

### Sumario:

Este artigo demonstrar uma introdução de Orientação a Objetos para demonstrar a classe CL\_GUI\_FRONTEND\_SERVICES. Esta classe é responsável por todos os serviços que envolvem o FrontEnd com o sistema operacional, serviços que dizem recursos disponíveis no sistema operacional tais como: Criar, Deletar, Copiar, Verificar a Existência, de Arquivos ou Diretórios, Coletar informações de versões de arquivos dos programas, acessar informações de Redes como IP, HostName, Versão do sistema operacional (Plataforma do SO (WINDOWS, LINUX)), Executar programas externos do SAP, tais como comandos do SO. Será apresentado os principais métodos desta classe, demonstrando o que cada método faz e como utilizá-los.

### Sobre o Autor:

Uderson Luis Fermino, formado em Ciências da Computação pela Faculdade de Pesquisa e Ensino IPEP, atua no mercado a 2 anos como desenvolvedor Java nas plataformas: (J2SE, J2EE e J2ME), com participação em grandes projetos envolvendo estas tecnologias. É consultor ABAP com experiências em REPORT, ALV (GRID, LIST, BLOCK, OO, TREE, HIERARQUICK), IDOC, ALE, ONLINE, SAPSCRIPT, SMARTFORM, NETWEAVER (JCO, BSP, WebDynpro).

## O que é Orientação a Objeto?

Orientações a objeto ou para ser mais precisas, programação orientada a objeto, é um método de resolução de problemas no qual a solução do software reflete objetos do mundo real, todos os elementos de software são baseados em objetos do mundo real.

Uma introdução compreensiva a orientação a objeto como um todo iria muito além dos limites da introdução aos objetos ABAP. Esta documentação introduz uma seleção de termos que são usados universalmente na orientação a objeto e também ocorre em objetos ABAP. Em sessões subseqüentes, continua-se a discutir em mais detalhes como esses termos são usados em objetos ABAP

## Objetos

Um objeto é seção de código fonte que contém dados e fornece serviços. Os dados formam os atributos do objeto. Os serviços são conhecidos como métodos (também conhecido como operações ou funções). Tipicamente, métodos operam em dados privados (os atributos, ou estado do objeto), que é apenas visível para os métodos do objeto. Logo os atributos de um objeto não podem ser modificados diretamente pelo usuário, mas apenas pelos métodos do objeto. Isso garante a consistência interna do objeto, estes dados são do tipo privados do objeto, podemos ter atributos públicos, que são alterados os seus estados sem utilização de métodos e atributos do tipo protegidos somente podem ser alterados pelo objeto.

## Classes

As classes são o corpo do objeto. Uma classe é uma INSTANCIA de um objeto que é carregado em memória para fazer a manipulação dos atributos e métodos pertencente ao objeto que a classe é instanciada

## Referências a Objetos

Em um programa, você identifica e endereça objetos usando referências únicas a objetos. Referência a objetos permite que acesse os atributos e métodos de um objeto.

Em programação orientada a objeto, objetos geralmente têm as seguintes propriedades:

## Encapsulamento

Objetos restringem a visibilidade de seus recursos (atributos e métodos) aos outros usuários. Todo objeto tem uma **interface**, que determina como os outros objetos podem interagir com ele. A implementação do objeto é encapsulada, isso é, invisível for a do próprio objeto.

## Polimorfismo

Métodos idênticos (mesmo nome) se comportam diferentemente em diferentes classes. Orientação orientada a objeto contém construções chamadas interfaces. Elas permitem que enderece métodos com mesmo nome em diferentes objetos. Apesar de a forma de endereçamento é sempre a mesma, a implementação do método é específica a uma particular classe.

### **Herança**

Você pode usar uma classe existente para derivar uma classe nova. Classes derivadas herdam os dados e métodos da superclasse. No entanto, eles podem substituir métodos existentes esta técnica é chamada de sobre carga de método, ou re-escrita de método, e também adicionar novos. Herança pode ser analogicamente comparada com a Herança Familiar.

### **Usos de Orientação a Objeto**

Abaixo estão algumas vantagens da programação orientada a objeto:

- Sistemas de software complexos se tornam mais fáceis de serem compreendidos, já que a estrutura orientada a objeto fornece uma representação muito mais próxima da realidade do que as outras técnicas de programação.
- Em um sistema orientado a objeto bem desenvolvido, é possível implementar mudanças a nível de classe, sem ter que realizar alterações em outros pontos do sistema. Isto reduz a quantidade total de manutenção requerida.
- Através do polimorfismo e herança, a programação orientada a objeto permite que reutilize componentes individuais.
- Em um sistema orientado a objeto, a quantidade de trabalho de manutenção e revisão envolvido é reduzida, já que muitos problemas podem ser detectados e corrigidos em fase de projeto.

Para atingir estes objetivos requer:

- Linguagens de programação orientada a objetos
- Técnicas de programação orientadas a objeto não necessariamente dependem em linguagens de programação orientada a objeto. No entanto, a eficiência da programação orientada a objeto depende diretamente de como as técnicas de programação orientada a objetos são implementadas no sistema kernel.
- Ferramentas de orientação a objeto

Ferramentas de orientação a objeto permitem que se criem programas orientados a objetos em linguagem orientada a objetos. Eles permitem que se modele e guarde objetos e relações entre eles.

- Modelamento orientado a objeto

O modelamento orientado a objeto de um sistema de software é o mais importante, mais demorado, e o requerimento mais difícil para alcançar acima dos objetivos. Design orientado a objeto envolve mais do que apenas programação orientada a objeto, e fornece vantagens lógicas que são independentes da verdadeira implementação.

Em ABAP/4 é possível criar objetos de diversas maneiras, será apresentado duas formas:

- Declaração de referência do OBJETO, mais o comando CREATE OBJECT
- Usando o MODELO do Editor.

Referência : é declarado um objeto especificando o tipo que o objeto terá, está declaração é feita através do comando **TYPE REF TO**.

### **SINTAXE:**

**DATA** nome\_do\_objeto **TYPE REF TO** nome\_da\_classe\_de\_referencia.

Ex.:

```
DATA OBJSERVICE TYPE REF TO CL_GUI_FRONTEND_SERVICES.
```

Declaração de um objeto de nome OBJSERVICE que faz referência para a classe CL\_GUI\_FRONTEND\_SERVICES.

Para criar o objeto, usa-se o comando CREATE OBJECT. Porém não existe criação de objeto em OO, mais instancia de objetos, para melhorar o termos, alocação de dados em memória referente aquele objeto.

### **SINTAXE:**

CREATE OBJECT nome\_do\_objeto\_referenciado.

Exemplo

```
CREATE OBJECT OBJSERVICE
```

## CL\_GUI\_FRONTEND\_SERVICES – UDERSON LUIS FERMINO

---

Quando um objeto é instanciado em ABAP/4, usa-se o comando ->, conhecido de indicador de atributos ou métodos.

Exemplos.:

Usar o método GET\_TEMP\_DIRECTORY, do objeto OBJSERVICE.

```
OBJSERVICE->GET_TEMP_DIRECTORY( CHANGING TEMP_DIR = INITIALFOLDER
                                EXCEPTIONS CNTL_ERROR = 1
                                ERROR_NO_GUI = 2
                                NOT_SUPPORTED_BY_GUI = 3 ).
```

Quando o método, ou atributo da classe usada é STATIC (STATICO), não é necessário utilizar o comando “CREATE OBJECT”, pois os métodos já residem em memória, podendo ser utilizados a qualquer momento.

Para utilizar um método ou atributo static de uma classe usa-se o comando “=>”, será apresentado alguns métodos da classe CL\_GUI\_FRONTEND\_SERVICES, todos os métodos aqui apresentados são static, onde usaremos o comando “=>” para chamar os métodos ou atributos desta classe.

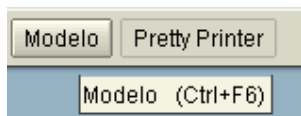
Observação.: são implemente métodos estáticos, quando houver uma extrema necessidade, pois lembrando que quando os métodos ou atributos são estáticos ele ficar alocados em memórias, durante a vida do programa, no caso do da classe CL\_GUI\_FRONTEND\_SERVICES, durante o tempo de vida do SAPGUI-FRONTEND.

A segunda forma apresentada aqui será a instanciar objetos com ajuda do editor MODELO.

1º Declarar o objeto conforme já explicado.

```
DATA OBJSERVICE TYPE REF TO CL_GUI_FRONTEND_SERVICES.
```

2º No Editor ABAP (neste artigo foi usado a versão), clique em “MODELO”.



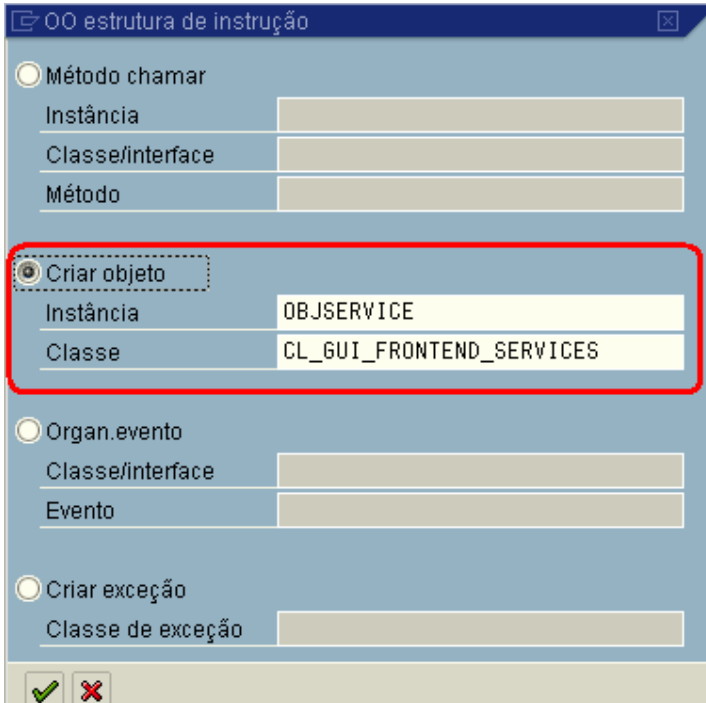
2º Marcar a opção Modelo /para objetos ABAP.

The image shows a dialog box titled "Inserir modelo" with a close button in the top right corner. It contains a list of radio button options for selecting a model type. The option "Modelo p/objetos ABAP" is selected and highlighted with a red dashed box. Below this option, there is a table with columns "ID", "Tp.", "E", and "N°". Other options include "CALL FUNCTION", "MESSAGE", "SELECT \* FROM", "PERFORM", "AUTHORITY-CHECK", "WRITE", "CASE p/Status", "Objeto dados estruturado" (with sub-options "Com campos de estrutura" and "com TYPE->estrutura"), "CALL DIALOG", and "Outro modelo". At the bottom left, there are two buttons: a green checkmark and a red X.

ID	Tp.	E	N°

### 3º Marcar a opção “Criar Objeto”

- INSTÂNCIA = Nome do objeto declarado
- CLASSE = Nome da classe que o objeto faz referencia.



The screenshot shows a dialog box titled "OO estrutura de instrução". It has three main sections, each with a radio button:

- Método chamar:** Includes fields for "Instância", "Classe/interface", and "Método".
- Criar objeto (highlighted with a red box):** Includes fields for "Instância" (containing "OBJSERVICE") and "Classe" (containing "CL\_GUI\_FRONTEND\_SERVICES").
- Organ.evento:** Includes fields for "Classe/interface" and "Evento".
- Criar exceção:** Includes a field for "Classe de exceção".

At the bottom left, there are two small icons: a green checkmark and a red 'X'.

O código que será gerado:

```
CREATE OBJECT OBJSERVICE
* EXCEPTIONS
*   NOT_SUPPORTED_BY_GUI = 1
*   CNTL_ERROR           = 2
*   others                = 3
.
IF SY-SUBRC <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*           WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.
```

4º Chamar método do objeto instanciado.

The image shows a software dialog box titled "OO estrutura de instrução". It has four radio button options: "Método chamar" (selected and highlighted with a red box), "Criar objeto", "Organ. evento", and "Criar exceção". Each option has associated input fields. The "Método chamar" section contains fields for "Instância" (value: OBJSERVICE), "Classe/interface" (value: CL\_GUI\_FRONTEND\_SERVICES), and "Método" (value: GET\_PLATFORM). The "Criar objeto" section has fields for "Instância" (value: OBJSERVICE) and "Classe" (value: CL\_GUI\_FRONTEND\_SERVICES). The "Organ. evento" section has fields for "Classe/interface" and "Evento". The "Criar exceção" section has a field for "Classe de exceção". At the bottom left, there are checkmark and X icons.

- INSTÂNCIA = Nome do objeto instanciado
- CLASSE/INTERFACE = Nome da classe ou interface que o objeto faz referencia.
- MÉTODO = Nome do método a ser chamado.

Observação: Como os métodos desta classe são estáticos conforme já explicado, não é necessário realizar estes passo, somente é preciso informar a classe e o método a ser chamado, conforme a figura:

OO estrutura de instrução

Método chamar:

Instância	
Classe/interface	CL_GUI_FRONTEND_SERVICES
Método	GET_PLATFORM

Criar objeto

Instância	
Classe	

Organ.evento

Classe/interface	
Evento	

Criar exceção

Classe de exceção	
-------------------	--

## CL\_GUI\_FRONTEND\_SERVICES – UDERSON LUIS FERMINO

---

A classe CL\_GUI\_FRONTEND\_SERVICES,

Os métodos que serão descritos abaixo, podem ser utilizados sem instanciar a classe, pois estes métodos são do tipo STATIC, onde todos os tipos de dados declarados como static quando o objeto é declarado podem ser utilizados a qualquer momento, pois estes ficam na memória esperando serem utilizados.

Segue a lista dos métodos e suas logo abaixo da lista será descrito como utilizar cada método.

- CHECK\_HANDLE
- CLASS\_CONSTRUCTOR
- CLIPBOARD\_EXPORT
- CLIPBOARD\_IMPORT
- DIRECTORY\_BROWSE
- DIRECTORY\_CREATE
- DIRECTORY\_DELETE
- DIRECTORY\_EXIST
- DIRECTORY\_GET\_CURRENT
- DIRECTORY\_LIST\_FILES
- DIRECTORY\_SET\_CURRENT
- ENVIRONMENT\_GET\_VARIABLE
- ENVIRONMENT\_SET\_VARIABLE
- EXECUTE
- FILE\_COPY
- FILE\_DELETE
- FILE\_EXIST
- FILE\_GET\_SIZE
- FILE\_GET\_VERSION
- FILE\_OPEN\_DIALOG
- FILE\_SAVE\_DIALOG
- FILE\_GET\_ATTRIBUTES
- FILE\_SET\_ATTRIBUTES
- GET\_COMPUTER\_NAME
- GET\_DESKTOP\_DIRECTORY
- GET\_DRIVE\_TYPE
- GET\_FREE\_SPACE\_FOR\_DRIVE
- GET\_IP\_ADDRESS
- GET\_PLATFORM
- GET\_SAPGUI\_WORKDIR
- GET\_SAPGUI\_DIRECTORY
- GET\_SYSTEM\_DIRECTORY
- GET\_TEMP\_DIRECTORY
- GET\_USER\_NAME
- GET\_WINDOWS\_DIRECTORY
- GUI\_DOWNLOAD
- GUI\_UPLOAD
- IS\_TERMINAL\_SERVER

- REGISTRY\_DELETE\_KEY
- REGISTRY\_DELETE\_VALUE
- REGISTRY\_GET\_VALUE
- REGISTRY\_SET\_VALUE
- REGISTRY\_GET\_DWORD\_VALUE
- REGISTRY\_SET\_DWORD\_VALUE
- CONSTRUCTOR
- GET\_GUI\_PROPERTIES
- CHECK\_GUI\_SUPPORT
- ADD\_SUPPORT\_BIT
- REMOVE\_SUPPORT\_BIT
- GET\_GUI\_VERSION
- GET\_UPLOAD\_DOWNLOAD\_PATH

### FILE\_GET\_VERSION

Método que retorna a versão de um determinado arquivo deve-se passar o endereço e o nome do arquivo, e no argumento CHANGING uma variável do tipo STRING que será conterá o valor da versão do arquivo especificado.

DATA VERSION TYPE STRING.

CLEAR VERSION .

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>FILE\_GET\_VERSION

EXPORTING

FILENAME = 'C:\Adobe\Reader.0\Reader\AdobeUpdateCheck.exe'

CHANGING

VERSION = VERSION

EXCEPTIONS

CNTL\_ERROR = 1

ERROR\_NO\_GUI = 2

NOT\_SUPPORTED\_BY\_GUI = 3

WRONG\_PARAMETER = 4

OTHERS = 5.

WRITE: / VERSION.

FILENAME = Nome do arquivo a ser consultado

VERSION = Variável do tipo string que recebe a versão do arquivo consultado

### GET\_IP\_ADDRESS

Método para pegar o IP (Internet Protocol) da Máquina que está rodando o FRONTEND, este método retorna um IP no formato STRING, devendo ser passado uma variável do tipo STRING para o método, que conterá o valor de retorno do método, contendo o endereço IP.

```
DATA: IP TYPE STRING.
CALL METHOD CL_GUI_FRONTEND_SERVICES=>GET_IP_ADDRESS
RECEIVING
  IP_ADDRESS                = IP
EXCEPTIONS
  CNTL_ERROR                = 1
  ERROR_NO_GUI              = 2
  NOT_SUPPORTED_BY_GUI      = 3
  OTHERS                    = 4.
```

WRITE: / IP.

IP\_ADDRESS = retorna uma variável do tipo String contendo IP da máquina que o FrontEnd Roda.

Este método traz somente o ip da máquina que está rodando o FRONTEND, pode acontecer que precisamos saber o IP de uma determinada máquina da rede, por exemplo o servidor, como temos a variável de sistema SY-HOST que armazena em memória o nome HOST do servidor SAP, queremos descobrir o IP do servidor, para este tipo de problema podemos utilizar a função **RFC\_HOST\_TO\_IP**, que recebe um nome de uma determinada máquina na rede (HOST) e retorna o IP da mesma. Os argumentos devem ser de entrada e saída devem ser variáveis do tipo de C do tamanho Máximo de 100 caracteres. Usando esta função podemos descobrir qualquer IP da rede.

```
DATA IP(100) TYPE C.

CALL FUNCTION 'RFC_HOST_TO_IP'
EXPORTING
  RFCHOST                = 'treinamento03'
IMPORTING
  RFCIP                  = IP
EXCEPTIONS
  HOST_TO_IP_CONVERSION_ERROR = 1
  OTHERS                  = 2.
```

WRITE: IP.

RFCHOST = Recebe como entrada o HOSTNAME do tipo C de 100.  
RFCIP = Retorna o IP relativo ao HOSTNAME, o retorno é do tipo C de 100.

### GET\_COMPUTER\_NAME

Método para capturar o nome (HOST) da maquina, deve-se passa como argumento de alteração uma variável do tipo STRING que conterà o nome da maquina que está rodando o FRONTEND.

DATA HOST TYPE STRING.

CALL METHOD

CL\_GUI\_FRONTEND\_SERVICES=>GET\_COMPUTER\_NAME

CHANGING

COMPUTER\_NAME = HOST

EXCEPTIONS

CNTL\_ERROR = 1

ERROR\_NO\_GUI = 2

NOT\_SUPPORTED\_BY\_GUI = 3

Others = 4.

WRITE: / HOST.

COMPUTER\_NAME = Retorna o HOSTNAME da maquina que está rodando o FRONTEND, sendo uma variável do tipo STRING.

Idêntico ao método **GET\_IP\_ADDRESS**, este método somente retorna o endereço host da maquina que está rodando o FRONTEND, caso seja necessário captura o endereço nome (HOST) de uma maquina da rede este método não será capaz, para consegui esta opção é possível utilizar a função RFC\_IP\_TO\_HOST, que recebe como argumento um endereço de internet (IP/ INTERNET PROTOCOL) e retorna uma variável do tipo C de tamanho Maximo de 100 caracteres, contendo o nome da maquina (HOST) relativo a um determinado IP.

DATA HOST (100) TYPE C.

CALL FUNCTION 'RFC\_IP\_TO\_HOST'

EXPORTING

RFCIP = '192.168.0.3'

IMPORTING

RFCHOST = HOST

EXCEPTIONS

IP\_TO\_HOST\_CONVERSION\_ERROR = 1

OTHERS = 2.

WRITE: HOST.

RFCIP = Recebe uma variável do tipo c de 100, contendo IP.

RFCHOST = Retorna uma variável do tipo c de 100, contendo o HOSTNAME relativo ao IP de entrada.

### FILE\_DELETE

Método para apagar um determinado arquivo, deve-se passar o endereço (caminho) e o nome do arquivo a ser deletado (apagado/excluído), é uma variável do tipo I que no argumento RC que irá retornar um numero informando a deleção.

DATA: RC TYPE I.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>FILE\_DELETE

EXPORTING

FILENAME = 'C:\test.txt'

CHANGING

RC = RC

EXCEPTIONS

FILE\_DELETE\_FAILED = 1

CNTL\_ERROR = 2

ERROR\_NO\_GUI = 3

FILE\_NOT\_FOUND = 4

ACCESS\_DENIED = 5

UNKNOWN\_ERROR = 6

NOT\_SUPPORTED\_BY\_GUI = 7

WRONG\_PARAMETER = 8

Others = 9.

FILENAME = Recebe uma Variável do tipo string contendo o nome do arquivo, concatenado do caminho.

RC = Variável do tipo I, que indicará com 1 quando a ação do método foi realizada com sucesso e 0 insucesso.

### FILE\_GET\_SIZE

Método para capturar o tamanho de um determinado arquivo, deve-se passar endereço e o nome do arquivo e uma variável do tipo I que recebera a informação do tamanho do arquivo especificado.

DATA : SIZE TYPE I .

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>FILE\_GET\_SIZE

EXPORTING

FILE\_NAME = 'C:\ping.txt'

IMPORTING

FILE\_SIZE = SIZE

EXCEPTIONS

FILE\_GET\_SIZE\_FAILED = 1

CNTL\_ERROR = 2

ERROR\_NO\_GUI = 3

NOT\_SUPPORTED\_BY\_GUI = 4

Others = 5.

WRITE: / SIZE.

FILENAME = Recebe uma Variável do tipo string contendo o nome do arquivo, concatenado do caminho.

FILE\_SIZE = Recebe uma variável do tipo I que será carregada pelo método como tamanho do arquivo especificado.

### FILE\_EXIST

Método para verificar se um determinado arquivo existe ou não este método recebe como parâmetro um endereço e nome de um determinado arquivo, e retorna um char de 1 contendo o valor 'X' caso existe e um SPACE caso não exista o arquivo.

DATA: V\_RESULT(1) TYPE C.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>FILE\_EXIST

EXPORTING

FILE = 'C:\audio\Speclab\INSTALL.LOG'

RECEIVING

RESULT = V\_RESULT

EXCEPTIONS

CNTL\_ERROR = 1

ERROR\_NO\_GUI = 2

WRONG\_PARAMETER = 3

NOT\_SUPPORTED\_BY\_GUI = 4

Others = 5.

WRITE V\_RESULT.

FILE = Recebe uma Variável do tipo string contendo o nome do arquivo, concatenado do caminho.

RESULT = Retorna uma 1 caso existe e 0 quando o arquivo não existir, está variável é do tipo I.

### FILE\_COPY

Método, responsável, por copiar um arquivo de um determinado local para outro local, por exemplo, copiar um arquivo que está no diretório C:\temp para o diretório C:\temp2. Este método recebe como argumento um endereço de origem e o nome do arquivo a ser copiado e um endereço de destino e o nome que o arquivo terá a quando ser copiado.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>FILE\_COPY

EXPORTING

SOURCE = 'C:\audio\Speclab\INSTALL.LOG'

DESTINATION = 'C:\INSTALL.LOG '

OVERWRITE = 'X'

EXCEPTIONS

CNTL\_ERROR = 1

ERROR\_NO\_GUI = 2

WRONG\_PARAMETER = 3

DISK\_FULL = 4

ACCESS\_DENIED = 5

FILE\_NOT\_FOUND = 6

DESTINATION\_EXISTS = 7

UNKNOWN\_ERROR = 8

PATH\_NOT\_FOUND = 9

DISK\_WRITE\_PROTECT = 10

DRIVE\_NOT\_READY = 11

NOT\_SUPPORTED\_BY\_GUI = 12

Others = 13.

SOURCE = Recebe o caminho e o nome do arquivo a ser copiado, deve ser uma variável do tipo string.

DESTINATION = Recebe o caminho e o nome que o arquivo terá quando ser copiado para este destino, o nome do arquivo não necessita ser o mesmo do SOURCE (ORIGEM).

### DIRECTORY\_CREATE

Método, responsável pela criação de um determinado diretório no Sistema Operacional, este método independe do sistema operacional que o FRONTEND está rodando, ele se ajusta a criar o diretório de acordo cada Sistema operacional (Lembrando que os comandos de criação de diretórios mudam de SO para SO), este método recebe como argumento o endereço juntamente do nome do diretório a ser criado e retorna um argumento do tipo I que retorna um numero 0, para criação e -1 para erro de criação.

Ex.:

DATA RC TYPE I.

```
CALL METHOD CL_GUI_FRONTEND_SERVICES=>DIRECTORY_CREATE
EXPORTING
  DIRECTORY                = 'C:\teste'
CHANGING
  RC                        = RC
EXCEPTIONS
  DIRECTORY_CREATE_FAILED  = 1
  CNTL_ERROR                = 2
  ERROR_NO_GUI              = 3
  DIRECTORY_ACCESS_DENIED  = 4
  DIRECTORY_ALREADY_EXISTS = 5
  PATH_NOT_FOUND           = 6
  UNKNOWN_ERROR            = 7
  NOT_SUPPORTED_BY_GUI     = 8
  WRONG_PARAMETER         = 9
  Others                    = 10.
```

WRITE RC.

**DIRECTORY** = Recebe o caminho do diretório a ser criado, lembrando que um diretório somente é um diretório se ele for criado diretamente na RAIZ, por exemplo, a RAIZ do SO Windows é uma letra ALFABETICA, geralmente “C:” para a primeira partição do disco rígido e assim prossegue para as demais partições ou drives, com exceção ao drive de disquete que se inicia na RAIZ “A:”. Para SO “LINUX, UNIX”, e outro que seguem o padrão UNIX a RAIZ é iniciada pela letra “/” independente da partição, somente mudando para os drives externos. Com estes este conceito em mente, quando é criando um diretório que não esteja em uma RAIZ mais sim dentro de um diretório, este é chamado de subdiretório. O tipo de dados de entrada é uma variável do tipo string

**RC** = Retorna uma variável do tipo I contendo informação sobre a criação do Diretório ou Subdiretório.

### DIRECTORY\_EXIST

Método, responsável pela verificação de existência de um determinado diretório ou subdiretório, recebendo como argumento o endereço do diretório a ser pesquisado e retornado uma variável do tipo C, de tamanho 1 contendo o valor "X", para a existência do diretório e SPACE para a não existência.

DATA RESULT TYPE C.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>DIRECTORY\_EXIST

EXPORTING

DIRECTORY = 'C:\teste'

RECEIVING

RESULT = RESULT

EXCEPTIONS

CNTL\_ERROR = 1

ERROR\_NO\_GUI = 2

WRONG\_PARAMETER = 3

NOT\_SUPPORTED\_BY\_GUI = 4

Others = 5.

WRITE RESULT.

DIRECTORY = Recebe o caminho e o nome do diretório a ser verificado, sendo uma variável do tipo string.

RESULT = Retorna uma variável do tipo c de 1 contendo a informação da verificação.

### DIRECTORY\_DELETE

Método responsável pela deleção/exclusão de um determinado diretório, este método recebe como argumento um endereço de um determinado diretório e retorna uma variável do tipo c de tamanho 1, contendo 'X' para deleção efetuada com sucesso e SPACE como deleção não efetuada.

DATA: RC TYPE C.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>DIRECTORY\_DELETE

EXPORTING

DIRECTORY = 'C:\teste'

CHANGING

RC = RC

EXCEPTIONS

DIRECTORY\_DELETE\_FAILED = 1

CNTL\_ERROR = 2

ERROR\_NO\_GUI = 3

PATH\_NOT\_FOUND = 4

DIRECTORY\_ACCESS\_DENIED = 5

UNKNOWN\_ERROR = 6

NOT\_SUPPORTED\_BY\_GUI = 7

WRONG\_PARAMETER = 8

Others = 9.

WRITE RC.

DIRECTORY = Recebe o caminho e o nome do diretório a ser excluído, sendo uma variável do tipo string.

RC = Retorna uma variável do tipo I contendo informação sobre a exclusão do Diretório ou Subdiretório.

## CL\_GUI\_FRONTEND\_SERVICES – UDERSON LUIS FERMINO

---

Método responsável pela verificação do Sistema operacional que está rodando o FRONTEND, este método retorna uma variável do tipo I, contendo um determinado numero que o sistema SAP, tem embutido contendo uma tabela para este numero, onde cada numero esta associado a uma plataforma, para descobrir o numero relativo a plataforma a classe CL\_GUI\_FRONTEND\_SERVICES, contem atributos do tipo constantes (valores pré-definido) contendo o tipo de plataforma, caso queiramos retornar o nome d plataforma basta verificar qual valor contido nestas constante relativo ao valor retornado pelo método.

As constantes são:

HKEY\_USERS  
PLATFORM\_UNKNOWN  
PLATFORM\_WINDOWS95  
PLATFORM\_WINDOWS98  
PLATFORM\_NT351  
PLATFORM\_NT40  
PLATFORM\_NT50  
PLATFORM\_MAC  
PLATFORM\_OS2  
PLATFORM\_LINUX  
PLATFORM\_HPUX  
PLATFORM\_TRU64  
PLATFORM\_AIX  
PLATFORM\_SOLARIS

DATA: PLATAFORMA TYPE I,  
NOME\_PLATF TYPE STRING.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>GET\_PLATFORM  
RECEIVING  
PLATFORM = PLATAFORMA  
EXCEPTIONS  
ERROR\_NO\_GUI = 1  
CNTL\_ERROR = 2  
NOT\_SUPPORTED\_BY\_GUI = 3  
Others = 4.

PLATFORM = Retorna uma variável do tipo I, contendo um numero relativo ao SO.

## CL\_GUI\_FRONTEND\_SERVICES – UDERSON LUIS FERMINO

---

Para verificar qual o numero relativo que o SAP armazena basta usar qualquer tipo de condição múltipla para verificar qual constante que o SAP armazena o numero relativo do SO.

Ex.:

CASE PLATAFORMA.

```
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_WINDOWS95.  
  NOME_PLATF = 'WN32_95'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_WINDOWS98.  
  NOME_PLATF = 'WN32_98'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_NT351.  
  NOME_PLATF = 'WN32'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_NT40.  
  NOME_PLATF = 'WN32'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_NT50.  
  NOME_PLATF = 'WN32'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_MAC.  
  NOME_PLATF = 'MC'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_OS2.  
  NOME_PLATF = 'PM'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_LINUX.  
  NOME_PLATF = 'MF'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_HPUX.  
  NOME_PLATF = 'MF'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_TRU64.  
  NOME_PLATF = 'MF'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_AIX.  
  NOME_PLATF = 'MF'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_SOLARIS.  
  NOME_PLATF = 'MF'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_MACOSX.  
  NOME_PLATF = 'MF'.  
WHEN 14. "PLATFORM_WINDOWSXP  
  NOME_PLATF = 'WN32'.  
WHEN CL_GUI_FRONTEND_SERVICES=>PLATFORM_UNKNOWN.  
  NOME_PLATF = '??'.  
WHEN OTHERS.  
  NOME_PLATF = '??'.  
ENDCASE.
```

WRITE NOME\_PLATF .

CL\_GUI\_FRONTEND\_SERVICES=>PLATFORM\_XXXX = Onde XXX e  
equivale ao valor numérico de cada constante.

### GET\_USER\_NAME

Método responsável, pelo retorno de usuário logado no sistema, recebe como argumento uma variável do tipo STRING.

DATA NOME TYPE STRING.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>GET\_USER\_NAME

CHANGING

USER\_NAME = NOME

EXCEPTIONS

CNTL\_ERROR = 1

ERROR\_NO\_GUI = 2

NOT\_SUPPORTED\_BY\_GUI = 3

Others = 4.

WRITE NOME.

USER\_NAME = Retorna uma variável do tipo string contendo o nome do usuário logado no sistema.

Obs.: A variável sy-uname, armazena está mesma informação, sendo mais rápido utilizar está variável do que o método.

### EXECUTE

Método, responsável por executar funções externar do sistema SAP, este método é um pouco complexo e será descrito por partes.

Exemplos de alguns aplicativos que é possível executarem com este método.

NOTEPAD.EXE	=	Bloco de notas
CMD.EXE	=	Prompt de comando
WINWORD.EXE	=	Microsoft Word
CALC.EXE	=	Calculadora
FIREFOX.EXE	=	Browser de Internet

```
CALL METHOD CL_GUI_FRONTEND_SERVICES=>EXECUTE
EXPORTING
  DOCUMENT                = ''
  APPLICATION              = 'CMD.EXE'
  PARAMETER                = 'ping 192.168.0.4'
*  DEFAULT_DIRECTORY      =
*  MAXIMIZED               = 'X' "Inicializa Maximizado
*  MINIMIZED               = 'X' "Inicializa Minimizado
*  SYNCHRONOUS             =
*  OPERATION               = 'OPEN'
EXCEPTIONS
  CNTL_ERROR              = 1
  ERROR_NO_GUI            = 2
  BAD_PARAMETER           = 3
  FILE_NOT_FOUND          = 4
  PATH_NOT_FOUND          = 5
  FILE_EXTENSION_UNKNOWN = 6
  ERROR_EXECUTE_FAILED    = 7
  SYNCHRONOUS_FAILED      = 8
  NOT_SUPPORTED_BY_GUI    = 9
  OTHERS                  = 10.
```

Quando for executado uma determinada aplicação que exige parâmetros de entrada, como no caso um prompt de comando, usa-se o parâmetro APPLICATION passando para este parâmetro o nome do aplicativo a ser chamado, e para o parâmetro PARAMETER passa os(s) parâmetros necessários para executar o aplicativo.

Observe que no exemplo é chamado o prompt de comando DOS, passando o comando ping, quando o aplicativo prompt for aberto ele executará o comando ping.

Caso seja necessário apenas abri o aplicativo pode se usar os parâmetros: DOCUMENT ou APPLICATION.

Exemplo: abri um documento do WORD ou um bloco de Nota.

## CL\_GUI\_FRONTEND\_SERVICES – UDERSON LUIS FERMINO

---

Para abri um endereço da internete basta passar a URL( Uniform Resorce Location ) para o parâmetro DOCUMENT. Exe.: [www.fliino.com.br](http://www.fliino.com.br)

### FILE\_OPEN\_DIALOG

Método responsável pela procura de arquivo, este método apenas faz a procura de arquivo especificado, para coletar um determinado arquivo é necessário usar o método GUI\_UPLOAD. A grande vantagem de usar este método é que ele permite selecionar mais de um arquivo e fazer com que o usuário especifique o caminho correto do arquivo. Este método apenas serve para coletar o caminho correto de um determinado arquivo, após o caminho estar coletado, é possível fazer diversas operações como, por exemplo, apagar copiar salvar em memória, etc....

DATA: IT\_TAB TYPE FILETABLE,  
RC TYPE I.

REFRESH: IT\_TAB.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>FILE\_OPEN\_DIALOG  
EXPORTING

WINDOW_TITLE	= 'Select File'
DEFAULT_EXTENSION	= '.doc'
DEFAULT_FILENAME	= 'ABAP.doc'
FILE_FILTER	= '.doc'
MULTISELECTION	= 'X'
CHANGING	
FILE_TABLE	= IT_TAB
RC	= RC.

WINDOW\_TITLE = Titulo da Janela.



DEFAULT\_EXTENSION = Extensão que deverá iniciar como padrão, apenas iniciar como padrão, mais pode ser alterada a extensão.



DEFAULT\_FILENAME = Nome do arquivo a ser procurado, que iniciará como padrão, podendo ser alterado.



## CL\_GUI\_FRONTEND\_SERVICES – UDERSON LUIS FERMINO

---

FILE\_FILTER = Extensão que deverá ser coletada, de arquivos, neste caso somente arquivos com a extensão especificada, poderá ser coletado.

MULTISELECTION = Permite a seleção de múltiplos arquivos, onde X equivale à múltipla escolha, onde se inicia está opção com SPACE.

Todos os tipos de dados de entrada são do tipo string.

FILE\_TABLE = Retorna uma estrutura do tipo FILE\_TABLE, contendo o(s) nome(s) do(s) arquivo(s) selecionado(s).

Exemplo:

Criaremos um Selection-Screen, com bloco e frame, que conterà um select-options, que armazena o(s) endereço(s) do diretório(s) ou arquivo(s).

Este select-options, quando for acionado pelo evento “AT SELECTION-SCREEN ON VALUE-REQUEST FOR” irá chamar o método, e coletar múltiplos endereços de arquivos.

### **SELECTION-SCREEN BEGIN OF BLOCK M WITH FRAME.**

SELECT-OPTIONS SO\_FPATH FOR RLGRAP-FILENAME.

### **SELECTION-SCREEN END OF BLOCK M.**

### **AT SELECTION-SCREEN ON VALUE-REQUEST FOR SO\_FPATH-LOW.**

REFRESH: IT\_TAB.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>FILE\_OPEN\_DIALOG  
EXPORTING

WINDOW\_TITLE = 'Select File'

DEFAULT\_EXTENSION = '.doc'

DEFAULT\_FILENAME = 'ABAP.doc'

FILE\_FILTER = '.doc'

MULTISELECTION = 'X'

CHANGING

FILE\_TABLE = IT\_TAB

RC = RC.

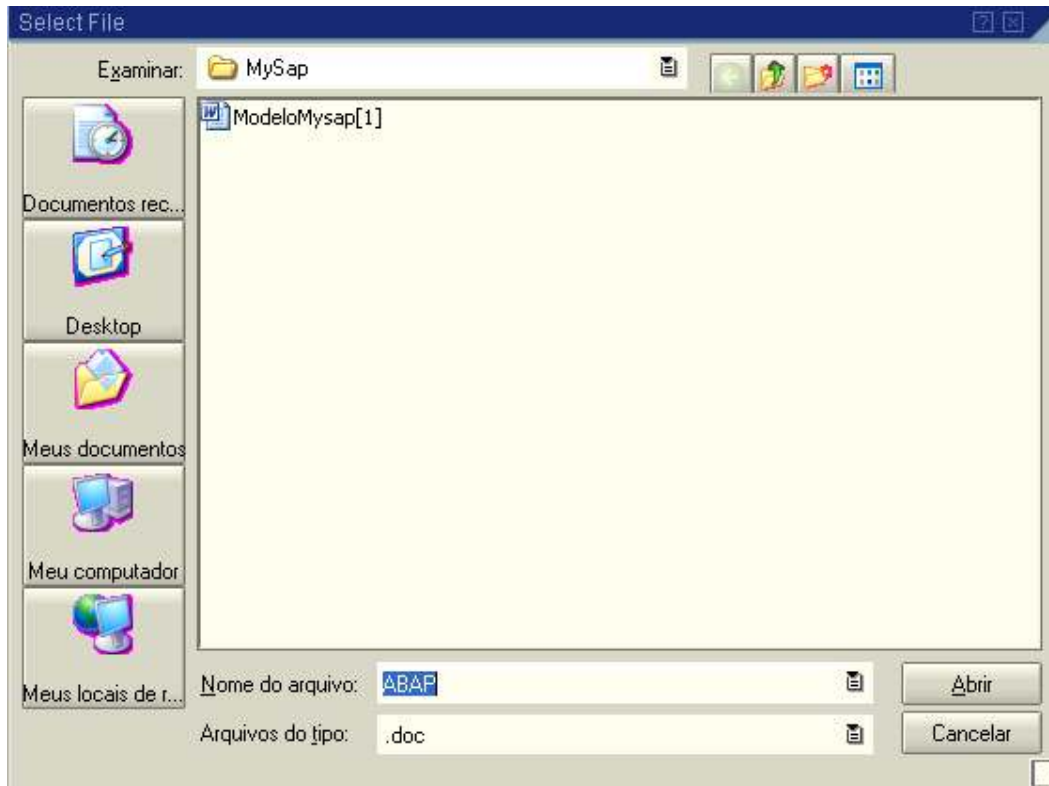
LOOP AT IT\_TAB INTO SO\_FPATH-LOW.

SO\_FPATH-SIGN = 'I'.

SO\_FPATH-OPTION = 'EQ'.

APPEND SO\_FPATH.

ENDLOOP.



## DIRECTORY\_BROWSE

Método responsável pela coleta de um endereço de um determinado diretório equivalente ao método **FILE\_OPEN\_DIALOG**. Com este método pode-se pesquisar em modo gráfico a seleção de determinados diretórios. São passados três parâmetros para este método os três parâmetros que são passados são do tipo STRING uma é o título da janela (WINDOW\_TITLE) o outro é o endereço da pasta inicial que deverá ser feito a busca (INITIAL\_FOLDER) e o outro é o endereço da pasta pesquisada (PICKEDFOLDER) este último parâmetro é um retorno do método CHANGING ().

DATA: PICKEDFOLDER TYPE STRING.

DATA: INITIALFOLDER TYPE STRING.

CALL METHOD CL\_GUI\_FRONTEND\_SERVICES=>DIRECTORY\_BROWSE

EXPORTING

WINDOW\_TITLE = 'Selecione um arquivo'

INITIAL\_FOLDER = INITIALFOLDER

CHANGING

SELECTED\_FOLDER = PICKEDFOLDER

EXCEPTIONS

CNTL\_ERROR = 1

ERROR\_NO\_GUI = 2

NOT\_SUPPORTED\_BY\_GUI = 3

Others = 4.

P\_FILE = PICKEDFOLDER.

WINDOW\_TITLE = Título da Janela.

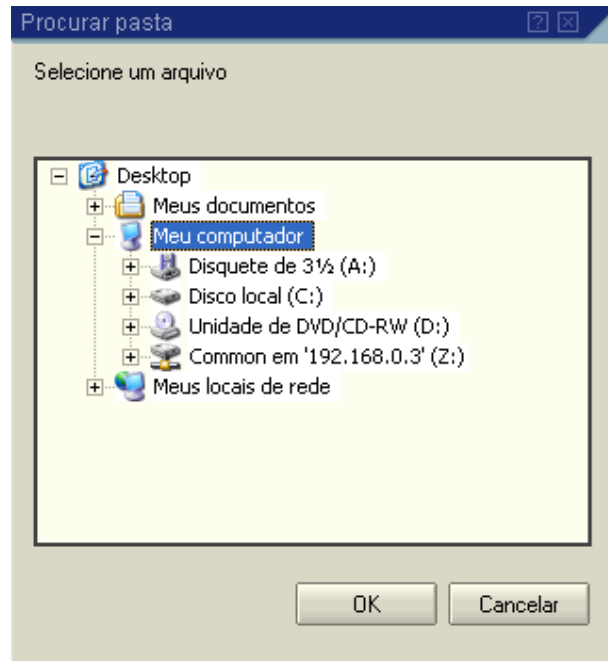


INITIAL\_FOLDER = Pasta Inicial, diretório que deverá iniciar para a pesquisa, lembre-se que iniciar diretamente no diretório, específico é uma boa engenharia de software, pois facilita o tempo do usuário.



SELECTED\_FOLDER = Variável contendo o caminho do arquivo pesquisado.

Todos os dados de entradas e saídas são do tipo string.



## GUI\_UPLOAD

Método, responsável pela captura de dados de um determinado arquivo, que está armazenado em algum local do SO. Este método é equivalente a função GUI\_UPLOAD, pois dentro do método é acionado é a própria função.

```
CALL METHOD CL_GUI_FRONTEND_SERVICES=>GUI_UPLOAD
```

```
EXPORTING
  FILENAME           = lf_file
  FILETYPE           = 'ASC'
  HAS_FIELD_SEPARATOR = 'X'
  HEADER_LENGTH      = 0
  DAT_MODE           = SPACE
  CODEPAGE           = SPACE
  IGNORE_CERR        = ABAP_TRUE
  REPLACEMENT        = 'X'
  READ_BY_LINE       = 'X'
IMPORTING
  FILELENGTH         = lf_filelength
  HEADER              =
CHANGING
  DATA_TAB          = lt_datatabc
EXCEPTIONS
  FILE_OPEN_ERROR    = 1
  FILE_READ_ERROR    = 2
  NO_BATCH            = 3
  GUI_REFUSE_FILETRANSFER = 4
  INVALID_TYPE       = 5
  NO_AUTHORITY        = 6
  UNKNOWN_ERROR       = 7
  BAD_DATA_FORMAT    = 8
  HEADER_NOT_ALLOWED = 9
  SEPARATOR_NOT_ALLOWED = 10
  HEADER_TOO_LONG    = 11
  UNKNOWN_DP_ERROR   = 12
  ACCESS_DENIED       = 13
  DP_OUT_OF_MEMORY   = 14
  DISK_FULL           = 15
  DP_TIMEOUT          = 16
  NOT_SUPPORTED_BY_GUI = 17
  ERROR_NO_GUI        = 18
  OTHERS              = 19.
```

FILENAME = Caminho e nome do arquivo a ser coletado, este parâmetro pode ser preenchido, antes deste método, utilizando os métodos:

- **DIRECTORY\_BROWSE**
- **FILE\_OPEN\_DIALOG**

FILETYPE = Tipo do arquivo a ser coletado.

- ASC = Arquivo do tipo tabela ascii (character).
- DAT = Arquivo do tipo genetico (word(doc), excel(xls), etc. )
- BIN = Arquivo do tipo binario, os dados viram em formato binario, geralmente usados para trabalhar no formato X (Hexadecimal).

HAS\_FIELD\_SEPARATOR = Coloca um delimitador co final de cada linha do arquivo, esté delimitador é um arquivo (#).

- X retira o delimitador
- SPACE insere o delimitador

Por default o metodo insere o SPACE, que firá com o delimitador.

Todos os tipos de entradas de dados são do tipo STRING.

FILELENGTH = Retorna o Tamanho do arquivo, o tipo da variavel que recebe o tamanho do arquivo é do tipo I.

DATA\_TAB = Retorana os dados do arquivo no formato de tabela, existe N formas de criar a tabela que rceberá os dados, aqui neste artigos será apresentado apenas uma form, ficando a disposição da pesquisa, por conta do leitor.

1º Declara-se um tipo:

```
TYPES: TY_CLINE(1024) TYPE C.
```

2º Declara-se uma estrutura:

```
DATA: ST_DATATABC TYPE STANDARD TABLE OF TY_CLINE.
```

Quando o método retorna a tabela, basta utilizar qualquer comando de iteração do ABAP/4 para capturar os dados. Um dos jeitos de captura usando iteração é utilizando um LOOP AT INTO.

Exe.:

Declara

```
DATA ST_TABLINE TYPE TY_CLINE.
```

```
LOOP AT st_datatabc INTO st_tabline .
```

```
WRITE: / lf_tabline.
```

```
ENDLOOP.
```

## GUI\_DOWNLOAD

Métodos responsáveis em criar um determinado arquivo com dados para um diretório, inverso do método gui\_upload, que carrega um arquivo, este método, recebe uma tabela contendo dados e cria um arquivo, no caminho especificado pelo programador, o arquivo que este método pode criar pode ser de qualquer tipo, igualmente ao método gui\_upload, onde BIN representa qualquer tipo de dados, possível (XML, ASC, DOC, XLS).

```
CALL METHOD CL_GUI_FRONTEND_SERVICES=>GUI_DOWNLOAD
EXPORTING
  BIN_FILESIZE           = LF_FILELENGTH
  FILENAME               = 'C:\testeDownload.txt'
  FILETYPE              = 'ASC'
  APPEND                 = SPACE
  WRITE_FIELD_SEPARATOR = SPACE
  HEADER                = '00'
  TRUNC_TRAILING_BLANKS = SPACE
  WRITE_LF              = 'X'
  COL_SELECT            = SPACE
  COL_SELECT_MASK       = SPACE
  DAT_MODE              = SPACE
  CONFIRM_OVERWRITE     = SPACE
  NO_AUTH_CHECK         = SPACE
  CODEPAGE              = SPACE
  IGNORE_CERR           = ABAP_TRUE
  REPLACEMENT           = '#'
  WRITE_BOM             = SPACE
  TRUNC_TRAILING_BLANKS_EOL = 'X'
IMPORTING
  FILELENGTH             =
  CHANGING
    DATA_TAB            = DATATAB[]
EXCEPTIONS
  FILE_WRITE_ERROR      = 1
  NO_BATCH               = 2
  GUI_REFUSE_FILETRANSFER = 3
  INVALID_TYPE          = 4
  NO_AUTHORITY          = 5
  UNKNOWN_ERROR         = 6
  HEADER_NOT_ALLOWED    = 7
  SEPARATOR_NOT_ALLOWED = 8
  FILESIZE_NOT_ALLOWED  = 9
  HEADER_TOO_LONG       = 10
  DP_ERROR_CREATE        = 11
  DP_ERROR_SEND          = 12
  DP_ERROR_WRITE         = 13
  UNKNOWN_DP_ERROR      = 14
  ACCESS_DENIED          = 15
  DP_OUT_OF_MEMORY      = 16
  DISK_FULL              = 17
  DP_TIMEOUT             = 18
  FILE_NOT_FOUND         = 19
  DATAPROVIDER_EXCEPTION = 20
  CONTROL_FLUSH_ERROR    = 21
  NOT_SUPPORTED_BY_GUI   = 22
  ERROR_NO_GUI           = 23
  others                 = 24.
```

## CL\_GUI\_FRONTEND\_SERVICES – UDERSON LUIS FERMINO

---

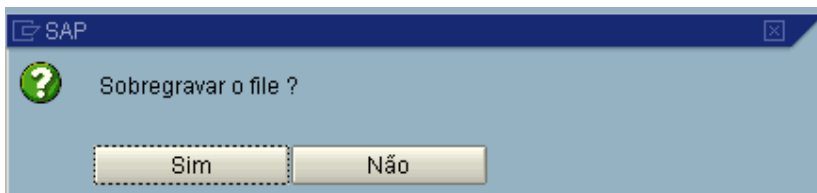
`BIN_FILE` =

`FILENAME` = Caminho do arquivo a ser salvo, mais o nome que o arquivo terá.

`FILETYPE` = Tipo do arquivo a ser salvo (BIN – (para todos os tipo), DAT, ASC).

`CONFIR_OVERWRITER` = Este paramento define uma confirmação quando o arquivo que será criado já existir, ela informará o usuario se deseja realmente sobre-escrever o arquivo.

- X = Define a confirmação
- SPACE = Não pede confirmação, sobre-escreve o arquivo sempre, por default, está é a opção que a função carrega.



Como vsta na lista logo no inicio deste tutorial, a classe `CL_GUI_FRONTEND_SERVICES`, possui difersos métodos e atributos, porem aqui neste artigo somente foi descritos os mais usuais, e que provavelmente usaremos nos dia-dia de programação ABAP/4.