String	txservico	Valor da taxa de serviço

Exemplo

```

...
long rc;
// abre sessão
rc = scope.compraCartaoCredito("123","000");

```

compraCartaoDebito

Aciona o Scope para efetuar uma transação de compra com cartão de débito.

Classe	Retorno	Método
Scope	Int	compraCartaoDebito(String valor)

Parâmetros

[in]	String	valor	Valor da trasacao com máximo de 12 dígitos.
------	--------	-------	---

Exemplo

```

...

```

```

long rc;
// abre sessão
rc = scope.compraCartaoDebito("123");

```

cancelamento

Aciona o Scope para efetuar o cancelamento de uma transação previamente autorizada.

Classe	Retorno	Método
Scope	Int	cancelamento(String valor, String txservico)

Parâmetros

[in]	String	valor	Valor da trasacao com máximo de 12 dígitos.
[in]	String	txservico	Valor da taxa de serviço

Exemplo

```

...
long rc;
// abre sessão
rc = scope.cancelamento("1243","000");

```

obtemListaCarteiraVirtual

Aciona o Scope para efetuar uma transação para conhecer todas as carteiras virtuais disponíveis.

Classe	Retorno	Método
Scope	Int	obtemListaCarteiraVirtual (ScopeCarteirasVirtuais carteirasVirtuais)

Parâmetros

[out]	String	carteirasVirtuais	Classe com a lista de carteiras virtuais.
-------	--------	-------------------	---

Exemplo

```

int RC = ScopeConstantes.CodigosRetorno.RCS_SUCESSO;

ScopeCarteirasVirtuais buffer = new ScopeCarteirasVirtuais();
buffer.setVersao("01");

RC = scope.obtemListaCarteiraVirtual(buffer);

```

```
if (RC == ScopeConstantes.CodigosRetorno.RCS_SUCESSO) {  
  
    for (int i = 0; i < buffer.getQtdProdutos(); i++) {  
  
        ScopeRegistroCarteiraVirtual carteiraVirtual = buffer.getRegistroBandeira(i);  
  
        //Incluir tratamento para os registros retornados nos seguintes métodos:  
  
        // carteiraVirtual.getCodBandeira()  
        // carteiraVirtual.getSzNomeBandeira()  
        // carteiraVirtual.getCodRede()  
        // carteiraVirtual.getSzNomeRede()  
  
    }  
}
```

OBSERVAÇÃO: Todos os métodos e classes podem ser consultados pelo JavaDoc do Scope.

Apêndice F – Identificando a versão do SCOPE Client

Antes de atualizar o SCOPE Client ou reportar um problema para a NCR, muitas vezes é necessária não só a verificação da versão do SCOPE Server, mas também do SCOPE Client. Abaixo exibimos as diversas formas de verificar a versão do SCOPE Client.

Verificando no SCOPE Server

A verificação da versão do SCOPE Client no Server, cuja tela é exibida na Figura 9, é simples. Basta clicar no PDV cuja versão do SCOPE Client deseja-se consultar, e clicar no botão “PDV: <empresa>/<filial>/<PDV>”.

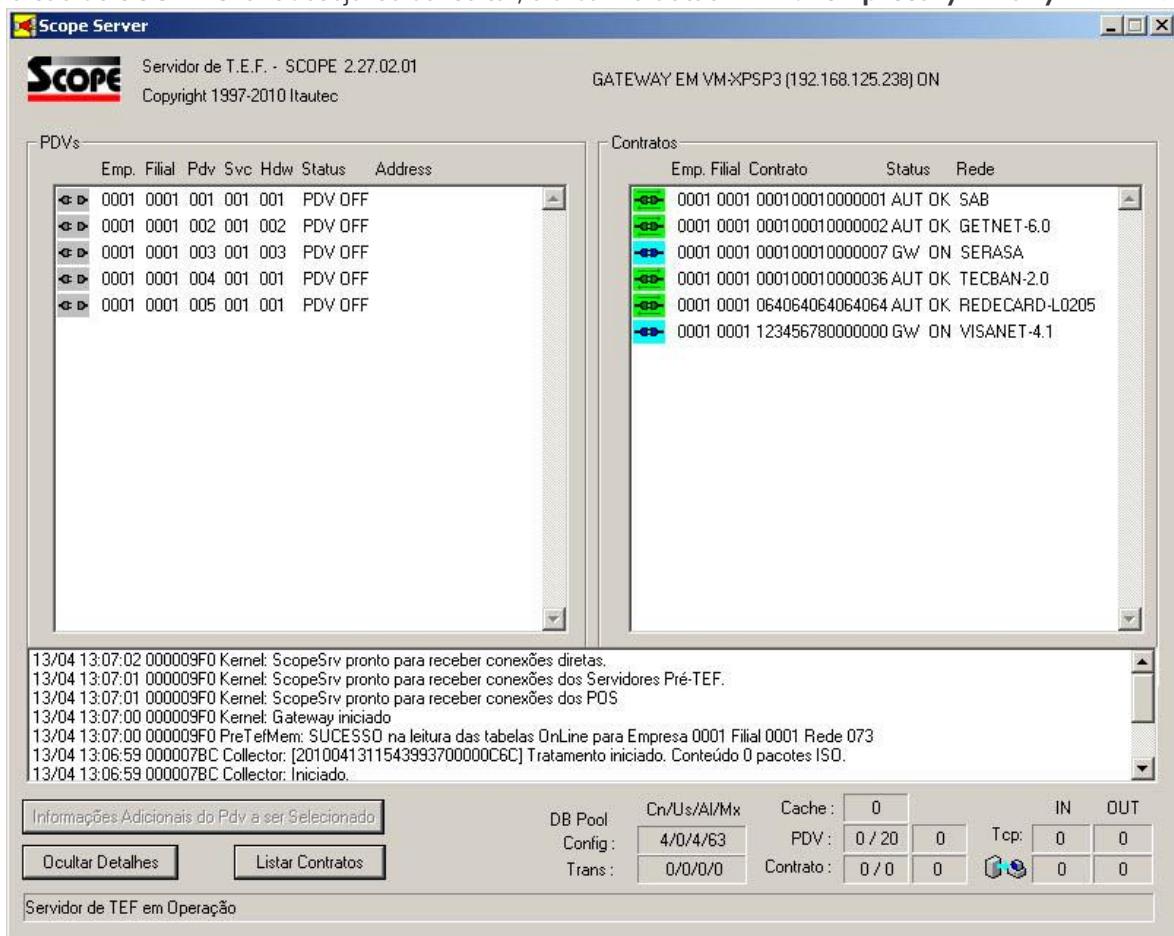


Figura 9: janela principal do servidor SCOPE

A janela similar a da Figura 10 será apresentada. Além de outras informações, nela está a versão do SCOPE Client representado no campo ScopeAPI (neste exemplo vemos 2.27.02.01). Percebe-se que embora tenha clicado num PDV específico na janela principal do servidor SCOPE, a janela com as informações do PDV possibilita a consulta de outros PDV's pelo campo “Emp. Filial Pdv”.

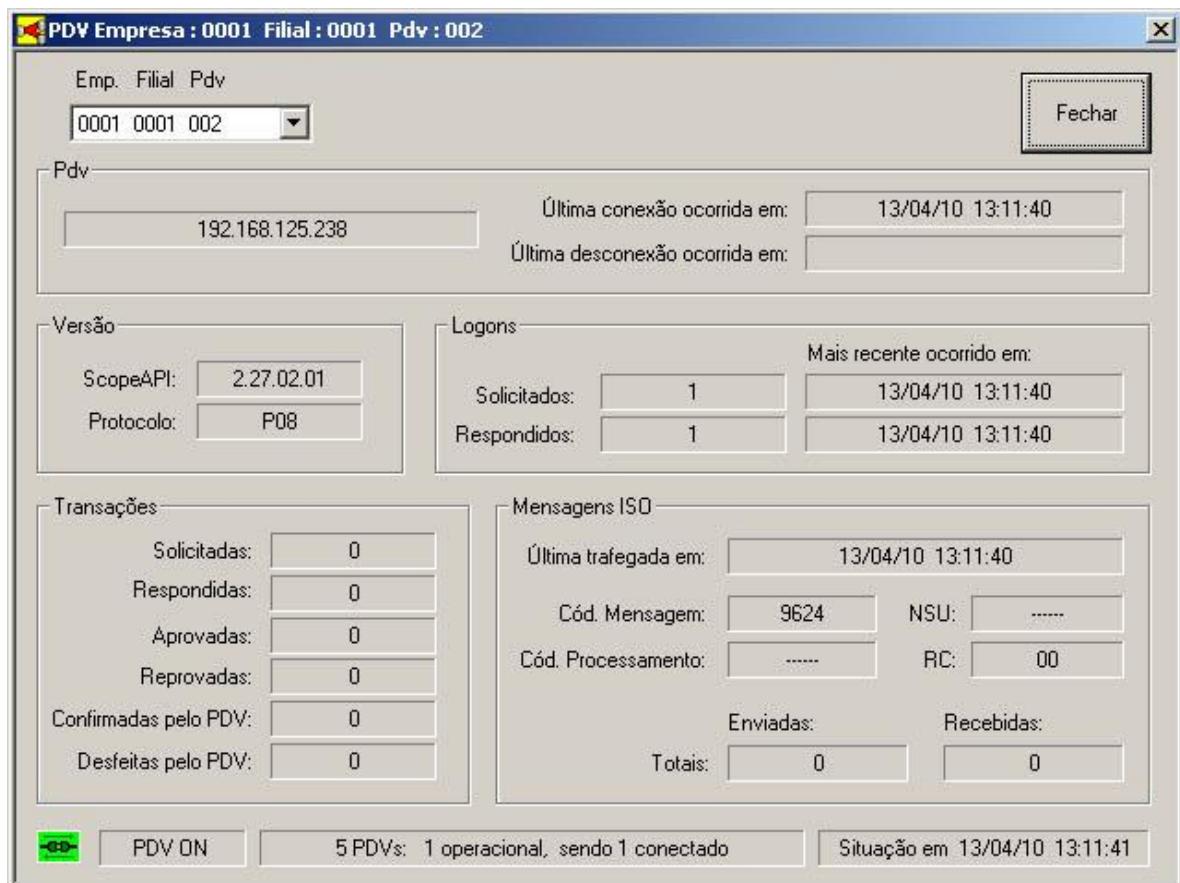


Figura 10: informações adicionais do PDV

IMPORTANTE: na consulta da versão do SCOPE Client não é preciso que este esteja conectado no servidor, mas é obrigatório que ele o tenha feito ao menos uma vez desde o *start* do servidor SCOPE.

Verificando no ambiente do PDV

Não é necessário ter acesso ao servidor SCOPE para consulta da versão do SCOPE Client e as vezes, por política da empresa, nem todo colaborador tem acesso ao servidor. Para estes casos, deve-se verificar a versão do SCOPE Client no próprio ambiente do PDV.

SCOPE Client para MS-Windows®

Para isso, deve-se conhecer o local de instalação das DLL's do SCOPE. Localize a biblioteca ScopeAPI.dll e abra a janela de propriedades do arquivo (clicando nela com o botão direito e na opção "Propriedades"). A janela "Propriedade de ScopeAPI.dll" será aberta. Clique na aba "Versão" e será possível visualizar informações como na Figura 11. No campo "Versão do arquivo:" encontra-se a versão do SCOPE Client (no exemplo vemos "2.27.2.1", que é a mesma do exemplo da sessão anterior).

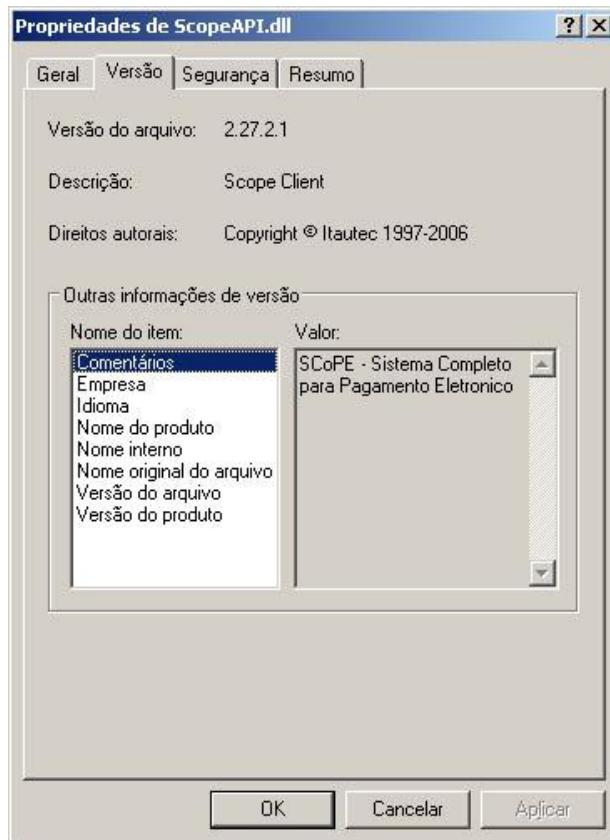


Figura 11: janela de propriedade de ScopeAPI.dll

SCOPE Client para Linux

Em Linux, no próprio nome das bibliotecas já está a versão. Elas normalmente são instaladas no diretório '/usr/lib' (ver [Instalação do SCOPE Client para Linux](#)), bastando a listagem dos arquivos do SCOPE. Execute:

```
$ ls libScope*
```

As bibliotecas e seus versionamentos são definidos da seguinte forma:

- libScopeApi.so.<a>.<bbb><cc>.<dd>
- libScopeCom.so.<a>.<bbb><cc>.<dd>
- libScopeJava.so.<a>.<bbb>.<e>

, onde:

- <a> é a versão da interface do SCOPE Client, cuja alteração deste implica em incompatibilidade entre as bibliotecas (exemplo: alteração de parâmetros de funções ou nomes de constantes). Este item faz parte do soname utilizado em Linux com shared libraries para prover informação de compatibilidade.
- <bbb> é a versão e o release do produto SCOPE. Esta versão é representada sem o ponto como, por exemplo, 225 é o SCOPE 2.25.
- <cc> é a versão do pacote do SCOPE. Perceba que não há ponto separador entre a versão, o release e o pacote.
- <dd> é o build do pacote.
- <e> é a versão da camada de interface Java.

Um exemplo da saída gerada por esse comando está na Figura 12. No exemplo, é apresentada a versão 2.25.07.01.

```
[root@localhost lib]# [root@localhost lib]# ls libScope*
libScopeApi.so          libScopeCom.so.1           libScopeSerial.so
libScopeApi.so.1          libScopeCom.so.1.22507.01  libScopeSerial.so.1
libScopeApi.so.1.22507.01 libScopeJava.so         libScopeSerial.so.1.02
libScopeCom.so            libScopeJava.so.1.225.08
[root@localhost lib]#
```

Figura 12: visualizando a versão do SCOPE Client na linha de comando do Linux

Apêndice G – Formato do Código de Barras InComm

Este apêndice tem como objetivo documentar o formato do código de barras da bandeira InComm.

O número do cartão para identificação da bandeira InComm pelo SCOPE pode ser obtido por meio de cartão magnético ou por código de barras.

A leitura e o tratamento do código de barras da InComm será realizado pela automação comercial, a qual enviará o número do PAN já formatado para o SCOPE. O layout do código de barras possui, sequencialmente, 11 dígitos referentes ao código do produto e 19 dígitos referentes ao número do cartão (PAN), o qual deve ser justificado à direita e preenchido com zeros à esquerda se for menor que 19 dígitos, como podemos observar na Figura 13.



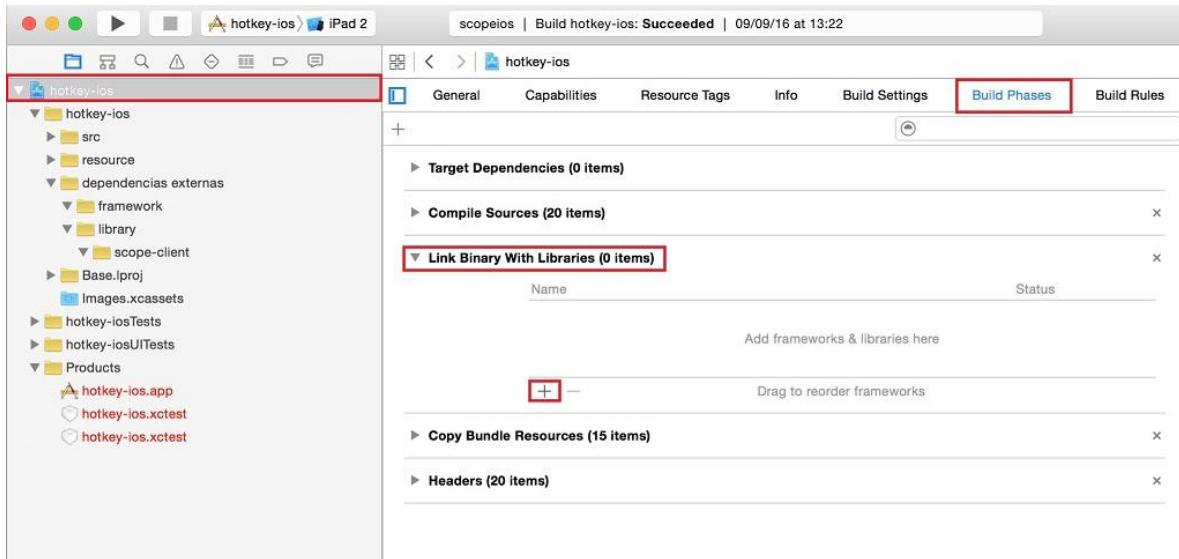
Figura 13: leiaute do código de barras da InComm

Após o código de barras ser lido e tratado pela automação, o PAN formatado será enviado para o SCOPE que irá entender o modo de entrada como cartão digitado.

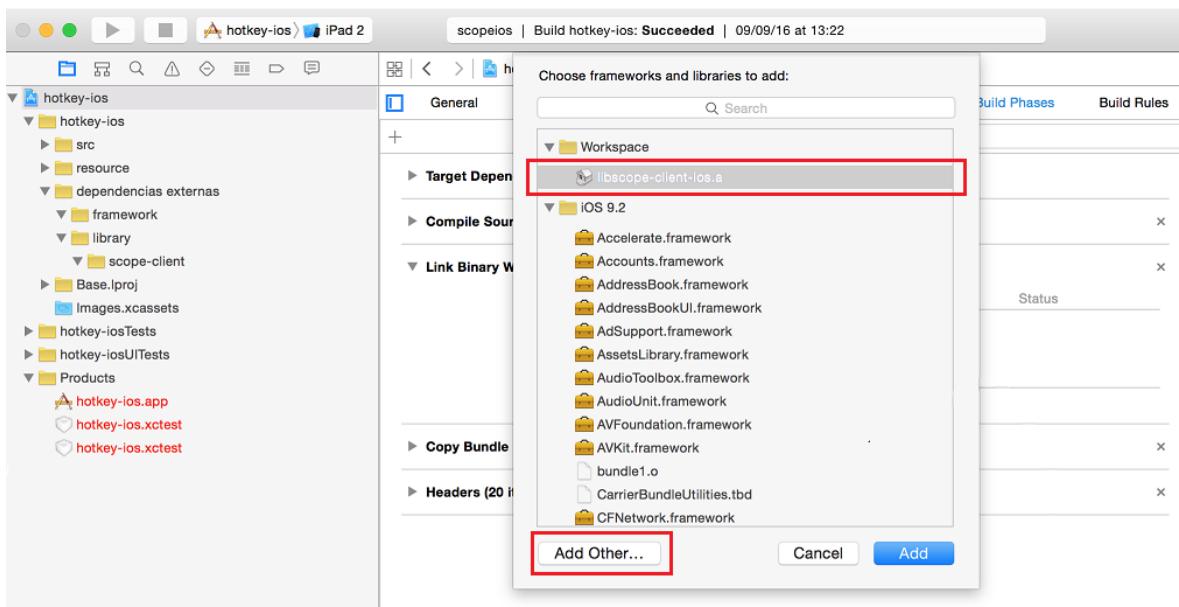
Apêndice H – iOS

Adicionando a biblioteca do SCOPE

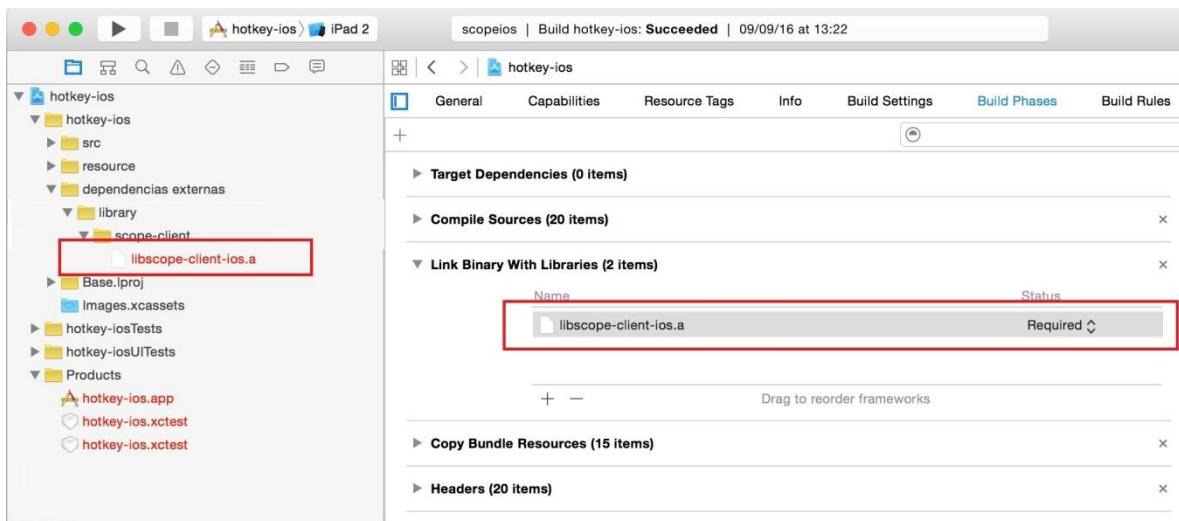
Após criado um projeto do tipo *.app, deverá ser realizado a importação da biblioteca estática do SCOPE (**libscope-client-ios.a**).



Será aberto uma nova janela para seleção da biblioteca, conforme imagem abaixo. Caso a biblioteca estiver no workspace selecione e pressione "Add", caso contrário, selecione a opção "Add Other...", localize a biblioteca em seu MAC e selecione "Add".

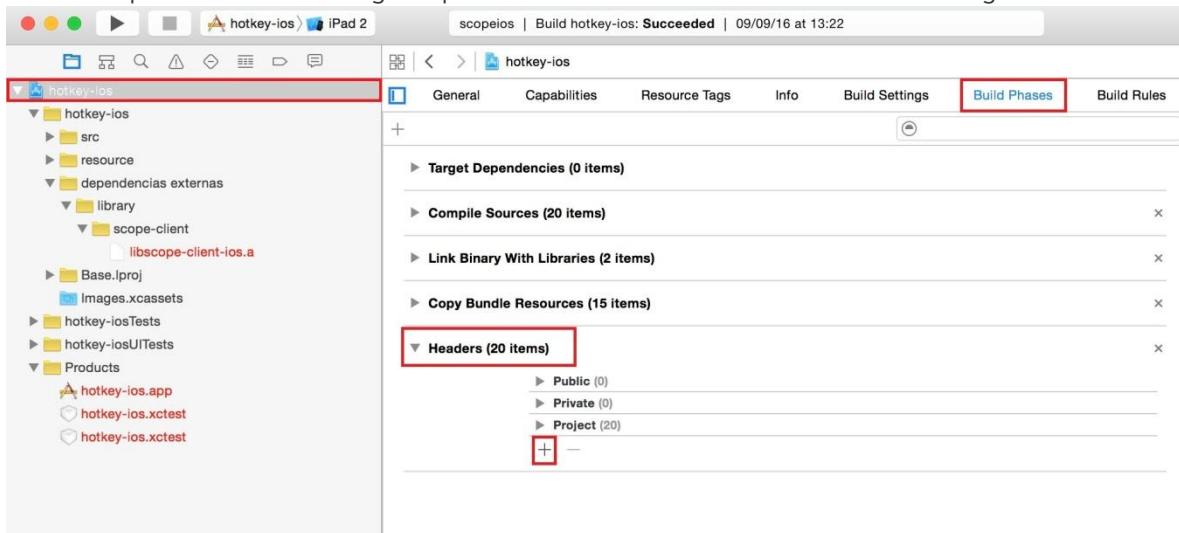


Após ser adicionado a biblioteca do SCOPE deverá ser apresentado o componente dentro do projeto e também na aba "Build Phases" no grupo "Link Binary...", conforme mostrado na imagem abaixo.



Adicionando o SCOPEAPI.H

Após adicionado à biblioteca do SCOPE, deve-se importar o header **scopeapi.h**. Este arquivo contém todas as definições e protótipos que permitem a manipulação das funções exportáveis do SCOPE Client iOS. Para importação deste arquivo é necessário seguir o procedimento demonstrado na imagem abaixo.

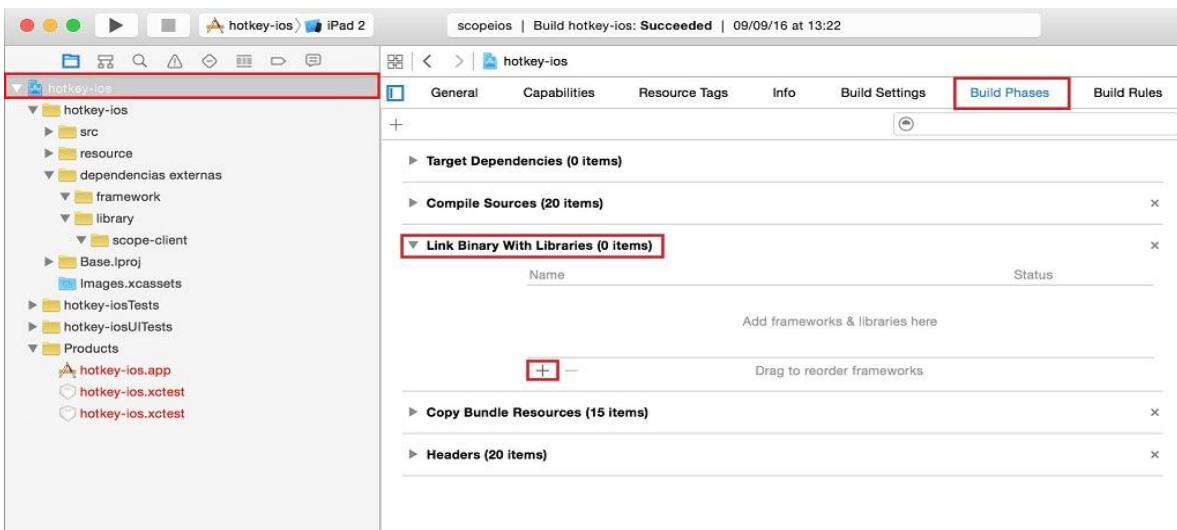


Aplicado o procedimento acima, é só procurar o arquivo **scopeapi.h** no MAC ou no workspace do projeto e adicioná-lo.

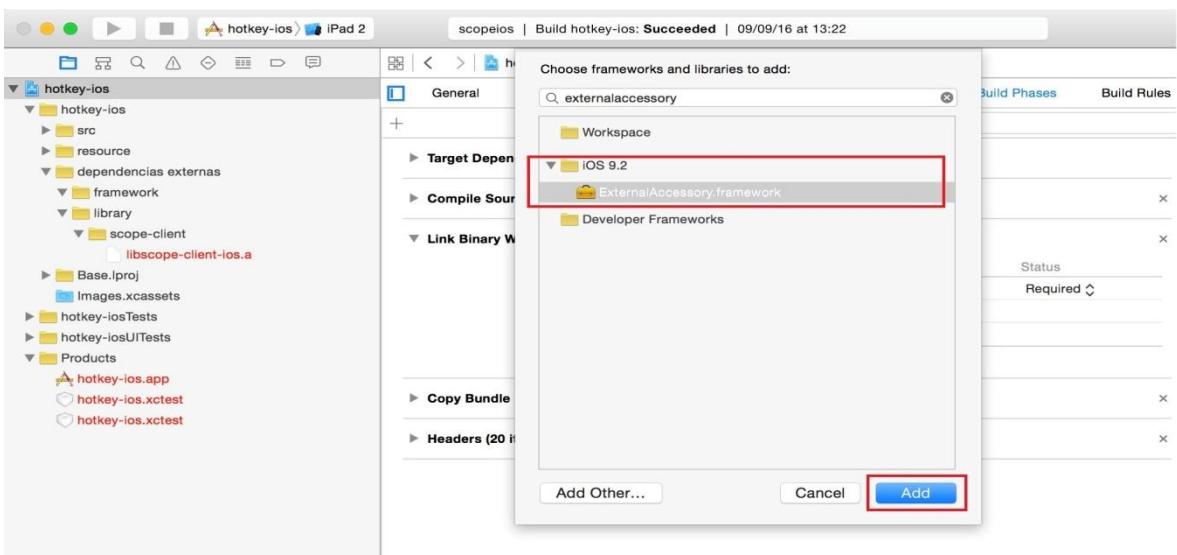
Importando o Framework ExternalAccessory

O framework *ExternalAccessory* fornece suporte para comunicação com hardware externo conectado a um dispositivo baseado em iOS utilizando como meios de comunicação: Apple Lightning, wireless ou bluetooth. As aplicações que utilizam este framework devem obrigatoriamente configurar os protocolos de hardware no

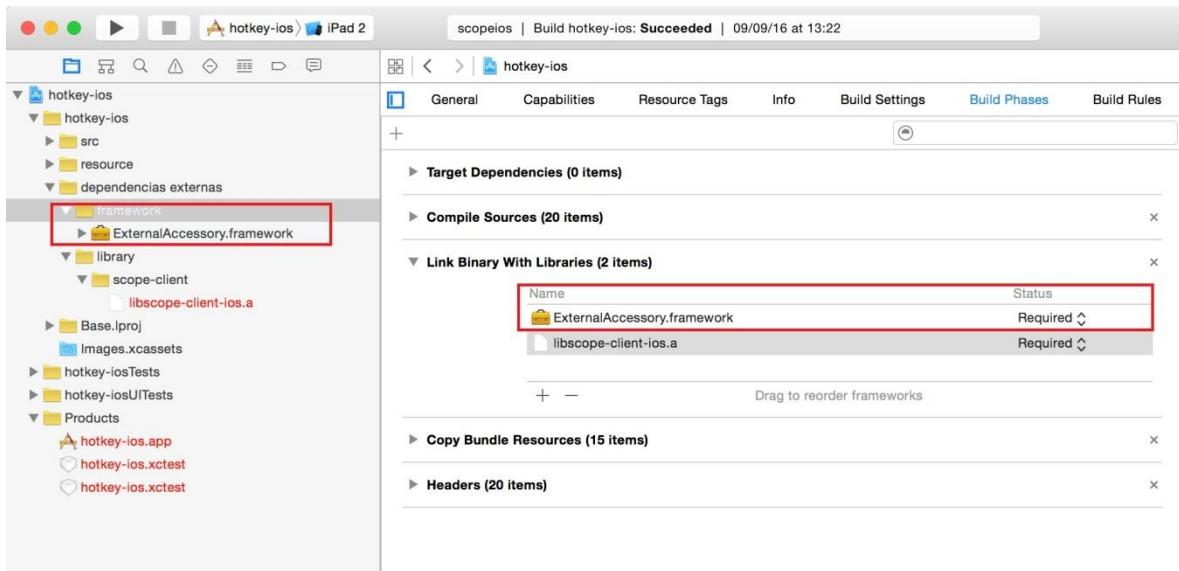
arquivo *info.plist* que indicam quais os dispositivos de hardware que sua aplicação conseguirá se comunicar. Na imagem abaixo é apresentado o procedimento para importação do framework.



Pesquisar pelo framework utilizando as palavras chaves "external" ou "externalaccessory". Após encontrado o componente, é só selecioná-lo e pressionar o botão "Add".



Adicionado o framework, o projeto deverá apresentá-lo da seguinte forma.

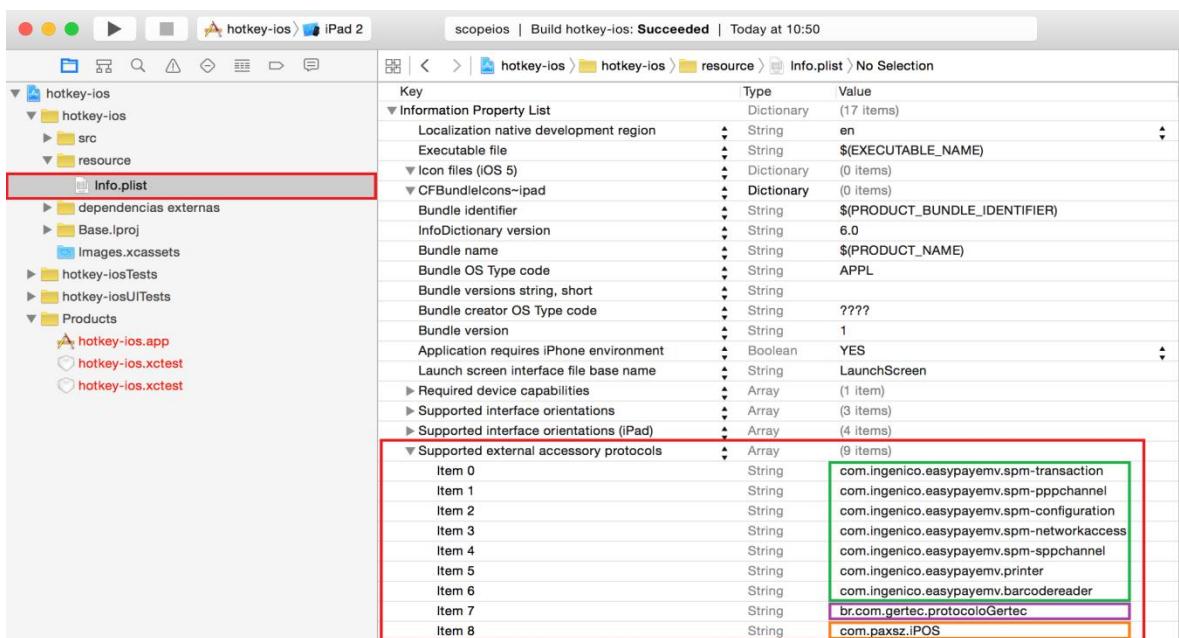


Configurando o arquivo INFO.PLIST

O `info.plist` é um arquivo XML que contém as configurações padrões de um projeto criado em ambiente iOS, como por exemplo, versão do aplicativo e dentre outras informações. Conforme citado acima, é necessário uma configuração no arquivo `info.plist` para que a aplicação identifique quais dispositivos de hardware serão permitidos se conectar por meio do bluetooth utilizando a biblioteca do SCOPE

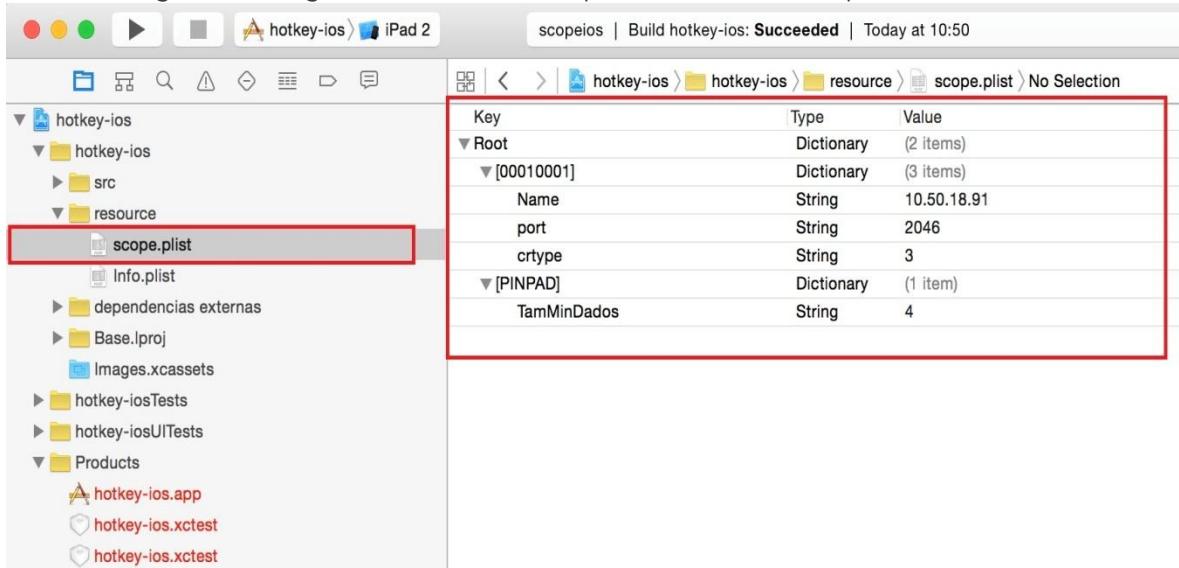
Adicionando os protocolos de hardware

Para configurar os dispositivos permitidos para conexão é necessário informar cada protocolo de hardware pertencente ao dispositivo dentro da chave "Supported external accessory protocols", conforme é apresentado na imagem a seguir.



SCOPE.PLIST

O scope.plist é um arquivo do tipo *property list* que permite a parametrização de configurações no SCOPE Client iOS. Este arquivo substitui em ambiente iOS o arquivo de configuração padrão do SCOPE: scope.ini. Para o correto funcionamento do Client esse arquivo deve ser criado obrigatoriamente, e adicionado à Bundle da aplicação. Abaixo segue uma imagem da estrutura do arquivo e como deve ser parametrizado.



A estrutura do arquivo scope.plist deve ser obedecida, conforme é apresentado na imagem acima, onde temos a seguinte hierarquia:

Estrutura do arquivo scope.plist:

- > **Root** (Type: Dictionary)
 - > **Sessão** (Type: Dictionary)
 - > **Chave** (Type: String)
 - > **Valor**

Exemplo de configuração conforme a estrutura acima:

- > Root (Não se altera este campo)
- > [00010001]
 - > Name
 - > 10.50.18.91

Apêndice I – POS Ingenico

Ferramentas para Desenvolvimento

Os softwares utilizados para desenvolvimento nesta plataforma, sendo eles o **IngeDev**, **LLT**, **SST** e **SAT**.

- IngeDev é a IDE (baseada em Eclipse) utilizada para desenvolvimento das aplicações na plataforma da Ingenico. Para o Tetra é permitido o desenvolvimento nas linguagens de programação C/C++.
- LLT é utilizado para acessar o terminal localmente, permitindo por exemplo, importar componentes, aplicações, DLLs e entre outros arquivos para o terminal (POS).
- SST e SAT são aplicações utilizadas no processo de assinatura dos componentes que serão depositados ao terminal. Uma assinatura de aplicações e de outros componentes serve como um método de autenticação realizado pela Ingenico para garantia que o terminal pode contar tal aplicação.

Módulos SCOPE POS

Na plataforma Telium o SCOPE é distribuído em quatro módulos, sendo elas as bibliotecas estáticas para manipulação dos arquivos de configuração do SCOPE e a biblioteca que define uma interface entre a aplicação e o SCOPE Client permitindo que a aplicação comercial utilize todas as funcionalidades do SCOPE Client inseridas em uma DLL, e por fim, uma aplicação open source utilizada para exemplificação no momento da integração ou como uma aplicação para pagamento.

SCOPE API POS

Módulo responsável por criar uma interface entre a aplicação comercial e o SCOPE Client, permitindo manipular todas as funcionalidades disponíveis na DLL do SCOPE. Todo o conjunto de funções estão disponíveis nesta interface, como as funções `ScopeCompraCartaoCredito`, `ScopeCompraCartaoDebito` e `ScopeCancelamento`.

- **Plataformas Disponíveis:**
 - Telium Tetra.
- **Tipo do componente:**
 - Biblioteca Estática (static library).
- **Funções disponíveis:**
 - Fornece todas as funções, estruturas e quaisquer outros recursos disponíveis no SCOPE Client.

SCOPE CONFIG POS

Módulo que permite a manipulação do arquivo de configuração do SCOPE (`scope.properties`) na plataforma Telium. O arquivo de propriedades do SCOPE possibilita que a aplicação forneça um método no qual o usuário possa configurar o SCOPE Client utilizando as mesmas chaves que são inseridas atualmente no `scope.ini` em outros sistemas operacionais suportados pelo SCOPE Client. O `scope.properties` não necessita de sessão, acatando apenas as chaves e valores que são utilizados pelo Client como a chave `Name` e `Port`. O número limite de propriedades deste configurador é de até 150 chaves.

As configurações são representadas por uma lista ligada chamada `LPScopeProperties`. A lista irá conter todas as propriedades já configuradas, caso não houver nenhuma, é definido por padrão as duas chaves obrigatórias do SCOPE: `Name (127.0.0.1)` e `Port (2046)`.

- **Plataformas Disponíveis:**
 - Telium Tetra.
- **Tipo do componente:**
 - Biblioteca Estática (static library).
- **Funções disponíveis:**

Função	Descrição
ScopeLoadPropertiesOfFile	Carrega do arquivo de configuração do SCOPE todas as propriedades definidas. O conteúdo carregado do arquivo virá com um formato TDTD (Tamanho-Dados) da chave e seu valor respectivamente. O buffer obtido com as propriedades serializadas nesta função pode ser convertido em uma lista do tipo <code>LPScopeProperties</code> por meio da função <code>ScopeDeserializeProperties</code> . As propriedades serializadas obtida neste processo deve sempre ser liberada da memória utilizando a função <code>ScopeCleanPropertiesBuffer</code> após o seu uso.
ScopeSavePropertiesOfFile	Salva a lista de propriedades <code>LPScopeProperties</code> no arquivo de configuração do SCOPE. Para quaisquer ações realizadas na lista de propriedades, para que sejam salvas, esta função deve ser invocada após a atualização e serialização da lista, o processo de serialização pode ser realizado pela função <code>ScopeSerializeProperties</code> .
ScopeAddNewProperty	Adiciona uma nova chave na lista de propriedades <code>LPScopeProperties</code> .
ScopeDeleteProperty	Remove uma chave da lista de propriedades <code>LPScopeProperties</code> .
ScopeFindProperty	Identifica na lista de propriedades uma chave e obtém o seu valor.
ScopeSerializeProperties	Serializa os dados contidos em uma lista de propriedades (<code>LPScopeProperties</code>) em um buffer de propriedades para que possam ser salvos no arquivo de configuração do SCOPE. Esta função só deve ser chamada caso houver a necessidade de salvar as propriedades no arquivo por meio da função <code>ScopeSavePropertiesOfFile</code> .
ScopeDeserializeProperties	Converte as propriedades serializadas (carregadas em memória do arquivo de configuração) em uma lista de propriedades <code>LPScopeProperties</code> . Esta

	função só deve ser chamada após a lista serializada de propriedades ser obtida pela função <i>ScopeLoadPropertiesOfFile</i> .
ScopeRestartPropertiesFile	Reseta as definições do arquivo de propriedades do SCOPE, removendo todas as propriedades já definidas e mantendo apenas as propriedades obrigatórias <i>Name</i> e <i>Port</i> .
ScopeIsPropertyExists	Verifica se existe uma chave configurada na lista de propriedades do SCOPE.
ScopeCleanAllProperties	Libera a lista de propriedades <i>LPScopeProperties</i> da memória.
ScopeCleanPropertiesBuffer	Libera o buffer contendo as propriedades serializadas obtidas do arquivo da memória.

SCOPE CLIENT POS

O SCOPE Client possui todas as funcionalidades/serviços disponíveis como é distribuído atualmente em outros sistemas operacionais, dentre os serviços principais estão disponíveis as funções *ScopeCompraCartaoCredito*, *ScopeCompraCartaoDebito* e *ScopeCancelamento*.

- **Plataformas Disponíveis:**
 - Telium Tetra.
- **Tipo do componente:**
 - Biblioteca Dinâmica (dynamic library - DLL).
- **Funções disponíveis:**
 - Todas as funções disponíveis na API do SCOPE.
- **Dependências:**
 - Biblioteca estática de configuração do SCOPE (*scope-config-lib*).
 - Biblioteca estática da API do SCOPE (*scope-api-lib*).

HOTKEY POS

O Hotkey é um aplicativo de exemplificação para os integradores do SCOPE Client, mas pode vir a servir como uma aplicação para execução das funções disponibilizadas pelo SCOPE.

- **Plataformas Disponíveis:**
 - Telium Tetra.
- **Tipo do componente:**
 - Aplicação (Application).
- **Dependências:**
 - Biblioteca estática de configuração do SCOPE (*scope-config-lib*).
 - Biblioteca estática da API do SCOPE (*scope-api-lib*).
 - Biblioteca dinâmica (ou DLL) do SCOPE Client (*scope-client-dll*).

Importando o SCOPE CLIENT para uma Aplicação

O procedimento de importação do SCOPE Client a uma aplicação no ambiente de desenvolvimento consiste apenas em introduzir duas bibliotecas estáticas que são responsáveis pela comunicação com a DLL e manipulação de seu arquivo de configuração.

Para efeito de organização e boas práticas, é aconselhável que se crie um diretório a parte dentro do projeto para as bibliotecas do SCOPE, mantendo o contexto do que é o projeto em desenvolvimento e o que foi importado para ele, como mostra a imagem a seguir que contém o diretório **Scope** e dentro deste diretório, subdiretórios para cada módulo que será importado ao projeto que faz parte da API do SCOPE, neste caso o **scope-api-lib** e **scope-config-lib**.

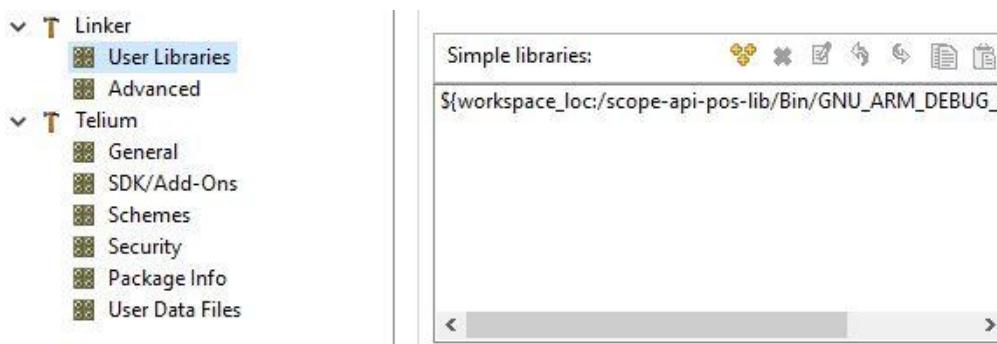


Não existe uma ordem para que seja feito a importação das bibliotecas do SCOPE, mas vamos mostrar neste documento a importação do **scope-api-lib** que servirá de base para a importação de qualquer outra biblioteca estática. Para importar uma biblioteca ao seu projeto no IngeDev, siga os passos a seguir:

Pressione o botão direito do mouse sobre o seu projeto e selecione a opção **Properties**.

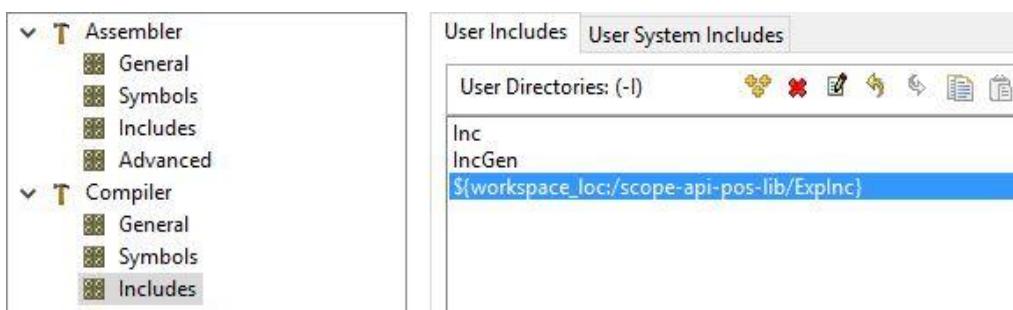
Nas propriedades do projeto irá aparecer várias opções. Selecione a opção **Telium Plus**, e em seguida, a opção **Build Configurations**.

O menu **Build Configurations** permite a configuração completa do projeto, como inclusão de bibliotecas, inclusão de arquivos header, configuração do nome do projeto, escolha do SDK para desenvolvimento, modos de compilação do projeto e diversas outras opções. Continuando o procedimento de inclusão da biblioteca **scope-api-lib**, selecione a opção **User Libraries**, visualize o quadro **Simple Libraries** à direta e pressione o botão simbolizado por três sinais de mais. Esta opção te permite selecionar a biblioteca estática que será importada ao seu projeto, o arquivo que representa uma uma biblioteca estática na plataforma Telium Tetra é um arquivo de extensão ***.a**, neste caso identifique no sistema de arquivos (*filesystem*) a biblioteca do scopeapi (**scope-api-lib.a**) e selecione-o, após a seleção irá aparecer o caminho da biblioteca conforme é demonstrado na imagem abaixo:



Após importado a biblioteca, é preciso importar seus arquivos de cabeçalho. No leque de opções do menu **Build Configurations** selecione a opção **Compiler** e a seguir a opção **Includes**.

Na já selecionada aba **User Includes** aparecerá o botão para adicionar o caminho dos arquivos de cabeçalho da biblioteca que estão representados pelo ícone com três sinais de mais. Identifique o caminho onde estão localizados os arquivos e selecione seu diretório. Para tornar mais simples a importação dos arquivos é preferível que o diretório com os arquivos esteja colocado no workspace do projeto. Após inserção dos arquivos o quadro User Directories deverá exibir o caminho onde está localizado os arquivos que foram importados, abaixo segue uma imagem mostrando um exemplo:



Após a inserção das bibliotecas estáticas no projeto, é importante que a DLL do SCOPE Client e alguns outros componentes estejam carregados no terminal. Para carregar no terminal a DLL é necessário que se tenha posse da aplicação **LLT** fornecida pela Ingénico.

Para adicionar a DLL do SCOPE Client no terminal siga os passos a seguir:

Abra o LLT e conecte o terminal via USB ao computador. O LLT indicará automaticamente na aba **Plugged terminals** quando o terminal estiver conectado e pronto para estabelecer uma conexão, conforme mostra a imagem abaixo:

Plugged terminals		Transfer tasks		
S...	Device	Telium range	Description	Identifier
	COM6	Telium 2	Telium legacy generic	usbcdcacm\VID_07
	COM16	Telium Tetra	Move5000	usbcdcacm\VID_0E

Antes de conectar pelo LLT o terminal deve estar em **Modo LLT** (estado para transferência/recepção de dados). Neste exemplo será mostrado como colocar o terminal Move5000 em modo LLT:

Pressione o botão de menu (botão **F**) do terminal.

Selecione a opção **Control Panel**.

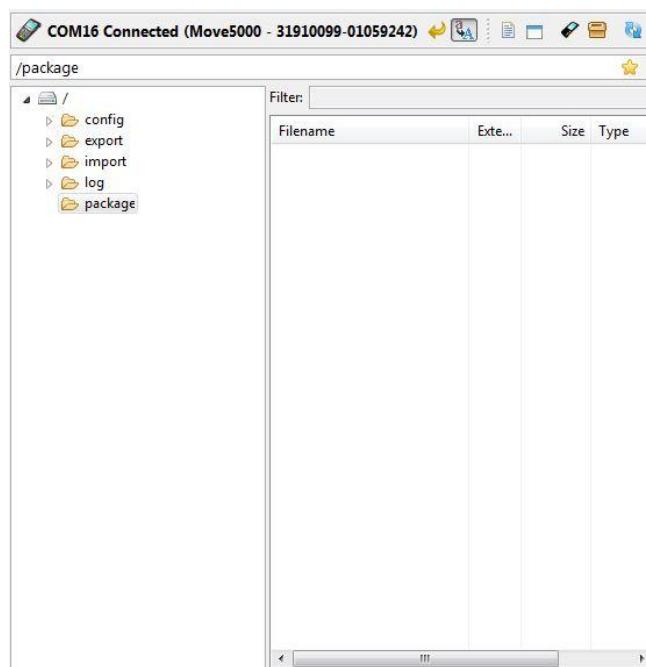
Selecione a opção **Software Management**.

Selecione a opção **Evolution** e em seguida a opção **Local upgrade** caso o terminal estiver conectado via USB.

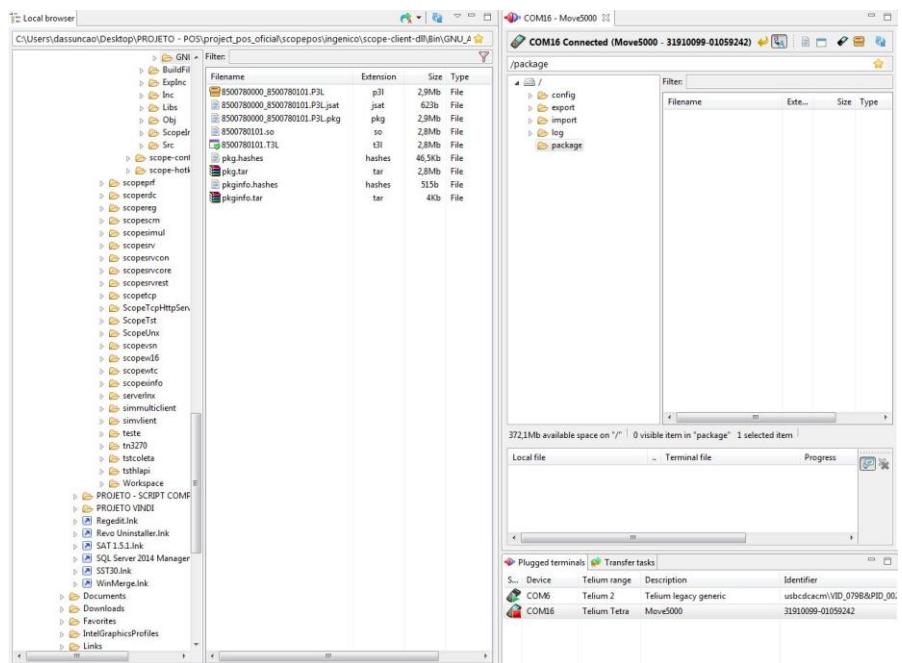
O terminal deverá exibir uma mensagem **Connection in progress**.

Após o procedimento anterior, selecione com duplo clique na aba **Plugged terminals** no LLT o terminal no qual será conectado, neste caso, o terminal que está em modo LLT.

No LLT aparecerá a seguinte tela:



Executado todo o procedimento anterior com sucesso, o terminal estará apto a receber qualquer componente no qual for passado para ele. No lado esquerdo está situado a aba **Local browser** contendo todo o sistema de arquivos do computador e ao lado direito o sistema de arquivos do terminal.



Para envio da DLL para o terminal identifique onde ela está localizada no computador na aba esquerda do LLT (**Local browser**) e a seguir selecione a opção na aba do terminal sinalizada na imagem abaixo:



Esta opção faz com que o LLT identifique automaticamente onde um componente, por meio de sua extensão, deve ser dirigido ao sistema de arquivos do terminal. No exemplo de um terminal com o sistema operacional Telium Tetra, o sistema de arquivos é constituído pelos seguintes diretórios:

config: Estão situados todos os arquivos de configuração utilizados pelo terminal.

export: Exibe alguns e arquivos componentes que foram importados ao terminal.

import: Diretório destinado a receber qualquer tipo de arquivo que seja diferente de um "executável" na plataforma Telium. Como por exemplo, um arquivo ***.txt**. Após o envio de arquivos para o terminal que são direcionados à este diretório, quando o terminal reinicia, todos estes arquivos são transferidos para o diretório **export**.

log: Diretório destinado à locação de todos os arquivos log que são gerados pelo terminal ou aplicações que são executadas no terminal e salvam seus arquivos log dentro deste diretório.

package: Diretório para onde são enviados os arquivos do tipo aplicação, DLL ou qualquer outro que faça parte deste grupo.

Após todo o procedimento anterior tendo sido executado com sucesso, é só identificar o pacote com o SCOPE Client e importar o arquivo ***.P3L** ao terminal, simplesmente arrastando o arquivo na aba **Local browser** (situada no lado esquerdo da tela do LLT) para a aba do terminal (situada no lado direito da tela do LLT).

Abaixo o exemplo do pacote do SCOPE para a plataforma Telium Tetra:

 8500780000_8500780101.P3L	01/06/2017 09:57	Arquivo P3L	2.942 KB
 8500780000_8500780101.P3L.jsat	01/06/2017 09:57	Arquivo JSAT	1 KB
 8500780000_8500780101.P3L.pkg	01/06/2017 09:57	Arquivo PKG	2.936 KB
 8500780101.so	01/06/2017 09:57	Arquivo SO	2.880 KB
 8500780101.T3L	01/06/2017 09:57	Arquivo T3L	2.880 KB
 pkg.hashes	01/06/2017 09:57	Arquivo HASHES	47 KB
 pkg.tar	01/06/2017 09:57	WinRAR archive	2.881 KB
 pkginfo.hashes	01/06/2017 09:57	Arquivo HASHES	1 KB
 pkginfo.tar	01/06/2017 09:57	WinRAR archive	4 KB

Feito isto, o terminal identifica automaticamente onde a DLL deve ser depositada e indica abaixo da aba do terminal no LLT o percentual de execução do **Download** (importação de arquivos para o terminal). Após efetuado o Download, apenas desconecte o terminal do LLT clicando no terminal conectado na aba **Plugged terminals** e após a desconexão ele irá reiniciar caso houver a necessidade, neste caso o terminal irá reinicializar.

Apêndice J - Glossário

A

Acquirer: veja **Adquirente**

Adquirente: é a entidade ou associação de entidades financeiras que a partir de transações efetuadas com cartões nos estabelecimentos comerciais (Merchant) associados, estabelece o devido vínculo com as entidades autorizadoras (authorizing agent) (VISA, MasterCard, Amex).

Aplicação frente de caixa: software executado no PDV com a finalidade de realizar a venda de produtos da empresa.

Aplicação de PDV: veja *Aplicação frente de caixa*.

Authorizing agent: veja **Autorizador**

Autorizador: é a organização (VISA, MasterCard, Amex) que gerencia e controla operações com cartões de crédito, passando informações entre o Adquirente e o Banco emissor.

B

Banco de dados: também conhecido como base de dados, são arquivos ou sistemas com uma estrutura regular que organizam informações. Essas estruturas podem ter a forma de uma tabela: cada tabela é composta por linhas e colunas. As informações utilizadas para um mesmo fim são agrupadas em uma base de dados.

Bandeira: entidade detentora de marcas e logotipos utilizados em cartões de crédito, débito e outros meios de pagamentos. Exemplos: Visa, Mastercard, American Express, Visa Electron, Maestro, Cheque Eletrônico.

Banco emissor: é a entidade financeira associada a uma ou mais organizações autorizadoras e que é responsável pela emissão de cartões para seus clientes.

BIN (bank identification number): número de identificação do banco representado pelos primeiros 6 dígitos do cartão.

E

Endereço IP (Internet Protocol): trata-se de uma tecnologia que permite a comunicação padronizada entre computadores, mesmo que estes sejam de plataformas diferentes, cada máquina possui um endereço IP que a diferencie das demais.

Estabelecimento: ou estabelecimento comercial é a entidade que aceita o cartão (card acceptor) como forma de pagamento referente à comercialização de um bem ou serviço prestado ao portador do cartão (card holder).

G

Grupo de serviço: um serviço no SCOPE pertence sempre a um grupo, que pode definir a forma de pagamento, ou ainda, a tecla finalizadora do PDV. Exemplos: cartão de crédito, cartão de débito, consulta de cheque, recarga de celular, estorno. Ver Serviço.

GUI: do inglês '*Graphic User Interface*' que é a interface gráfica exibida para o usuário da aplicação.

I

IATA: International Air Transportation Association

Issuer: Veja Banco emissor.

P
PDV: acrônimo para 'Ponto de Venda'. Terminal inteligente utilizado na operação de pagamento. Veja também PoS.

PoS: termo em inglês 'Point of Sale'. Veja também PDV.

R
Rede autorizadora: empresa que concentra o recebimento de transações TEF de diversos estabelecimentos, e as autoriza através de um sistema autorizador. Exemplos: Visanet, REDE, TecBan.

S
SCOPE: O SCOPE é a solução Itautec para pagamentos eletrônicos

ScopeCNF (SCOPE Configurador): é o módulo responsável pela configuração e cadastramento de parâmetros para a solução SCOPE.

SCOPE Client: conjunto de bibliotecas localizadas na máquina em que a aplicação de frente de loja. O

conjunto de bibliotecas pode variar conforme o sistema operacional e a linguagem de programação utilizada.

ScopeGW (SCOPE Gateway): funciona como um roteador de mensagens entre um ou mais servidores SCOPE e uma ou diversas redes.

ScopeSRV (SCOPE Server): é o módulo principal da solução SCOPE. É responsável por estabelecer o contato inicial de todos os contratos cadastrados na base de dados com as respectivas redes, garantindo também o fluxo de transações com estas redes

Serviço: o termo serviço no contexto do SCOPE define o objetivo de uma transação. Exemplos: compra com cartão de crédito à vista, compra com cartão de crédito parcelada pelo estabelecimento (sem juros), compra com cartão de crédito parcelada de administradora (com juros), compra com cartão de débito à vista, compra com cartão de débito pré-datado, compra com cartão de débito voucher, consulta cheque, estorno crédito, estorno débito.

Servidor: é o computador que administra e fornece programas e informações para os outros computadores conectados em rede.

SGBD: Sistema de Gerenciamento de Banco de Dados. É o conjunto de programas de computador (softwares) que controlam a criação, manutenção e uso de Bases de Dados.

T
TSR: abreviação do termo em inglês '*Terminate and Stay Resident*'. Veja Módulo residente.

Revisões

Histórico de Alterações:

Revisão	Data	Alteração
1.218	31/03/2025	<ul style="list-style-type: none">Inclusão do tópico PIN ImpressoInclusão da máscara Cod_Produto_Digital no tópico de dados disponíveis das transações.Inclusão do código de serviço Solicitação de PIN Impresso.Inclusão do código de erro do SCOPE para dados incompletos.Inclusão dos estados de coleta estendidos para seleção de provedor, seleção de produto digital e coleta de produto digital.
1.217	18/03/2025	<ul style="list-style-type: none">Alteração no Apêndice E – Android. Inclusão de tratamento para terminal POS Positivo L400.
1.216	19/02/2025	<ul style="list-style-type: none">Alteração no Apêndice E – Android. Inclusão de tratamento para terminal POS e revisão deste apêndice.
1.215	19/02/2025	<ul style="list-style-type: none">Alterados tópicos sobre as funções ScopeOpen e ScopeClose, incluindo recomendação para realizar conexão por demanda.Alteração no fluxograma do tópico Sessão de Transação, refletindo a recomendação de realizar conexão por demanda.
1.214	27/01/2025	<ul style="list-style-type: none">Inclusão das funções ScopeStartLog e ScopeStopLog responsáveis pelo carregamento e liberação da biblioteca de tratamento de logs csmsg.dll.
1.213	02/01/2025	<ul style="list-style-type: none">Inclusão de exemplo de seleção de moeda para DCC Digitado da Cielo R2024
1.212	30/12/2024	<ul style="list-style-type: none">Inclusão de exemplo de chamada de método obtemListaCarteiraVirtual no apêndice E – Android.
1.211	16/10/2024	<ul style="list-style-type: none">Revisão das funções de Reimpressão de Comprovante;A função ScopeReimpressaoComprovante pode ser adotada como padrão;As antigas ScopeReimpressaoComprovantePagamento, ScopeReimpressaoOffLine e ScopeReimpressaoOnLine, embora mantidas por compatibilidade, estão obsoletas.
1.210	30/09/2024	<ul style="list-style-type: none">Cielo R2022 alterada para R2024
1.209	06/09/2024	<ul style="list-style-type: none">Inclusão de informações sobre Ispb da instituição pagadora para transações Pix.
1.208	27/08/2024	<ul style="list-style-type: none">Correção da informação do range de valores retornado pelo Scope Client através da função ScopeStatus durante a transação. Página 33.
1.207	02/07/2024	<ul style="list-style-type: none">Inclusão de versão de especificação PagSeguro 2.0.4Inclusão da bandeira DINERS DEBInclusão do serviço Crédito para Pagamento de Fatura

		<ul style="list-style-type: none"> Inclusão do serviço Consulta Pré-Autorização Inclusão da função ScopeCreditoPagamentoFatura Inclusão da função ScopeConsultaPreAutorizacao Inclusão da Máscara 4, bit 0x00080000
1.206	28/05/2024	<ul style="list-style-type: none"> Atualização dos campos do comando ScopePPStartGetData disponíveis para ABECS 2.12
1.205	14/05/2024	<ul style="list-style-type: none"> Atualização de alguns códigos de retornos do client
1.204	04/04/2024	<ul style="list-style-type: none"> Atualização do tópico sobre o Client Android para melhor entendimento.
1.203	22/03/2024	<ul style="list-style-type: none"> Alteração do nome da rede TENDENCIA para DBR Inclusão das bandeiras BL-BRADESCO, MEGAVALECARD CREDITO, ELO VOUCHER PAT e MEGAVALECARD DEBITO
1.202	12/12/2023	<ul style="list-style-type: none"> Cielo R2022: Crediário e Parcelado Cliente
1.201	29/11/2023	<ul style="list-style-type: none"> Revisão do tópico sobre pinpad Bluetooth no apêndice "Android". Inclusão do tópico sobre pinpad USB no apêndice "Android".
1.200	30/10/2023	<ul style="list-style-type: none"> Inclusão da versão de especificação da MURY Lojista 3.03. Inclusão das bandeiras MURY CRED e MURY VOUCHER. Alteração do nome da rede FOXWIN para MURY.
1.199	15/08/2023	<ul style="list-style-type: none"> Adicionado o item a respeito do carregamento dinâmico da API do Scope Client.
1.198	06/07/2023	<ul style="list-style-type: none"> Alteração do valor do campo Status DCC para 0x00040000. Inclusão do novo campo do ObtemCampos, Máscara 4, Tamanho do BIN. Inclusão da versão de especificação da SOLUCARD Lojista 3.03. Inclusão das bandeiras SOLUCARD CRED, SOLUCARD BEN. Inclusão da versão de especificação da CONNECT Lojista 3.03.
1.197	21/06/2023	<ul style="list-style-type: none"> Inclusão da versão de especificação da USECRED Lojista 3.03. Inclusão das bandeiras USECRED ALIM, USECRED REF, USECRED COMB e USECRED BEN. Alteração do nome da bandeira USECRED para USECRED CRED.
1.196	24/04/2023	<ul style="list-style-type: none"> Atualização para remover espec. v6.01 da BRADESCARD, mantendo v6.02.
1.195	13/02/2023	<ul style="list-style-type: none"> Inclusão da versão de especificação da DM Lojista 3.03
1.194	18/01/2023	<ul style="list-style-type: none"> Crediário para REDE LO701: Novo formato stDadosCreditorioCredito para ScopeObtemDadosCreditorioCredito
1.193	12/12/2022	<ul style="list-style-type: none"> Inclusão das redes BKBANK e BSCASH. Inclusão da versão de especificação da BKBANK Lojista 3.03. Inclusão da versão de especificação da BSCASH Lojista 3.03. Inclusão das bandeiras BKBANK BEN e BSCASH CRED.
1.192	28/11/2022	<ul style="list-style-type: none"> Inclusão da versão de especificação da CrediShop Lojista 3.03
1.191	21/11/2022	<ul style="list-style-type: none"> Inclusão de tópicos sobre a Recarga de Celular BRADESCO e Modo Compatibilidade
1.190	10/11/2022	<ul style="list-style-type: none"> Novos campos do ObtemCampos, Máscara 4, para DCC – Conversão Dinâmica de Moedas Novos Serviços de Consulta DCC Crédito e Débito

1.189	04/11/2022	<ul style="list-style-type: none"> Inclusão da rede ALGORIX-EMV Inclusão da versão de especificação da Algorix EMV Lojista 3.03 Inclusão das bandeiras UZE CRED e UZE VOUCHER
1.188	29/09/2022	<ul style="list-style-type: none"> Atualização para rede SMARTNET
1.187	22/08/2022	<ul style="list-style-type: none"> Inclusão do parâmetro PPTLILOTE no arquivo scope.ini
1.186	17/05/2022	<ul style="list-style-type: none"> Inclusão da nova estrutura de lista de mercadorias genéricas (LM01) que contempla um maior número de campos quando for necessário enviar uma lista de SKU's mais completa para as autorizadoras. Estrutura montada no estado de coleta TC_COLETA_LISTA_MERCADORIAS (0xFCCB).
1.185	06/05/2022	<ul style="list-style-type: none"> Inclusão das funções ScopeCompraCartaoCreditoMarketplace e ScopeCompraCartaoDebitoMarketplace, para GetNetLAC 2.99, conforme Melhoria 64318.
1.184	14/03/2022	<ul style="list-style-type: none"> Inclusão da rede Seicon Inclusão da versão de especificação da Seicon Lojista 3.03. Inclusão dos serviços Consulta Parâmetros Débito e Consulta Parâmetros Crédito Inclusão das bandeiras SEICON CRED e SEICON VOUCHER para serem reteadas para rede SEICON.
1.183	07/03/2022	<ul style="list-style-type: none"> Atualização para espec. v6.01/v6.02 da BRADESCARD no tópico Redes com tratamentos específicos.
1.182	22/02/2022	<ul style="list-style-type: none"> Inclusão das seções SCOPELOGAPI, SCOPELOGPRF e SCOPELOGSRL no arquivo scope.ini para a geração de logs no Scope Client utilizando a biblioteca csmsg para os sistemas Windows e Linux.
1.181	17/02/2022	<ul style="list-style-type: none"> Inclusão da lista de valores do modo de entrada (Entry Mode) da Máscara Modo_Estrada (0x00000008) no Apêndice A.
1.180	17/12/2021	<ul style="list-style-type: none"> Alteração no requisito de tamanho mínimo do buffer que recebe a lista de Carteiras Virtuais para 1024 bytes, para evitar que aplicações passem o tamanho anterior de 618 bytes e provoque o retorno de um erro da função ScopeMenuRecuperaltens.
1.179	12/11/2021	<ul style="list-style-type: none"> Inclusão da função setConfig para a plataforma Android.
1.178	26/10/2021	<ul style="list-style-type: none"> Remoção da Interface HLAPI.
1.177	18/10/2021	<ul style="list-style-type: none"> Atualização das informações sobre Carteira Virtual
1.176	07/10/2021	<ul style="list-style-type: none"> Renomeado Rede/Bandeira PrevSaúde para ORIZON.
1.175	30/09/2021	<ul style="list-style-type: none"> Detalhamento dos tipos de convênios retornados na função ScopeObtemMedicamentosComCRM().
1.174	30/09/2021	<ul style="list-style-type: none"> Inclusão das bandeiras Lecard para Algorix.
1.173	30/09/2021	<ul style="list-style-type: none"> Inclusão da versão de especificação da Stone 4.00. Inclusão das bandeiras para compor a versão de especificação da Stone 4.00.
1.172	30/09/2021	<ul style="list-style-type: none"> Inclusão das bandeiras CEAPAY e CONNECT CRED para serem reteadas para rede CONNECT.
1.171	30/09/2021	<ul style="list-style-type: none"> Inclusão das bandeiras Maxxcard e Redecompras para serem roteadas para Cielo.
1.170	15/09/2021	<ul style="list-style-type: none"> Revisão dos detalhes da BRADESCARD no tópico Redes com tratamentos específicos.

1.169	12/09/2021	<ul style="list-style-type: none"> Inclusão da Especificação 3.03 para a rede REDESOFTNEX Inclusão de Bandeiras para a rede REDESOFTNEX
1.168	11/08/2021	<ul style="list-style-type: none"> Acréscimo do parâmetro ColetasOpcionais na seção SCOPEAPIPOS do arquivo scope.ini
1.167	12/07/2021	<ul style="list-style-type: none"> Inclusão da função ScopeConsultaPgtoFaturaCPF; Inclusão da função ScopeConsultaSaldoCreditoCPF; Inclusão da função ScopeTransacaoFinanceiraCPF; Inclusão da função ScopePagamentoCPF; Revisão dos detalhes da BRADESCARD no tópico Redes com tratamentos específicos.
1.166	30/06/2021	<ul style="list-style-type: none"> Inclusão do código de erro: 60929 (0xEE01) - Timeout Server-NA
1.165	23/06/2021	<ul style="list-style-type: none"> Inclusão de novos obtemCampos: campo tid e campo referencia
1.164	04/06/2021	<ul style="list-style-type: none"> Inclusão da função ScopePPGetCOMPort
1.163	14/05/2021	<ul style="list-style-type: none"> Inclusão da função ForneceCampoEx com parametro de tamanho Inclusão do tratamento para RAPPI
1.162	15/04/2021	<ul style="list-style-type: none"> Inserido parâmetro RSAMaxTentativas na sessão [EmpresaFilial] do scope.ini
1.161	17/03/2021	<ul style="list-style-type: none"> ScopeConsultaPgtoFatura. Revisão dos detalhes da BRADESCARD no tópico Redes com tratamentos específicos.
1.160	03/03/2021	<ul style="list-style-type: none"> Criação da versão de especificação 2.17.1 Novos códigos para o ScopeForneceCampo (SCOPE_SOFT_DESCRIPTOR_135 e SCOPE_MCC)
1.159	01/03/2021	<ul style="list-style-type: none"> Inclusão da função ScopeSaque no tópico Funções Diversas Inclusão da rede Hub no tópico Redes com tratamentos específicos
1.158	19/01/2021	<ul style="list-style-type: none"> Inclusão de uma observação mais detalhada na utilização correta dos comandos multimídia.
1.157	15/01/2021	<ul style="list-style-type: none"> Inclusão de informações sobre Modo de Pagamento para Carteira Virtual.
1.156	08/10/2020	<ul style="list-style-type: none"> Inclusão da função ScopeCompraCartaoCreditoCPF. Revisão dos detalhes da BRADESCARD no tópico Redes com tratamentos específicos.
1.155	18/09/2020	<ul style="list-style-type: none"> Inclusão dos comandos multimídia para os PINpad ABECS.
1.154	14/09/2020	<ul style="list-style-type: none"> Inclusão de tratamento para Estados de Coleta Estendidos
1.153	17/07/2020	<ul style="list-style-type: none"> Acréscimo da Rede CONNECT Acréscimo da Bandeira BAHAMAS ELO CRÉDITO Criação da execução Especificação 3.02 (versão inicial da Connect).
1.152	06/08/2020	<ul style="list-style-type: none"> Inclusão de detalhes da rede BRADESCARD no tópico Redes com tratamentos específicos
1.151	08/07/2020	Adequação Padrão Visual NCR
1.150	08/05/2020	<ul style="list-style-type: none"> Inseridos parâmetros de configuração no tópico Sessão [SCOPEAPI]
1.149	15/05/2020	<ul style="list-style-type: none"> Rec. e exemplo de uso da função ScopelniciaTotalTEF Inclusão de capítulo "Perguntas Frequentes"

1.148	09/03/2020	Correção nas constantes do comando ScopePPStartGetData.
1.147	03/02/2020	GetNetLAC 2.99A: •Novo campo OrderId em stSplitPagamentoF01, devido Mudança de Requisito V2.99A.
1.146	03/02/2020	GetNetLAC 2.99A: •Observação adicional na função ScopeObtemDadosCredarioCredito
1.145	03/02/2020	GetNetLAC 2.99A: •Nova coleta TC_COLETA_SPLIT_PAGAMENTO (0xFCF7) •Novo código para ScopeForneceCampo SCOPE_DADOS_SPLIT_PAGAMENTO (39) •Novo tópico sobre Split de Pagamento para GetNetLAC
1.144	03/02/2020	GetNetLAC 2.99A: •Inclusão da função ScopeSimulacaoCredario como opção para Compra com Cartão de Crédito. •Nova função ScopeObtemDadosCredarioCredito. •Nova coleta TC_DECIDE_CREDARIO (0xFCF5) •Nova coleta TC_DECIDE_NOVA_CONSULTA (0xFCF8) •Novo código para ScopeForneceCampo SCOPE_DADOS_CREDARIO (40).
1.143	29/01/2020	<ul style="list-style-type: none"> Atualização do tópico Status de transação
1.142	21/01/2020	<ul style="list-style-type: none"> Criação da versão de especificação V5.1.0 para a ePharma Atualização da sessão “Compra de medicamento” a respeito do funcionamento da nova versão de especificação da epharma versão 5.1.0 Atualização das informações do parâmetro NumCpFiscal da função ScopeCompraMedicamento
1.141	19/11/2019	<ul style="list-style-type: none"> Inclusão de nova especificação para compras pelo modo Carteira Virtual
1.140	14/11/2019	<ul style="list-style-type: none"> Removida versão do Scope no rodapé do documento. Correção de texto e histórico.
1.139	30/10/2019	<ul style="list-style-type: none"> Inclusão de detalhes de como coletar senha para a rede FIC no item “Redes com tratamentos Específicos”
1.138	22/10/2019	<ul style="list-style-type: none"> Atualização para Credsystem; Novo estado de coleta TC_COLETA CPF_PP Novo campo para ScopeForneceCampo Criação da versão de especificação 2.60 Nova modalidade de pagamento de fatura por CPF com a bandeira Credsystem PL 344.
1.137	03/10/2019	<ul style="list-style-type: none"> Atualização para Ticket Log; Novo estado de coleta TC_COLETA_DADO_ESPECIAL; Nova função ScopeGetParamExt; Nova opção de uso das funções ScopeMenuRecuperaltens e ScopeMenuSelecionaltem Novo campo para ScopeForneceCampo SCOPE_AUTOMACAO_PERMITE_DESCONTO; Nova opção de uso do campo SCOPE_DADOS_LISTA_MERCADORIA; Nova função ScopeObtemProdutosFrota;

		<ul style="list-style-type: none"> Novos campos TicketLog_Desconto_Por_Item e Saldo_Disponivel_Litros_ou_Reais Novo tópico de orientações de implementação para Ticket Log Frota
1.136	02/10/2019	<ul style="list-style-type: none"> Implementação da coleta da data de vencimento na transação ScopeTransacaoPOS. Implementação da coleta do nome do portador do cartão na transação ScopeTransacaoPOS.
1.135	14/08/2019	<ul style="list-style-type: none"> Inclusão da especificação ECXCard V3.02 Inclusão das novas bandeiras ECXCARD CRED, ECXCARD COMB
1.134	14/08/2019	<ul style="list-style-type: none"> Acréscimo da Rede TODO-CARTÕES Acréscimo da Bandeira TODO VOUCHER – débito voucher Criação da execução Especificação 3.02
1.133	29/04/2019	<ul style="list-style-type: none"> Melhoria da seção Redes Padrão Lojista Frota
1.132	25/03/2019	<ul style="list-style-type: none"> Acréscimo da Rede BRADESCARD Acréscimo da Bandeira BRADESCARD CRÉDITO Acréscimo da Especificação 5.16 Acréscimo da função ScopeObtemDadosAdicionais (pág. 225)
1.131	21/03/2019	<ul style="list-style-type: none"> Inclusão da rede COOPERCARD e novas bandeiras COOPERCARD CRED, COOPERCARD ALIM, COOPERCARD REF, COOPERCARD BEN
1.130	06/02/2019	<ul style="list-style-type: none"> Atualização para Recarga de Celular via VEROBANRISUL R17
1.129	12/12/2018	<ul style="list-style-type: none"> Histórico Inicial

Este documento é de propriedade da **NCR BRASIL TECNOLOGIA E SERVIÇOS EM AUTOMAÇÃO LTDA**. Todos os direitos reservados.

As informações aqui contidas têm caráter técnico e informativo e não poderão ser copiadas, fotocopiadas, reproduzidas, traduzidas ou reduzidas a qualquer meio eletrônico ou forma legível por máquina sem a autorização prévia da **NCR BRASIL TECNOLOGIA E SERVIÇOS EM AUTOMAÇÃO LTDA**.