Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

Framework Corporatiu J2EE

Servei de FTP

Versió 1.3

Barcelona, 16 / març / 2007

Històric de modificacions

Data	Autor	Comentaris	Versió
13/01/2006	Atos Origin, sae openTrends	Versió inicial del document	1.0
10/02/2006	Atos Origin, sae openTrends	Canvi d'implementació, J-FTP per Jakarta Commons Net	1.0
13/02/2006	Atos Origin, SAE	Versió 1.2 d'OpenFrame	1.2
16/03/2006	Atos Origin, SAE	Versió 1.3 d'OpenFrame	1.3

Llegenda de Marcadors

Índex

1. Introducció

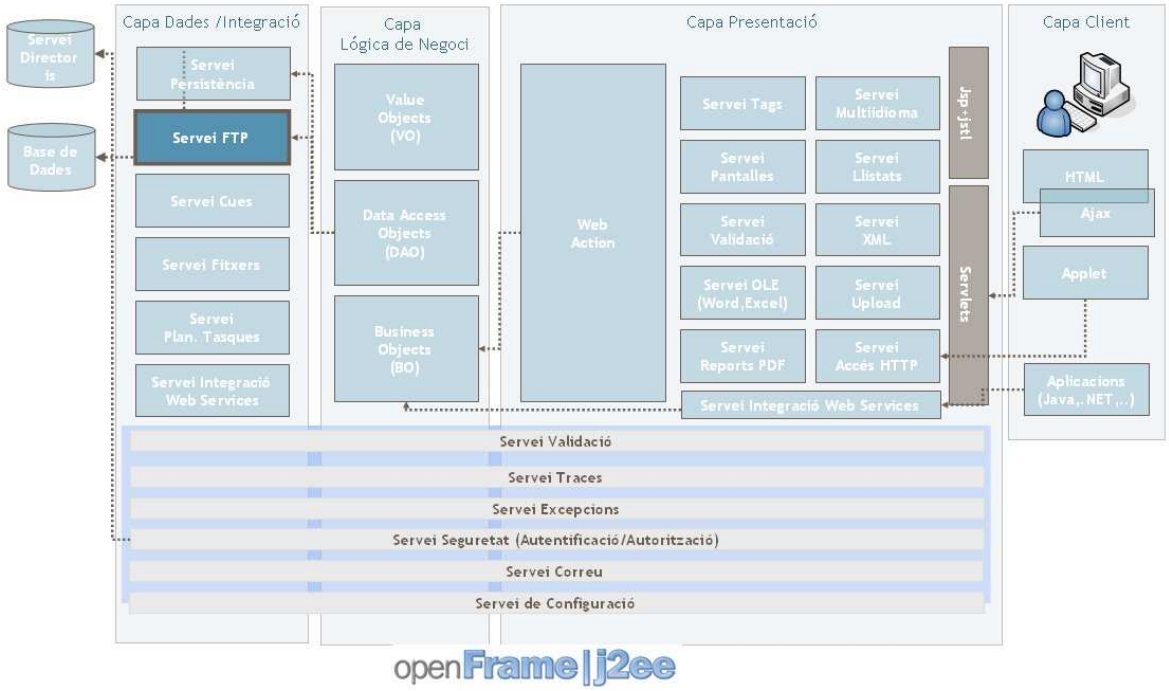
1.1. Propòsit

El servei de FTP d'openFrame permet enviar i rebre: arxius, directoris sencers o Streams, entre el servidor on s'executa l'aplicació a altres servidors.

El servei està basat en Jakarta Commons Net, es tracta d'un projecte *open source* englobat dins del projecte Jakarta que proporciona les funcionalitats clàssiques associades al protocol FTP, però com a característica destacada la possibilitat de transferir els fitxers o els streams en paral·lel.

1.2. Context i Escenaris d'Ús

El servei de FTP es troba dins dels serveis d'integració d'openFrame.



El seu ús és necessari en cas de voler intercanviar fitxers entre servidors utilitzant el protocol FTP.

1.3. Versions i Dependències

En el present apartat es mostren quines són les versions i dependències necessàries per fer ús del Servei.
Dins la llista de dependències es mostren diferenciades:

- Dependències bàsiques. Llibreries necessàries per fer ús del servei.
- Dependències addicionals. Aquestes dependències són necessàries per poder fer ús de característiques concretes del servei o per l'ús dels tests unitaris proporcionats amb el servei.

1.3.1. Versions

No s'han produït canvis respecte la versió 1.2.

1.3.2. Dependències Bàsiques

Nom	Tipus	Versió	Descripció
openFrame-core	jar	1.0	
openFrame-services-logging	jar	1.0	Utilitzar també les dependències del Servei de Logging
openFrame-services-exceptions	jar	1.0	Utilitzar també les dependències del Servei d'Excepcions
spring	jar	1.2.5	http://www.springframework.org
Jakarta Commons Net	jar	1.4.0	Llibreria open-source que facilita l'utilització del protocol FTP
servlet-api	jar	2.4	

En cas de utilitzar un servei de openFrame és important que es facin servir també les dependències del servei en qüestió.

1.3.3. Dependències Addicionals

- Proves Unitàries del Servei

Nom	Tipus	Versió	Descripció
junit	jar	3.8.1	
Spring-mock	jar	1.2.5	http://www.springframework.org

Veure l'apartat 'Instal·lació i configuració' per a més detall.

1.4. A qui va dirigit

Aquest document va dirigit als següents perfils:

- Programador. Per conèixer l'ús del servei.
- Arquitecte. Per conèixer quins són els components i la configuració del servei.
- Administrador. Per conèixer com configurar el servei en cadascun dels entorns en cas de necessitat.

1.5. Documents i Fonts de Referència

[1] Jakarta Commons Net http://jakarta.apache.org/commons/index.html

1.6. Glossari

FTP
File Transfer Protocol (Protocol de Transferència d'Arxius), es tracta d'un protocol de transferència de fitxers entre computadors.

Stream
La traducció de l'anglès significa: corrent, fluxe, fluir... S'asocia el terme a un fluxe de dades sense suport físic (fitxer).

Download
Rebre arxius des d'un servidor.

Upload
Enviar arxius a un servidor.

Socket
Port del servidor habilitat per realitzar la transferència de fitxers/streams mitjançant el protocol FTP.

2. Descripció Detallada

2.1. Arquitectura i Components

openFrame ofereix una arquitectura d'ús del protocol FTP totalment deslligada de qualsevol implementació.

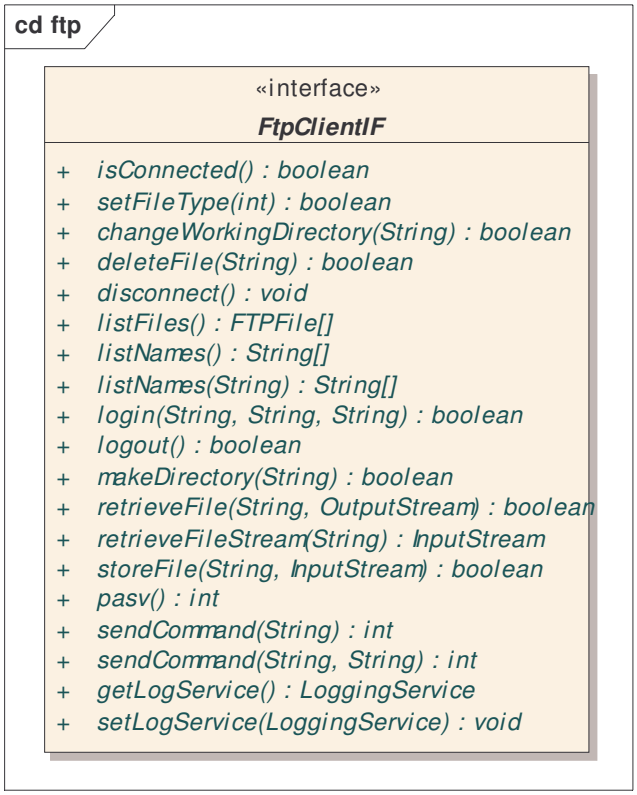
Els components podem classificar-los en:

- Interfícies i Components Genèrics. Interfícies del servei i components d'ús general amb independència de la implementació escollida.

- Implementació de les interfícies basada en Jakarta Commons Net.

2.1.1. Interfícies i Components Genèrics

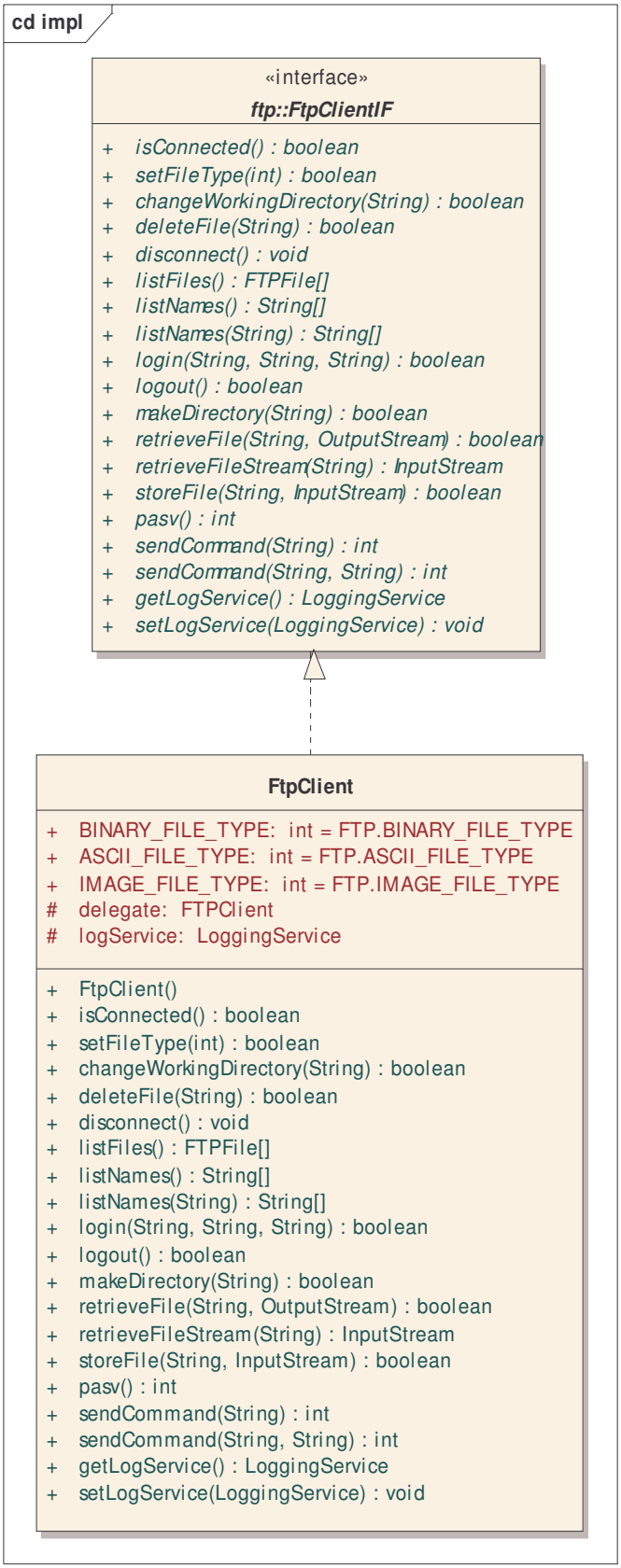
El servei d'utilització del protocol FTP defineix la següent interfície:



Component	Package	Descripció
FtpClientIF	net.opentrends.openframe.services.ftp	Interfície bàsica que permet accedir-hi a les funcionalitats del servei. **Ofereix entre d'altres (veure la documentació disponible al Javadoc per a més referència):** • **public boolean login(String userName, String password, String host): realitza una connexió al servidor FTP utilitzant un usuari i el seu password.** • **public InputStream retrieveFileStream(String serverFile): retorna un fitxer del servidor FTP dins d'una variable InputStream.** • **public void storeFile(String serverFileUploaded,**

Component	Package	Descripció
		InputStream inputStream): guarda al servidor en un fitxer amb nom assignat en la varible "serverFileUploaded" el stream passat.

2.1.2. Components implementació de Jakarta Commons Net


cd impl
«interface»
ftp::FtpClientIF
+ isConnected() : boolean
+ setFileType(int) : boolean
+ changeWorkingDirectory(String) : boolean
+ deleteFile(String) : boolean
+ disconnect() : void
+ listFiles() : FTPFile[]
+ listNames() : String[]
+ listNames(String) : String[]
+ login(String, String, String) : boolean
+ logout() : boolean
+ makeDirectory(String) : boolean
+ retrieveFile(String, OutputStream) : boolean
+ retrieveFileStream(String) : InputStream
+ storeFile(String, InputStream) : boolean
+ pasv() : int
+ sendCommand(String) : int
+ sendCommand(String, String) : int
+ getLogService() : LoggingService
+ setLogService(LoggingService) : void
FtpClient
+ BINARY_FILE_TYPE: int = FTP.BINARY_FILE_TYPE
+ ASCII_FILE_TYPE: int = FTP.ASCII_FILE_TYPE
+ IMAGE_FILE_TYPE: int = FTP.IMAGE_FILE_TYPE
# delegate: FTPClient
# logService: LoggingService
+ FtpClient()
+ isConnected() : boolean
+ setFileType(int) : boolean
+ changeWorkingDirectory(String) : boolean
+ deleteFile(String) : boolean
+ disconnect() : void
+ listFiles() : FTPFile[]
+ listNames() : String[]
+ listNames(String) : String[]
+ login(String, String, String) : boolean
+ logout() : boolean
+ makeDirectory(String) : boolean
+ retrieveFile(String, OutputStream) : boolean
+ retrieveFileStream(String) : InputStream
+ storeFile(String, InputStream) : boolean
+ pasv() : int
+ sendCommand(String) : int
+ sendCommand(String, String) : int
+ getLogService() : LoggingService
+ setLogService(LoggingService) : void

Component	Package	Descripció
FtpClient	net.opentrends.openframe.services.ftp.impl	Implementació de la interfície FtpClientIF pel component Jajarta Commons Net

Veure la documentació disponible al Javadoc per a més referència.

2.2. Instal·lació i Configuració

2.2.1. Instal·lació

La instal·lació del servei requereix de la utilització de la llibreria 'openFrame-services-ftp' i les dependències indicades a l'apartat 'Introducció - Versions i Dependències'.

Per la instal·lació del component Jakarta Commons Net, no es requereix res a part del JAR indicat en la dependència "Jakarta Commons Net".

2.2.2. Configuració

La configuració del Servei de FTP implica:

- Definir el servei.

Definició del Servei

```
<bean id="ftpService"
      ...>
```

La definició del servei requereix configurar un bean amb un identificador (es recomana usar 'ftpService') i els següents atributs:

Atributs:

Atribut	Requerit	Descripció
class	Sí	Implementació concreta del servei a utilitzar. Opcions: • net.opentrends.openframe.services.ftp.impl.FtpClient

També es poden configurar les següents propietats:

Propietat	Requerit	Descripció
logService	No	Referència a la definició del servei de traces.

Exemple:

```
...
   <!-- FTP service  -->
   <bean id="ftpService"
class="net.opentrends.openframe.services.ftp.impl.FtpClient">
          <property name="logService">
                 <ref local="loggingService"/>
          </property>
   </bean>

...
   <bean id="loggingService"
class="net.opentrends.openframe.services.logging.log4j.Log4JServiceImpl" init-
method="init">
          ...
   </bean>
...
```

2.3. Utilització del Servei

2.3.1. Realitzar connexió

Tal i com s'ha comentat a l'apartat 'Arquitectura i Components' les classes principals per realitzar transferències de fitxers/streams es troben al package 'net.opentrends.openframe.services.ftp'.

Per realitzar una connexió seguirem un patró com el mostrat en el següent exemple:

```
if (this.ftpClient != null ) {
   ftpClient.login(userName, password, host);
}
```

Realitzarem doncs els següents passos:

1) En primer lloc comprovarem que s'ha realitzat correctament la injecció a la nostra classe del servei de FTP.
2) Seguidament cridem al mètode `login()` per realitzar la connexió i l'autentificació al servidor remot mitjançant el protocol FTP.

Es crida el mètode `login()` que amb els següents paràmetres:

ordre	Requerit	Tipus	Descripció
1	Si	String	Nom de l'usuari de connexió mitjançant el protocol FTP.
2	Si	String	Contrasenya de l'usuari de connexió mitjançant el protocol FTP.
3	Sí	String	Nom del servidor remot (o IP) on es vol obrir una connexió mitjançant el protocol FTP.

El mètode retorna un valor booleà (TRUE/FALSE) indicant si la connexió s'ha realitzat correctament.

2.3.2. Realitzar desconnexió del servidor FTP

Per realitzar la desconnexió del servidor FTP seguirem un patró com el mostrat en el següent exemple:

```
if (this.ftpClient != null ) {
   if (ftpClient.isConnected()) {
      ftpClient.disconnect();
   }
}
```

Realitzarem doncs els següents passos:

1) En primer lloc comprovarem que s'ha realitzat correctament la injecció a la nostra classe del servei de FTP.
2) Realitzar la desconnexió del servidor FTP cridant al mètode `disconnect()`.

2.3.3. Realitzar download d'un fitxer

Per realitzar un download d'un fitxer seguirem un patró com el mostrat en el següent exemple:

```
if (this.ftpClient != null ) {
   ftpClient.login(userName, password, host);
   if (ftpClient.isConnected()) {
      InputStream stream = ftpClient.retrieveFileStream(serverFile);
      ftpClient.disconnect();
   }
}
```

Realitzarem doncs els següents passos:

1) En primer lloc comprovarem que s'ha realitzat correctament la injecció a la nostra classe del servei de FTP.
2) Seguidament cridem al mètode `login()` per realitzar la connexió i l'autentificació al servidor remot mitjançant el protocol FTP.

3) En cas de que la connexió al servidor remot s'hagi realitzat correctament es crida al mètode `retrieveFileStream()` per realitzar el download del fitxer del servidor remot.
4) Realitzar la desconnexió del servidor FTP.

Es crida el mètode `retrieveFileStream()` que amb els següents paràmetres:

ordre	Requerit	Tipus	Descripció
1	Si	String	Nom del fitxer del servidor remot que es vol realitzar el download.

I retorna:

ordre	Requerit	Tipus	Descripció
1	Si	InpuStream	Variable InputStream amb les dades del fitxer remot

2.3.4. Realitzar upload d'un fitxer

Per realitzar un upload d'un fitxer seguirem un patró com el mostrat en el següent exemple:

```
if (this.ftpClient != null ) {
   ftpClient.login(userName, password, host);
   if (ftpClient.isConnected()) {
      ftpClient.storeFile(serverFileUploaded, input);
      ftpClient.disconnect();
   }
}
```

Realitzarem doncs els següents passos:

1) En primer lloc comprovarem que s'ha realitzat correctament la injecció a la nostra classe del servei de FTP.
2) Seguidament cridem al mètode `login()` per realitzar la connexió i l'autentificació al servidor remot mitjançant el protocol FTP.
3) En cas de que la connexió al servidor remot s'hagi realitzat correctament es crida al mètode `storeFile()` per realitzar el upload del fitxer del servidor local al servidor remot.
4) Realitzar la desconnexió del servidor FTP.

Es crida el mètode `storeFile()` que amb els següents paràmetres:

ordre	Requerit	Tipus	Descripció
1	Si	String	Nom del fitxer del servidor remot on es vol deixar el fitxer.
2	Si	InputStream	Variable amb les dades a pujar al servidor FTP

2.3.5. Obtenir la llista de noms dels fitxers del servidor remot

Per obtenir la llista de fitxers en una carpeta d'un servidor remot seguirem un patró com el mostrat en el següent exemple:

```
if (this.ftpClient != null ) {
   ftpClient.login(userName, password, host);
   if (ftpClient.isConnected()) {
          String[] listFileNames = ftpClient.listNames(serverFolder)
   }
}
```

Realitzarem doncs els següents passos:

1) En primer lloc comprovarem que s'ha realitzat correctament la injecció a la nostra classe del servei de FTP.
2) Seguidament cridem al mètode `login()` per realitzar la connexió i l'autentificació al servidor remot mitjançant el protocol FTP.
3) En cas de que la connexió al servidor remot s'hagi realitzat correctament es crida al mètode `listNames()` per obtenir una llista de fitxers en una carpeta d'un servidor remot.

Es crida el mètode `listNames()` que amb els següents paràmetres:

ordre	Requerit	Tipus	Descripció
1	Sí	String	Nom de la carpeta del servidor remot d'on es vol obtenir la llista de fitxers.

El mètode retorna un array de Strings amb els noms dels fitxers de la carpeta del servidor remot.

2.3.6. Executar una comanda FTP

Per executar una comanda FTP seguirem un patró com el mostrat en el següent exemple:

```
if (this.ftpClient != null ) {
   ftpClient.login(userName, password, host);
   if (ftpClient.isConnected()) {
      String line = ftpClient.sendCommand(command);
   }
}
```

Realitzarem doncs els següents passos:

1) En primer lloc comprovarem que s'ha realitzat correctament la injecció a la nostra classe del servei de FTP.
2) Seguidament cridem al mètode `login()` per realitzar la connexió i l'autentificació al servidor remot mitjançant el protocol FTP.
3) En cas de que la connexió al servidor remot s'hagi realitzat correctament es crida al mètode `sendCommand()` per executar una comanda FTP en el servidor remot..

Es crida el mètode `sendCommand()` que amb els següents paràmetres:

ordre	Requerit	Tipus	Descripció
1	Sí	String	Comanda FTP a executar.

El mètode retorna un String el resultat de l'execució de la comanda FTP en el servidor remot.

2.4. Eines de Suport

2.5. Integració amb Altres Serveis

2.6. Preguntes Freqüents

3. Exemples

3.1. Tests Unitaris

Un exemple d'utilització del servei de FTP són els tests unitaris, a on s'obté el bean del servei a partir del fitxer de definició (applicationContext.xml) i s'envia un log amb les sortides especificades.

S'ha de tenir en compte que s'ha de disposar d'accés a un servidor remot mitjançant el protocol FTP per poder realitzar els tests.

```
package net.opentrends.openframe.services.ftp.test;

...
```

La utilització del servei es independent de la configuració del mateix. Així, un possible fitxer de configuració del servei seria:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>

   <!-- FTP service  -->
   <bean id="ftpService"
class="net.opentrends.openframe.services.ftp.impl.FtpClient">
          <property name="logService">
                 <ref bean="loggingService" />
          </property>
   </bean>


   <!-- LOGGING service  -->
   <bean id="loggingService"
class="net.opentrends.openframe.services.logging.log4j.Log4JServiceImpl" init-
method="init">
          <property name="configurator">
                 <ref local="loggingConfigurator"/>
          </property>
   </bean>

   <!-- configurator bean -->
   <bean id="loggingConfigurator"
class="net.opentrends.openframe.services.logging.log4j.xml.HostDOMConfigurator">
          <property name="configFileName">
                 <value>classpath:log4j-test.xml</value>
          </property>
   </bean>


</beans>
```

```
...

   public void testConnectAndLogin() throws Exception {
          FtpClientIF ftpClient = getFtpService();

          assertTrue(ftpClient.login(userName, password, host));

          if (ftpClient.isConnected()) {
                 ftpClient.disconnect();
          }

          if (getLogService() != null) {
                 getLogService().getLog(this.getClass().getName()).debug("End
testConnectAndLogin()");
          }
   }


   public void testConnectFailed() throws Exception {
          password       = "incorrectPasword";

          FtpClientIF ftpClient = getFtpService();

          assertFalse(ftpClient.login(userName, password, host));

          if (ftpClient.isConnected()) {
                 ftpClient.disconnect();
          }

          if (getLogService() != null) {
                 getLogService().getLog(this.getClass().getName()).debug("End
testConnectFailed()");
          }
   }


   public void testSendRawCommand() throws Exception {
          FtpClientIF ftpClient = getFtpService();
          ftpClient.login(userName, password, host);

          int returnCodeOk      = 257;
          int returnCode                = ftpClient.sendCommand("PWD");

          if (ftpClient.isConnected()) {
                 ftpClient.disconnect();
          }

          assertEquals(returnCodeOk, returnCode);
          if (getLogService() != null) {
                 getLogService().getLog(this.getClass().getName()).debug("End
testSendRawCommand()");
          }
   }


   public void testListFileNames() throws Exception {
          FtpClientIF ftpClient = getFtpService();
          ftpClient.login(userName, password, host);

          String[] listFileNames = ftpClient.listNames(serverFolder);
```

```
            if (ftpClient.isConnected()) {
                    ftpClient.disconnect();
            }

            assertTrue(0<listFileNames.length);
            if (getLogService() != null) {
                    getLogService().getLog(this.getClass().getName()).debug("End
testListFileNames()");
            }
    }


    public void testUploadFile() throws Exception {
            FtpClientIF ftpClient = getFtpService();
            ftpClient.login(userName, password, host);

          InputStream input;
          input = new FileInputStream(localFolder+localFile);

          String serverFileUploaded = localFile;
          ftpClient.storeFile(serverFileUploaded, input);

          input.close();

            // validate file at server
            boolean existsAtServer = false;
            String[] listFileNames = ftpClient.listNames(serverFolder);

            if (listFileNames != null) {
                    for (int i=0; i<listFileNames.length; i++) {
                            String fileName = (String)listFileNames[i];
                            if (localFile.equals(fileName)) {
                                    existsAtServer = true;
                                    break;
                            }
                    }
            }

            if (ftpClient.isConnected()) {
                    ftpClient.disconnect();
            }

            assertTrue(existsAtServer);

            if (getLogService() != null) {
                    getLogService().getLog(this.getClass().getName()).debug("End
testUploadFile()");
            }
    }


    public void testDownloadFile() throws Exception {
            FtpClientIF ftpClient = getFtpService();
            ftpClient.login(userName, password, host);

            InputStream inputStream = ftpClient.retrieveFileStream(serverFile);
            byte[] buf = new byte[4096];
            FileOutputStream fileDownload = new
FileOutputStream(localFolder+serverFile);
            for (int len=-1;(len=inputStream.read(buf))!=-1;) {
                    fileDownload.write(buf,0,len);
            }
            fileDownload.flush();


            if (ftpClient.isConnected()) {
                    ftpClient.disconnect();
```

```
            }

            File file = new File(localFolder+serverFile);
            assertTrue(file.exists());
            if (getLogService() != null) {
                    getLogService().getLog(this.getClass().getName()).debug("End
testDownloadFile()");
            }
   }


   private FtpClientIF getFtpService() {
         BeanFactory beanFactory = new
ClassPathXmlApplicationContext("applicationContext.xml");
         FtpClientIF ftpClient = (FtpClientIF)beanFactory.getBean("ftpService");
         logService = ftpClient.getLogService();

         assertNotNull(ftpClient);
         return ftpClient;
   }
...
```

4. Annexos