# JADE - User Manual

Quick intro to the JADE User Manual:

- for navigation use the menu in the sidebar for more detailed information such as:
  - Introduction & Information Sources
  - Architecture & Components
  - Getting Started & Using the JADE Client
- for contacts to SOS
  - send your comments, questions or if you are missing information as Feedback
  - send your questions about support & services by e-mail to: sales@sos-berlin.com
  - self-register in our Change Management System for tracking of issues, commenting and voting for issues

JOBSCHEDULER

**let the rabbit do the job**

---

## Recent Activity

- **Andreas Püschel**
  - JADE - Components updated 7 minutes ago • view change
  - JADE - User Manual updated 31 minutes ago • view change
  - JADE User Manual - File Selection, Handling and Renaming and the Target Directory updated 38 minutes ago • view change
  - JADE User Manual - Configuring and Running File Transfers updated 39 minutes ago • view change
  - JADE User Manual - Configuration Formats & Tools updated 39 minutes ago • view change

## Documentation Contributors

- Andreas Püschel (31 minutes ago)
- Alan Amos (4 hours ago)

## Articles to be reworked

- JADE User Manual - Credential Store
- JADE User Manual - Using the JADE Background Service
- JADE User Manual - Logging & Notification
- JADE - Installation - Client
- JADE User Manual - Jump Host File Transfer

**Pages**

**Navigation**

# JADE User Manual - Introduction

## Solution Scope

- JADE is a solution for Managed File Transfer.
- JADE addresses particular implementation, logging and reporting requirements not found in other solutions.
- JADE also solves error detection and error handling problems often found with shell script based file transfer implementations .
- JADE is available standalone or fully integrated into JobScheduler to enable JADE's managed file transfer capability to be integrated into automated workflows.

## Basic Features

- See the JADE - Features Knowledge Base article.

## Target Audience for this Document

- This manual provides information about the JADE architecture and components as well as instructions for file transfer configuration.
- It is intended to be used by system engineers who are responsible for file transfer configuration. Background knowledge of operating systems, file transfer protocols and network administration is a prerequisite for using JADE.
- This manual is targeted at both users who are new to JADE as well as those with more experience.

# JADE User Manual - Information Sources

- Architecture
- Tutorials
- Documentation

## Architecture

For a brief introduction to the JADE  architecture see:

- JADE User Manual - Architecture

For a broader and more thorough presentation see:

- JADE Architecture Guide

## Tutorials

- The JADE tutorials provide a basis for learning how to use the JADE Client. They are intended to function as a Getting Started Guide before proceeding to increasingly complex examples.
- The tutorials describe operation of the JADE Client from the command line, e.g. by using a batch script. The other methods of operating the Client are:
    - using the JADE API
    - using the JobScheduler's JADE JITL Jobs

Read more ...

## Documentation

- JADE - User Manual
    - The current document
- JADE - Reference Documentation
    - The JADE Parameter Reference explains the parameters for file transfer configuration.
    - The JADE API Reference shows the objects, methods and properties of the JADE API.
- The JADE section of the Product Knowledge Base
    - The HowTos, FAQs and Examples should be of particular use to users learning how to use JADE.

# JADE User Manual - Getting Started

- Prerequisites
- Download & Installation
- Tutorials
- FAQ, How To, Examples
- Getting Support

## Prerequisites

- An Oracle Java Runtime Environment 1.8 or later is required for JADE release 1.9 and newer.
- The JADE Client is pure Java and can be operated with any operating system architecture (32bit, 64bit)
    - Verify platform support from the Which platforms is JobScheduler provided for? article.

## Download & Installation

- Download the JADE Client from our JADE Download Page or from SourceForge.
- Follow the instructions given in the JADE - Installation - Client article.

## Tutorials

**Note**

These tutorials use the text-based file transfer configuration with `*.ini` files implemented up to Release 1.10.

They will still, however, provide a step-by-step introduction to getting JADE up and running before Release 1.11 is available.

In addition, two updated example configurations using the XML configuration introduced with Release 1.11 are described in this manual.

- Required Reading
  - Follow The JADE Client Command Line Interface - Tutorial 1 - Getting Started
- Recommended Reading
  - Follow The JADE Client Command Line Interface - Tutorial 2 - Simple File Selection
  - Follow The JADE Client Command Line Interface - Tutorial 3 - More Advanced File Selection

## FAQ, How To, Examples

- See the JADE - FAQ
- See the JADE - How To
- See the JADE - Examples

## Getting Support

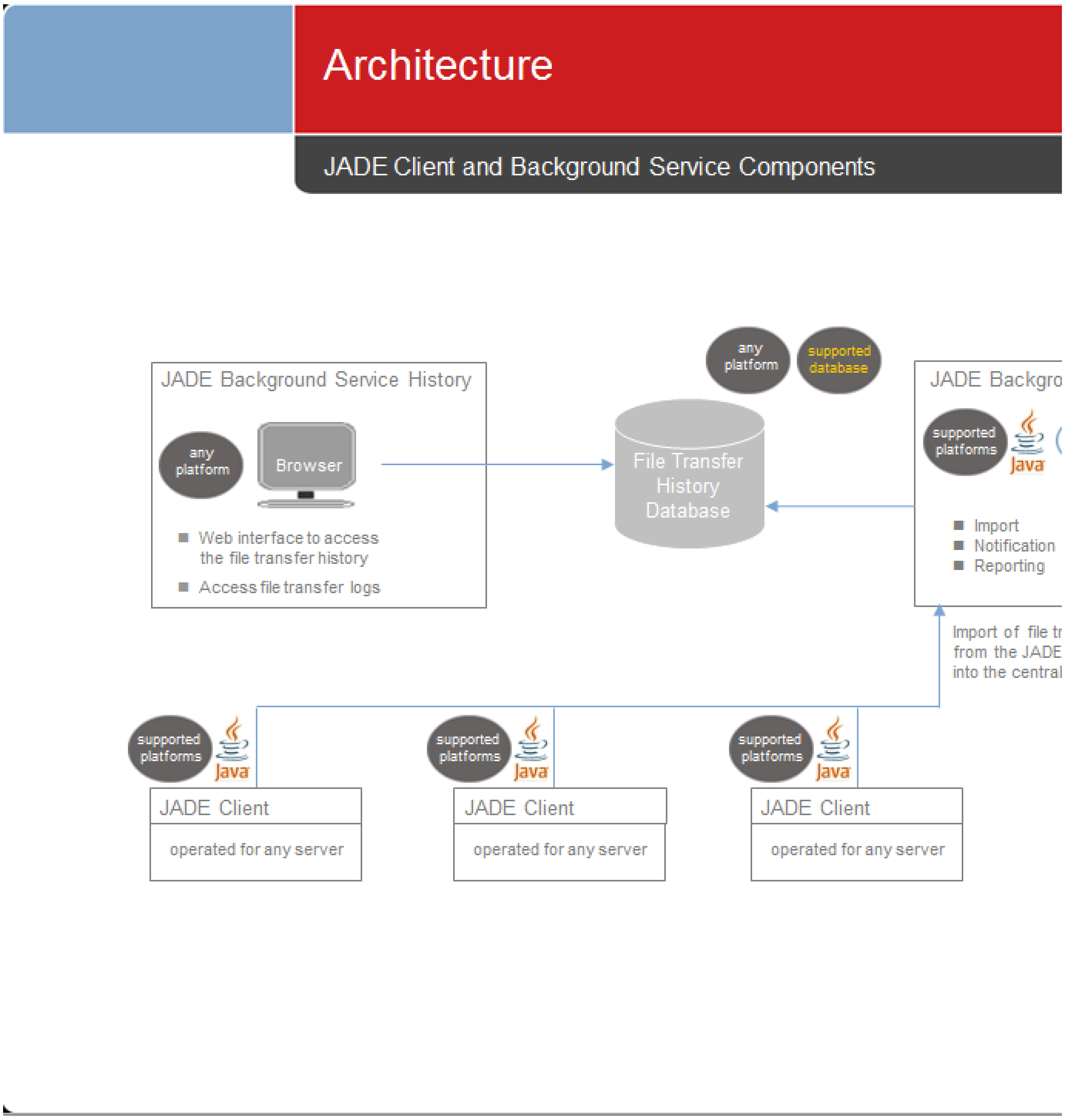
- Ticket System for bug reports, change requests and feature requests
  - SourceForge JADE Ticket System
  - SOS Ticket System for customers of a Commercial License
- User Forum for questions & answers
  - SourceForge JADE Discussion Forums

# JADE User Manual - Architecture

## Architecture

JADE comes with two main components: the *Client*, which is responsible for the File Transfer itself and the *Background Service*, which handles the File Transfer history.

Any number of Clients can use a single instance of the Background Service History as shown in the following diagram:



## Components

The JADE Client and Background Service are both made up of a number of sub-components:

### JADE Client

- JADE Command Line Interface
  - Command line utility for integration with batch files
- JADE API
  - Enables applications to use the JADE functionality.
- JADE JITL Jobs
  - Seamless integration with JobScheduler for automated transfer

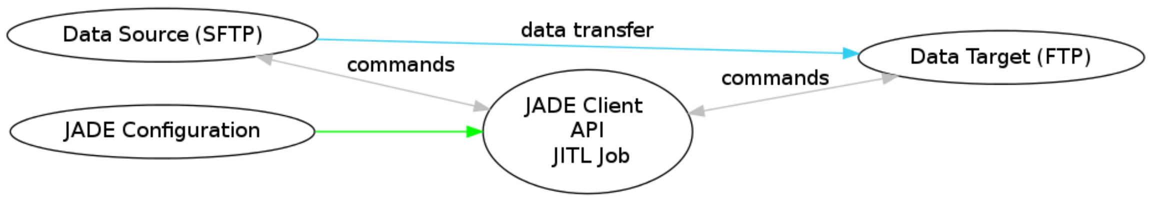
### JADE Background Service

- JADE Background Service Automation
    - Imports the transfer history of the JADE Clients into a database
    - Provides notifications in case of errors
- JADE Background Service History Viewer
    - GUI for monitoring file transfers
    - Allows to search and browse the file transfer history
- JADE Background Service Reporting
    - Generates custom reports for file transfers
    - Mails reports of effected transfers

## JADE Implementation Architecture

A unique key feature of JADE is Server-to-Server File Transfer (S2S). It provides the possibility of setting up a number of file transfers using a single server without installing clients/agents on each source/target host.

This is shown schematically in the diagram below:



JADE can transfer data using an intermediate server from a Source Server-to-Target Server (S2S) without touchdown on an intermediate server.

## Further information

For more information please see:

- our JADE Implementation Architecture page

**Pages**

**Navigation**

# JADE User Manual - Components

## Introduction

The main purpose of JADE is to transfer data such as files between data sources and data targets. In addition it can execute commands by SSH on some hosts and record a file transfer history. A detailed list of JADE's file transfer features can be found in the JADE - Features article.

The components provided by JADE can be grouped under two main headings - the JADE Client and the JADE Background Service.

## JADE Client Components

The JADE Client is the main component in JADE. The client is not only responsible for file transfer and SSH command execution but can carry out customizable logging, send the transfer history to the JADE Background Service, enable an audit trail and provide error handling. The architecture of the JADE Client and its associated components is described in detail in the JADE Implementation Architecture.

The JADE client can be operated through one of the following components.

### JADE Command Line Interface

The Command Line Interface allows JADE to be operated directly from the command line or from batch files.

- The JADE Command Line Interface is used e.g. for integration with existing scripts and to replace direct calls to FTP and SFTP command line programs.
- The JADE Command Line Interface allows to change the file transfer behavior, e.g. switch of protocol from FTP to SFTP, without the need to modify existing scripts. Such modifications can be applied to a JADE configuration file and would not affect existing file transfer scripts.

Read more ...

### JADE API

The JADE API is used as a component for an individual application that requires file transfer capabilities.

- The JADE API makes the functionality of JADE available to any application running in a Java Virtual Machine.
- The JADE API provides access to classes and methods for individual handling of file transfer needs.

Read more ...

### JADE JITL Jobs

Seamless integration with the SOS JobScheduler using the JobScheduler's JADE JITL Jobs (JobScheduler Integrated Template Library) means that the full range of the JobScheduler's workload automation capabilities can be used to

- automatically start and control file transfers using JobScheduler features such as Job Chains and Schedules,
- carry out advanced pre- and post-procesing around file transfer.

Read more ...

## JADE Background Service Components

The JADE Background Service provides its functionality by use of the SOS JobScheduler.

- It includes operations to collect the transfer history and log files from a number of JADE Clients and to add this information to the transfer history database.
- It includes the JADE Background Service History, a graphical interface to visualize the transfer history.

### JADE Background Service Automation

The JADE Background Service Automation is provided by a JobScheduler instance. This service can import the transfer history and log files from any number of JADE Clients into the transfer history database as well as sending notifications.

- The JADE Background Service collects the transfer history from individual JADE Clients and is either invoked by the JADE Client after each file transfer or started by the automatically by the JobScheduler.
- The Background Service feeds the transfer history database with information on each file transfer including file names, source and target systems and execution results.

Read more ...

### JADE Background Service History Viewer

The JADE Background Service History Viewer is a web front-end for accessing the file transfer history from the file transfer database which is used by the file JADE Background Service.

Read more ...

## Resources

### JADE General Information

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

### JADE Client Component

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

### JADE Command Line Component

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

### JADE JITL Job Component

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

### JADE Background Service Component

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

### JADE API Component

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

Error formatting macro: contentbylabel: java.lang.IllegalArgumentException: Null parameter in parameters list

**Pages**

- JADE Client
    - JADE Command Line Interface
    - JADE JITL Jobs
    - JADE API
- JADE Background Service
    - JADE Background Service Automation
    - JADE Background Service History Viewer
    - JADE Background Service database usage

**Navigation**

# JADE User Manual - Installation

## Installation of JADE Client

- JADE - Installation - Client

## Installation of JADE Background Service

- JADE - Installation - Background Service
- JADE - Installation - Background Service History Viewer

# JADE User Manual - Using the JADE Client

## Introduction

The JADE Client can be operated from the command line and batch files. In addition it can be operated by jobs that are provided as part of the SOS JobScheduler's JITL library and by way of an API, which is available for integration with other applications.
This section of the User Manual will concentrate on using the Client via the JADE Command Line Interface.

## Change in JADE Configuration Format

Use of the JADE Client will be significantly changed with the introduction of an XSD schema and XML configuration files with version 1.11.

### Up to Release 1.10.x - `*.ini` File Configuration

File transfer configuration has been carried out using text-based `*.ini` files up to and including Releases 1.10.x

A series of tutorials is available that provides guidance through setting up JADE and configuring basic file transfer situations.

A brief description of a simple `*.ini` file transfer configuration is available in the Configuration Formats & Tools section of this manual.

### Release 1.11.x onwards - XML Based Configuration

An XML based configuration using an XSD schema will be introduced with Release 1.11. The use of the schema will considerably simplify file transfer configuration for the JADE Client - particularly for complex file transfer situations.

A form-based editor is being developed that will guide the user during the configuration process and ensure that configurations are validated.

File transfer configurations specified for Client versions up to 1.10.x will need to be modified before they can be run with versions 1.11 onwards. A Parameter Mapping table is available to help conversion to the XML based configuration.

### Further Information

See Configuring File Transfers with the JADE Client for more information about the changes involved.

**Pages**

- JADE User Manual - Configuration Formats & Tools
- JADE User Manual - Configuring and Running File Transfers
    - JADE User Manual - Configuration 1 - An Overview
    - JADE User Manual - Configuration 2 - The Fragments Branch
    - JADE User Manual - Configuration 3 - The Profile Branch
- JADE User Manual - Example Configurations
    - JADE User Manual - Simple File Transfer with Basic Authentication
    - JADE User Manual - File Transfer with SSH authentication
- JADE User Manual - Connection Protocols
- JADE User Manual - Authentication
- JADE User Manual - Credential Store
- JADE User Manual - File Selection, Handling and Renaming and the Target Directory
- JADE User Manual - The Operation - What is to be done?

**Navigation**

# JADE User Manual - Configuration Formats & Tools

## Introduction

A significant change in the file transfer configuration of JADE will be introduced with release 1.11.This will mark the end of the use of text-based `*.ini` files and the introduction of XML-based configurations in line with an XSD schema. In parallel, a new XML Editor will be available that will automatically generate required elements where appropriate as well as allow configurations to be verified according to the schema.

The two configuration formats are compared on this page.

## Text-based file transfer configuration with `*.ini` Files up to Release 1.10

File transfers are configured in JADE up to Release 1.10 in text-based `.ini` files.

A `*.ini` *file* contains a number of *transfer profiles*: blocks that contain the actual configuration parameters. As profiles can be used to specify other profiles, it is possible to develop a set of reusable "configuration building blocks'" that can be set together as required.

The `*.ini` file and transfer profile to be used for a file transfer operation are specified when a command is sent to the JADE client.

`*.ini` files can be configured using a text or code editor.

### Limitations of the `*.ini` file approach

Whilst this approach is relatively straightforward to understand when used with simple file transfer situations:

- it is prone to configuration errors and
- becomes difficult to understand and therefore maintain when advanced and complex file transfer configurations are implemented.

For further Information about configuring JADE using `*.ini` approach see:

- The JADE Tutorials.
  These provide a step by step guide to using configuring and running JADE. Tutorial examples are designed to use our online demo server and can be tested with minimal configuration work. A configuration file is available for download to help you get the tutorial examples running as quickly as possible.
- An Example JADE settings file with profiles

## XML-based file transfer configuration from Release 1.11 onwards

## JADE - XML Configuration - XML Editor

- Manage JADE XML Configuration files
- Assign JADE XSD Schema to configuration files
- Validate XML configuration files
- The schema-aware XML Editor suggests elements and attributes for configuration
- The XML Editor includes the JADE - Reference Documentation
- The XML Editor will soon be improved to export compatible plain text configuration files (.ini) from valid XML configuration files for JADE releases before 1.11

The XML file transfer configuration introduced with release 1.11 of JADE allows parameter dependencies in JADE to be validated before the configuration file is transferred to an operating environment.

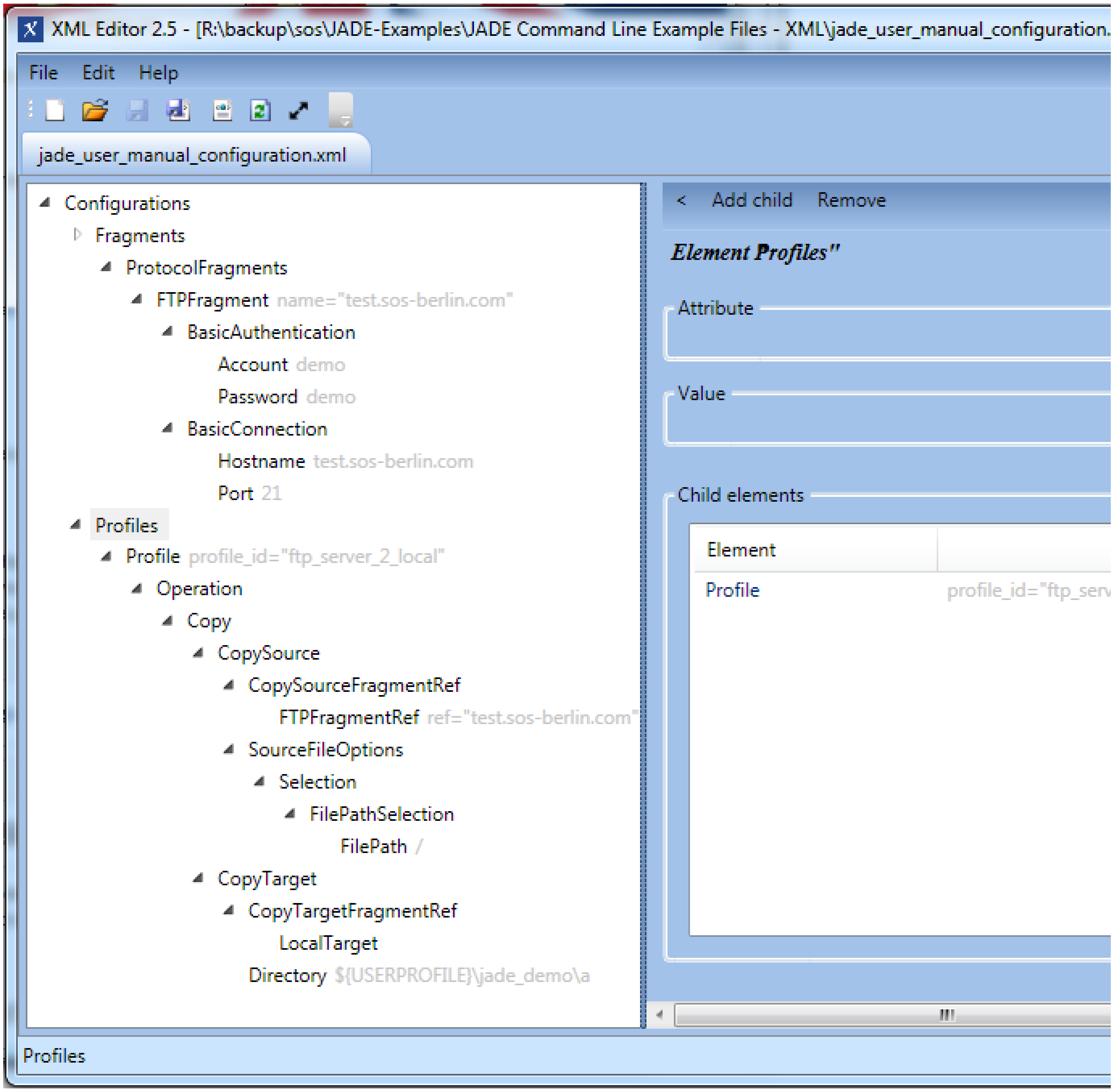
Dependencies and conflicts between configuration elements are avoided through the use of an XSD schema.

We have written an XML Editor that will simplify the creation and validation of schema-compatible file transfer configurations.

The basic principles behind the organization of the schema elements and parameters is described in Configuring File Transfers with the JADE Client.

A detailed description of individual parameters can be found in our Parameter Reference.

The following screen shot of the the SOS XML Editor shows the configuration for a Simple File Transfer with Basic Authentication example.


XML Editor 2.5 - [R:\backup\sos\JADE-Examples\JADE Command Line Example Files - XML\jade_user_manual_configuration.

```
Configurations
  Fragments
    ProtocolFragments
      FTPFragment name="test.sos-berlin.com"
        BasicAuthentication
          Account demo
          Password demo
        BasicConnection
          Hostname test.sos-berlin.com
          Port 21
  Profiles
    Profile profile_id="ftp_server_2_local"
      Operation
        Copy
          CopySource
            CopySourceFragmentRef
              FTPFragmentRef ref="test.sos-berlin.com"
            SourceFileOptions
              Selection
                FilePathSelection
                  FilePath /
          CopyTarget
            CopyTargetFragmentRef
              LocalTarget
            Directory ${USERPROFILE}\jade_demo\a
```

*Element Profiles"*

Profile profile_id="ftp_serv

Profiles

Although longer and seemingly more complex than a comparable configuration using the `*.ini` file approach (see The JADE Client Tutorial 1), the XML code for this example was produced using the XML Editor and therefore has been validated against the JADE Client XSD Schema. It is therefore conflict free and dependencies are correctly specified.

## Comparing `*.ini` File and XML Configurations

Comparison of the two formats reveal a significant difference in the concept behind the two approaches to configuration:

- **`*.ini` file configuration**
  Parameters such as *host*, *protocol*, *user*, *password* and *dir* (directory) are usually grouped separately in the `*.ini` file configuration depending on their use as source or target. This usually has no functional meaning (the files are processed from top to bottom) and is partly simply for convenience (grouping parameters together makes them easier to find) and partly to allow their reuse in the form of so-called profile fragments if required.
- **XML configuration**
  - Parameters are divided up into hierarchical branches - the *Profiles* and *Fragments* branches being the most relevant here. (Note that the expressions *Profiles* and *Fragments* are here defined differently to elements with the same names in the `*.ini` fil

e configuration.)
    - Authentication and connection parameters (*Account* & *Password* and *Hostname* & *Port* respectively) and the protocol (defined through the use of FTP) are part of the *Fragments* branch of the configuration.
    - The operation (*Copy*) and the source and target file paths (*FilePath* and *LocalTarget* respectively) are specified in the *Profiles* branch.
    - An *FTPFragmentRef* element in the *Profile* branch is used to specify the *FTPFragment* in the *Fragments* branch that contains the detailed information about the authentication and connection parameters.
- The division between *Fragments* and *Profiles* branches allows the reuse of elements whilst working within the strict hierarchical structure of XML.
    - The use of a reference in the *Profiles* branch (in the example, using the *FTPFragmentRef* element) to specify the authentication and connection parameters in the *Profiles* branch allows any number of Profiles elements to be defined and specified as required.

## Migrating from `*.ini` File to XML-based Configurations

### Parameter Mapping

With the change to the XML-based configuration not only has a hierarchical structure been introduced to configuration elements but many of the names used for configuration elements have been changed. We have therefore written a parameter mapping table to help users with conversion of their configurations.

## Further Information

### Related Sections of this User Manual:

- JADE User Manual - Configuration Formats & Tools (this page)
- JADE User Manual - Configuring and Running File Transfers
    - JADE User Manual - Configuration 1 - An Overview
    - JADE User Manual - Configuration 2 - The Fragments Branch
        - JADE User Manual - Configuring Alternative Fragments
    - JADE User Manual - Configuration 3 - The Profile Branch

See also:

- JobScheduler Monitoring Interface - XSD Schema Reference article for more detailed information about the XML-based configuration.
- Configuring and Running File Transfers which shows the configuration and command line calls for a simple file transfer in both settings.ini and XML-based configuration formats.

**Pages**

**Navigation**

# JADE User Manual - Configuring and Running File Transfers

## Introduction

Use of the JADE Client will be significantly changed with the introduction of an XSD schema and XML configuration files with Release 1.11.

Example configuration and command line calls for the XML-based configuration are presented on this page.

## XML-based file transfer configuration - available from Release 1.11 onwards

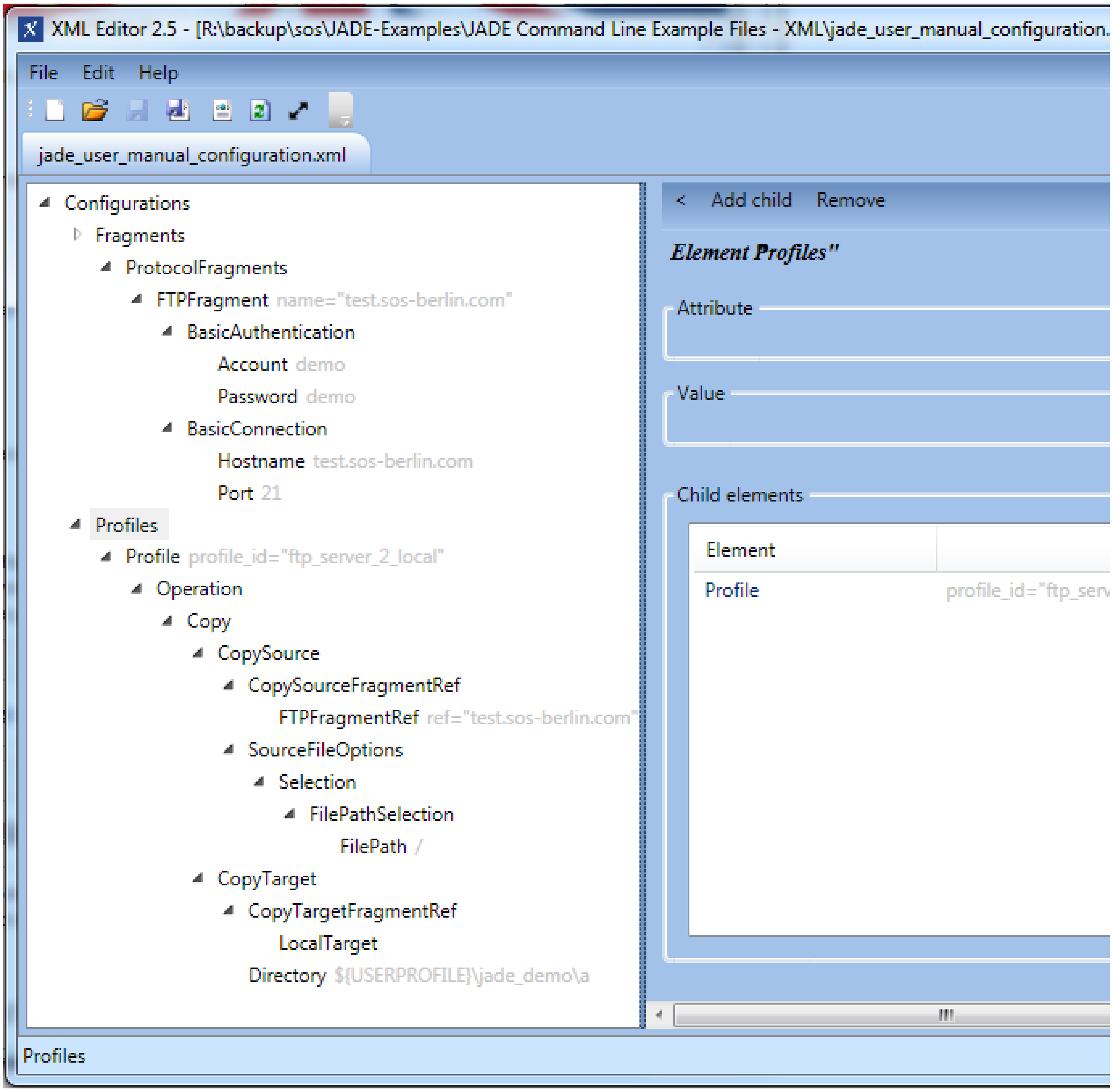
The XML file transfer configuration introduced with Release 1.11 of JADE allows parameter dependencies in JADE to be validated before the configuration file is transferred to an operating environment.

Dependencies and conflicts between configuration elements are avoided through the use of an XSD schema

The basic principles behind the organization of the schema and parameters will be described on this page. A detailed description of individual parameters can be found in our Parameter Reference and a description of the considerations involved in configuration of a simple file transfer can be found in the Simple File Transfer with Basic Authentication example.

The following screen shot of the SOS XML Editor shows the configuration of a simple FTP file transfer example - in this case the example downloads files from the SOS demo server to the local file system.

- The left hand pane of the Editor in the screen shot below shows the XML hierarchy of the configuration and the right hand pane the form used to display the child elements - in this case the child element of the Profiles element.


XML Editor 2.5 - [R:\backup\sos\JADE-Examples\JADE Command Line Example Files - XML\jade_user_manual_configuration.
File
Edit
Help
jade_user_manual_configuration.xml
<
Add child
Remove
Element Profiles"
Attribute
Value
Child elements
Element
Profiles

```
Configurations
  Fragments
    ProtocolFragments
      FTPFragment name="test.sos-berlin.com"
        BasicAuthentication
          Account demo
          Password demo
        BasicConnection
          Hostname test.sos-berlin.com
          Port 21
  Profiles
    Profile profile_id="ftp_server_2_local"
      Operation
        Copy
          CopySource
            CopySourceFragmentRef
              FTPFragmentRef ref="test.sos-berlin.com"
            SourceFileOptions
              Selection
                FilePathSelection
                  FilePath /
          CopyTarget
            CopyTargetFragmentRef
              LocalTarget
            Directory ${USERPROFILE}\jade_demo\a
```

| Element | |
|---|---|
| Profile | profile_id="ftp_serv |

The following code block shows the XML configuration file for the test host FTP file transfer example shown in the XML Editor screen shot above:

**A simple profile for transferring files by FTP from a test host to the local file system** Expand source

```
<?xml version="1.0" encoding="UTF-8"?>
<Configurations>
  <Fragments>
    <ProtocolFragments>
      <FTPFragment name="demo@test.sos-berlin.com">
        <BasicAuthentication>
          <Account><![CDATA[demo]]></Account>
          <Password><![CDATA[demo]]></Password>
        </BasicAuthentication>
        <BasicConnection>
          <Hostname><![CDATA[test.sos-berlin.com]]></Hostname>
          <Port><![CDATA[21]]></Port>
        </BasicConnection>
      </FTPFragment>
    </ProtocolFragments>
  </Fragments>
  <Profiles>
    <Profile profile_id="ftp_server_2_local">
      <Operation>
        <Copy>
          <CopySource>
            <CopySourceFragmentRef>
              <FTPFragmentRef ref="demo@test.sos-berlin.com" />
            </CopySourceFragmentRef>
            <SourceFileOptions>
              <Selection>
                <FilePathSelection>
                  <FilePath><![CDATA[/]]></FilePath>
                </FilePathSelection>
              </Selection>
            </SourceFileOptions>
          </CopySource>
          <CopyTarget>
            <CopyTargetFragmentRef>
              <LocalTarget />
            </CopyTargetFragmentRef>
            <Directory><![CDATA[${USERPROFILE}\jade_demo\a]]></Directory>
          </CopyTarget>
        </Copy>
      </Operation>
    </Profile>
  </Profiles>
</Configurations>
```

Although longer and seemingly more complex than a comparable settings file configuration (see JADE Tutorial 1), the XML code for this example was produced using the graphical editor and therefore has been validated against the JADE Client XSD Schema. It is error and conflict free and dependencies are correctly specified.

### Running a file transfer profile

A file transfer is started by specifying an XML configuration file and the profile in that file that is to be used.

For example, the profile listed above is called using:

**Calling a file transfer profile on a Windows system**

```
jade.cmd -settings="%USERPROFILE%\jade_demo\jade_user_manual_configuration.xml"
-profile="ftp_server_2_local"
```

**Calling a file transfer profile on a Unix system**

```
./jade.sh -settings="${HOME}/jade_demo/jade_user_manual_configuration.xml"
-profile="ftp_server_2_local"
```

## Further Information

### Related Sections of this User Manual:

- JADE User Manual - Configuration Formats & Tools
- JADE User Manual - Configuring and Running File Transfers (this page)
    - JADE User Manual - Configuration 1 - An Overview
    - JADE User Manual - Configuration 2 - The Fragments Branch
        - JADE User Manual - Configuring Alternative Fragments
    - JADE User Manual - Configuration 3 - The Profile Branch

See the Configuration Formats & Tools article for a comparison between the XML Configuration and its predecessor, the `*.ini` File.

See JobScheduler Monitoring Interface - XSD Schema Reference article for more detailed information about the XML configuration.

**Pages**

- JADE User Manual - Configuration 1 - An Overview
- JADE User Manual - Configuration 2 - The Fragments Branch
    - JADE User Manual - Configuring Alternative Fragments
- JADE User Manual - Configuration 3 - The Profile Branch

**Navigation**

# JADE User Manual - Configuration 1 - An Overview

## Introduction

*Profiles* form the starting points for specifying file transfers, with a *Profile* being specified each time a command is sent to JADE to start a file transfer operation.

Among the parameters specified within a *Profile* are the *ProtocolFragment(s)* to be used - i.e. one or more references specifying predefined connection configurations.

The subject of this page is the basic principles lying behind this two-stage configuration procedure.

## Configuration Elements in the JADE Schema

In the XSD Schema used to structure the JADE configuration parameters, the parent of all individual *Profile* elements, the *Profiles* element, lies at the top of one of the three main branches in the configuration, alongside the *Fragments* and *General* branches. This hierarchy is shown schematically below:

- *Profiles*

        - *Profile* (profile_id='p1')
            - *Operation*
                - etc.
                    - **FragmentRef* (attribute = 'f1')
        - *Profile* (profile_id='p2')
            - *Operation*
                - etc.
                    - **FragmentRef* (attribute = 'f2')
        - *Profile* (profile_id='p3')
            - etc.
    - *Fragments*
        - *ProtocolFragments*
            - *FTPFragment* (name= 'f1')
            - *FTPFragment* (name= 'f2')
            - *FTPSFragment* (name= 'f3')
            - etc.
    - *General*
        - *etc.*

The *Fragments* branch is a further main configuration branch in the schema, containing the *ProfileFragments* elements that specify in detail *how* t he transfer is to be carried out.

The *General* branch is used to specify overall aspects of JADE's operation that are not directly related it individual file transfers, such as logging. It is not directly relevant for the purpose of this article and is only mentioned here for completeness.

This two-stage configuration procedure - calling a *Profile* which contains a reference to a *Fragment* - allows a flexibility not possible with a single XML hierarchy and, in particular, allows profile fragment elements to be reused.

### *What is a Profile Element?*

A *Profile* can be seen as a specification of *what* is to be done and contains hierarchical information about:

- the *Operation* to be carried out - e.g. *Copy* or *Move*,
- the protocols to be used for the source and, if relevant, target, parts of the transfer and
- references that specify *ProtocolFragment* elements. These in turn define in detail *how* the file transfer connections are to be made.

Any number of profiles can be specified within a file transfer configuration.

### *What are ProtocolFragments Elements?*

*ProtocolFragment* elements can be seen as a specification of *how* the file transfer is to be carried out. Each file transfer *Profile* contains a reference calling at least one *ProtocolFragment*.

The *ProtocolFragments* elements contains hierarchical information about:

- the connection - e.g. the *Hostname* and *Port*
- the authentication method - e.g. *Account* and *Password*
- whether a proxy is to be used and the connection and authentication

*ProtocolFragments* are protocol-specific - that is, they apply for a specific protocol. The XSD schema defines which configuration elements can be used for the protocol in question.
For example, the Schema does not  allow SSH authentication to be specified for an *FTPFragment*.

*ProtocolFragments* can be thought of as a library of pre-configured connections that can be called up as required.

*ProtocolFragment* elements can be reused - i.e. a *ProtocolFragment* can be referenced from any number of profiles.

### *Configuration Procedure*

Whilst *Profiles* form the starting points when running a file transfer command, with the *ProtocolFragments* then being called from within a *Profile* E lement, the configuration procedure is usually carried out in the reverse order: The necessary *ProtocolFragments* are configured first, before the *P rofile* elements which call these fragments are specified.

### *Referencing Profiles and ProtocolFragments*

Both *Profiles* and *ProtocolFragments* can be seen as predefined file transfer specifications that are called up as required:

All *Profile* elements require a *profile_id* attribute which is used to call the profile from the JADE command.

All *ProfileFragments* require a *name* attribute which is used to reference the fragment from an element in the *Profile* element.

## Configuring File Transfers with the XML Editor

We recommend that you use the XML Editor to generate all the parts of your configuration file. The editor effectively functions like a wizard: due to the use of the JADE XSD schema in the editor, you will be effectively guided through the configuration process and end up with a validated configuration that you can implement as required.

## General Comments

The advantage of this approach - which may at first seen somewhat complex - is that:

- fragments can flexibly reused within the otherwise strict XML hierarchy and
- configurations can be validated against an XSD schema. Validation means that the possibility of configuration errors is greatly reduced.

A fragment can be used as both a source or as a target within the one configuration.

## Further Information

### *Related Sections of this User Manual:*

- JADE User Manual - Configuration Formats & Tools
- JADE User Manual - Configuring and Running File Transfers
    - JADE User Manual - Configuration 1 - An Overview (this page)
    - JADE User Manual - Configuration 2 - The Fragments Branch
        - JADE User Manual - Configuring Alternative Fragments
    - JADE User Manual - Configuration 3 - The Profile Branch

### *Other documents*

- The *Profile* and *ProtocolFragments* child elements are described in detail in the *Profiles* and *ProtocolFragments* sections of the Parameter Reference.

# JADE User Manual - Configuration 2 - The Fragments Branch

## Specifying Connection Parameters with Protocol Fragments

When a source or target is specified for a file transfer operation a reference is made to a *ProtocolFragment* element. *ProtocolFragments* are elements in the *Fragments* branch of the JADE XSD Schema and can be thought of as a series of predefined part configurations that can be called up as required for specific file transfer operations.

Protocol fragments usually form the starting point when *configuring* a file transfer. *Profiles*, which reference protocol fragments, can then be specified after the necessary protocol fragments have been defined.

Operationally, the starting point is a *Profile*, which then calls the protocol fragment(s).

### *How to generate Protocol Fragments*

In the JADE XML hierarchy *ProtocolFragments* are children of the *Fragments* element and in turn can have any number of child elements. Example protocol fragment elements would be the *FTPFragment* and the *SFTPFragment*.

Protocol fragment elements have descendants that specify connection parameters such as the authentication method, the connection type and proxy.

The diagram below shows the fragment hierarchy required to specify a connection to carry out file transfer from a remote source per FTP using password authentication.

You are recommended to use the XML Editor to generate the configuration. The editor uses the JADE parameter configuration XSD schema to guide parameter specification.

Starting at the *ProtocolFragments* element, the *Add child* button is used to insert valid elements at each level in the hierarchy until the configuration is complete.

Note that the links behind each parameter name in the diagram lead to the Parameter Reference page for that element, which provides detailed information about the parameter.

- *Fragments*

- *ProtocolFragments*
    - *FTPFragment* (name='f1')
        - *BasicConnection*
            - *Hostname*
            - *Port*
        - *BasicAuthentication*
            - *Account*
            - *Password*

Protocol fragment elements are protocol-specific - that is, there is a *ProtocolFragments* element defined in the XSD schema for each file transfer protocol. This approach enables the properties of each protocol to be reflected in the schema and allows dependencies and incompatibilities to be defined. A trivial example here would be that a *PassiveMode* element can be specified for an *FTPFragment* but not for an *SFTPFragment*.

Note that protocol fragments can be used for either a transfer source or target as required. This is described in more detail in the next section.

#### *Calling Protocol Fragments*

Any number of *ProtocolFragments* can be specified within a file transfer configuration and any number of fragments can be defined for a particular protocol. A particular fragment is referenced by a *name* attribute and this attribute is referenced by a corresponding fragment reference element in the configuration *Profiles* branch.

Operation-dependent source and target elements specify the *ProtocolFragments* element that is to be used. For example:

- A *Copy* operation requires that *CopySource* and *CopyTarget* elements are specified.
- The *CopySource* and *CopyTarget* elements in turn are used to specify *CopySourceFragmentRef* and *CopyTargetFragmentRef* elements, that respectively define the fragments to be used for each of the two parts of the operation.

See the Configuring Profiles and Fragments - 3 The Profile Branch article for more information about configuring the elements around a *ProtocolF ragment* reference.

#### *Optional fragments*

The following optional fragments can be specified in addition to *ProtocolFragments*:

- *AlternativeFragments* elements are used to specify a number of fragments. These fragments will be applied in order, should, for example, a server not be available. For example, it is conceivable that in some situations a less secure protocol would be tried if the original (more secure) one is not available.
  *AlternativeFragments* are described in more detail here.
- *NotificationFragments* specify how e-mails informing about successful or unsuccessful file transfer operations are to be sent.
- *CredentialStoreFragments* are used to retrieve file transfer authentication credentials from a secure source.

### Further Information

#### *Related Sections of this User Manual:*

- JADE User Manual - Configuration Formats & Tools
- JADE User Manual - Configuring and Running File Transfers
    - JADE User Manual - Configuration 1 - An Overview
    - JADE User Manual - Configuration 2 - The Fragments Branch (this page)
        - JADE User Manual - Configuring Alternative Fragments
    - JADE User Manual - Configuration 3 - The Profile Branch

#### *Other documents*

The use of the *Operations* element - the only *Profiles* child element whose use is required - is described in the Operation section of this manual.

## JADE User Manual - Configuring Alternative Fragments

#### *Summary*

*AlternativeFragments* elements are used in situations where:

- a specified - i.e. first choice - connection may not be available,

- where it is not acceptable to wait for the first choice connection
- and where alternative transfer configurations could be used.

### *Specifying Alternative Fragments*

Alternative fragment elements are specified using either a *ReadableAlternativeFragmentRef* element or a *WriteableAlternativeFragmentRef* element, both of which are, respectively, children of the *CopySourceFragmentRef* or *CopyTargetFragmentRef* elements in the configuration *Profiles* branch.

Readable... elements are used where only read permissions are required - i.e. for transfer sources, Writable... elements are used where a file is to be written - i.e. for transfer targets and for transfer sources where the file to be transfered is modified before being transferred. Note that this can be where pre-processing is involved.

In the XML hierarchy this looks like:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
                - *CopySourceFragmentRef* (one of the following)
                    - *FTPFragmentRef*
                    - *FTPSFragmentRef*
                    - etc.
                    - *ReadableAlternativeFragmentRef* (Has the *ref* Attribute specifying the source *ReadableAlternativeFragment* fragments)
            - *CopyTarget*
                - *CopyTargetFragmentRef* (one of the following)
                    - *FTPFragmentRef*
                    - *FTPSFragmentRef*
                    - etc.
                    - *WriteableAlternativeFragmentRef*

The *ReadableAlternativeFragmentRef* and *WriteableAlternativeFragmentRef* elements contain attributes that reference corresponding *ReadableAlternativefragment* and *WriteableAlternativeFragment* elements in the *Fragments* configuration branch.

The *ReadableAlternativeFragment* and *WriteableAlternativeFragment* elements then contain two or more alternative fragment elements that will be tried in turn if the first one is not able to make a connection.

In the XML hierarchy this looks like:

- *Fragments*
    - *ProtocolFragments*
    - *AlternativeFragments*
        - *ReadableAlternativefragment*
            - *AltFTPFragmentRef (name = ....)*
            - *AltFTPFragmentRef (name = ....)*
            - etc.
        - *WriteableAlternativeFragment*
            - *AltFTPFragmentRef (name = ....)*
            - *AltFTPFragmentRef (name = ....)*
            - etc.

Note that the alternative fragment elements are themselves only references to named protocol fragment elements.

The list above shows *AltFTPFragmentRef* elements - it is equally possible to specify elements with other protocols. A list of the *Alt...FragmentRef* elements available can be found in the parameter reference pages for *ReadableAlternativeFragment* and *WriteableAlternativeFragment* elements.

### *Further Information*

#### Related Sections of this User Manual:

- JADE User Manual - Configuration Formats & Tools
- JADE User Manual - Configuring and Running File Transfers
    - JADE User Manual - Configuration 1 - An Overview
    - JADE User Manual - Configuration 2 - The Fragments Branch
        - JADE User Manual - Configuring Alternative Fragments (this article)
    - JADE User Manual - Configuration 3 - The Profile Branch

# JADE User Manual - Configuration 3 - The Profile Branch

- Overview
- How to generate the Profile configuration

## Overview

File transfer parameters in JADE are defined hierarchically and are sorted into two branches - *Profiles* and *Fragments*. A *Profile* specifies *what* is to be done (e.g. copy from A to B) and the *Fragment* defines *how* this is to be done (e.g. using protocol X, authentication Y, etc.).

*Fragments* (see the link to the relevant section of this manual at the foot of this article) are generally configured before *Profiles*. The considerations necessary for defining the *Profile* branch of the configuration are described in this article.

## How to generate the Profile configuration

The diagram below shows the minimum parameter hierarchy (i.e. without optional parameters) required to specify a file transfer from a remote source per FTP protocol to the local file system.

When the XML Editor is used to generate the configuration you will be guided using the *Add child* button to insert valid elements starting with the *Profiles* element and proceed down the hierarchy.

Note that the links behind each element name in the diagram lead to the Parameter Reference Documentation for that element, which provides detailed information about the parameter.

- *Profiles*
    - *Profile* (profile_id= 'p1')
        - *Operation*
            - *Copy*
                - *CopySource*
                    - *CopySourceFragmentRef*
                        - *FTPFragmentRef* (Has the *ref* attribute which specifies the source *ProtocolFragment*)
                    - *SourceFileOptions*
                        - *Selection*
                - *CopyTarget*
                    - *CopyTargetFragmentRef*
                        - *LocalTarget* (A *ProtocolFragment* does not need to be defined here as the target is the local file system)
                    - *Directory*
    - Profile (profile_id= 'p2')
    - etc.

Each individual file transfer is headed up by a *Profile*, which is added as a child of the *Profiles* element.

The individual parameters that have to be specified in the *Profile* are as follows:

- The *Profile* requires a *profile_id* which is used to call the profile when the transfer is called from the command line.
- Each *Profile* element is required to have an *Operation* element.
  Note that the *Operation* specified defines subsequent descendant elements. For example, a *Rename* operation will require a different descendants to a *Move* operation. See the JADE User Manual - The Operation article for more information.
- In the example, a *Copy* operation is specified, which, in turn, requires that *CopySource* and *CopyTarget* elements are set.
- The *CopySource* element has two child elements:
    - a *CopySourceFragmentRef* element, which is required. This element in turn requires an *FTPFragmentRef* element, which contains the reference specifying the protocol fragment element. The protocol fragment then specifies the transfer connection parameters such as authentication, protocol, etc.
        - Note that the *FragmentRef* element (*FTPFragmentRef*), which is used to specify the fragment element, is protocol-specific. This allows optional pre- and post-processing child elements - which are not shown in the hierarchy - to be added to the profile as child elements of the *FTPFragmentRef* element. Such elements are protocol-specific.
    - a *SourceFileOptions* element which is also required - together with the *Selection* element - and is used to specify a range of parameters related to aspects of file selection.
- The *CopyTarget* element does not need a corresponding *FTPFragmentRef* child element as the target is the local file system. However, the *LocalTarget* element must be specified.
    - The *Directory* element is used to specify a target directory if a directory other than the user's root directory is used.

## Calling Profile Elements

A *Profile* is specified by its *profile_id* attribute, which is used to call the profile when the transfer is called from the command line.

An example showing the JADE command line call syntax can be seen in the Configuring and Running File Transfers article.

## Further Information

### ***Related Sections of this User Manual:***

- JADE User Manual - Configuration Formats & Tools
- JADE User Manual - Configuring and Running File Transfers
    - JADE User Manual - Configuration 1 - An Overview
    - JADE User Manual - Configuration 2 - The Fragments Branch
    - JADE User Manual - Configuration 3 - The Profile Branch (this page)
        - JADE User Manual - The Operation - What is to be done?

### *Other documents*

- The use of the *Operations* element - the only *Profiles* child element whose use is required - is described in the Operation section of this manual.

# JADE User Manual - Example Configurations

## Example configuration issues

The change from the `*.ini` file based configuration to the XSD schema based XML hierarchy is described in general in the Configuration Formats & Tools article in this user manual.

A simple configuration - for the transfer from a remote server using the FTP protocol to the local file system - was presented in the form of XML script and as a screen shot of the XML Editor in the Configuring and Running File Transfers article in this manual as a concrete example.

The considerations behind the specification of different elements in simple file transfer configurations are discussed in the articles linked below. The articles do not go into detail about the individual parameter properties - that can be done using the relevant sections of the JADE Parameter Reference, which are linked from the articles below. Instead, the examples concentrate on the general procedure we recommend is followed when configuring a file transfer as well as the considerations to be borne in mind when setting up the XML configuration hierarchy.

**Pages**

- JADE User Manual - Simple File Transfer with Basic Authentication
- JADE User Manual - File Transfer with SSH authentication

**Navigation**

# JADE User Manual - Simple File Transfer with Basic Authentication

## Introduction

The configuration of a simple file transfer with basic - i.e. password - authentication is described on this page.

A simple FTP file transfer example is used in which files are transferred from a remote server to the local file system.

The configuration of this example is described in detail in our JADE - XML Configuration - Sample Files page.

On this page the more general aspects of the configuration procedure will be described.

### *Example Configuration File*

- On our JADE - XML Configuration - Sample Files page there is complete configuration download file available to help users get the example up and running as quickly as possible. This configuration is a working example that will download files from the online SOS Demo server to a local file system.
- The transfer configuration is described here uses the XML-based configuration introduced with Release 1.11 of JADE. A similar example using the *Settings* parameters configuration that was implemented for releases before 1.11 can be found our JADE Tutorials along with a configuration download file.

## *Configuration Tool*

We recommend using the XML Editor to generate the XML configuration file. This editor uses the JADE XSD schema as a guide through each stage of the process. The schema is used to control and then validate the elements that can be specified in the configuration.

Using the XML Editor will significantly reduce the possibility of configuration errors.

Alternatively, a text editor can be used to write the configuration file manually.

## Configuration Steps

The configuration described in this example is held as simple as possible. More complex file transfer scenarios will be described later.

The configuration procedure involves the following steps:

1. Configuration of the *Fragments* - i.e. the protocol, connection and authentication methods to be used for the source and target parts of the operation.
2. Configuration of the *Profile* - i.e.
    - the operation to be carried out (e.g. copy or move),
    - the fragments specified in Step 1 that are to be used for the source and target parts of the transfer and
    - any specific directories that are to be used.
    - Note that the Profile can also be used to specify other elements such as any notifications to be made or preprocessing that is to be carried out. These will be described in other parts of this manual.
3. Calling the configuration and profile from the command line, a batch file or via the JADE API.

## *Step 1: Fragment Configuration Considerations*

### The Remote Source Fragment

The first step in the configuration is to configure the *ProtocolFragments* to be used for the transfer from the source. Other fragments are not required for this example.

Fragments are XML elements that specify *how* transfers are to be carried out. Protocol fragments can be thought of as predefined connection specifications. They are protocol-specific - the FTP protocol, for example, brings a different set of features and constraints than WebDAV protocol.

Fragments are not operation-specific - a fragment could be used as the target for one operation and then as a source for a subsequent one or in a *Copy* operation and then in a *Remove*.

Any number of fragments can be predefined in a configuration.

Fragments are given *name* attributes, with which they will be referenced from the file transfer profile as shown in the next step.

The fragment that is to be used for a particular part of the transfer operation is then specified in a profile element.

For the current example a fragment only has to be configured for the source part of the transfer as the transfer target is the local file system.

The XML hierarchy for the *FTPFragment* for this example is structured as follows:

- *ProtocolFragments* (parent element)
    - *FTPFragment* (with name attribute)
        - *BasicConnection*
            - *Hostname*
            - *Port number* (optional)
        - *BasicAuthentication*
            - *Account*
            - *Password* (optional)

Note that the XML hierarchy shown above has been simplified through the omission of optional elements that are not directly relevant.

### *FTP Transfer Properties*

The XML hierarchy shown above reflects key properties of FTP file transfer:

- Only a *BasicConnection* element with a *Hostname* and optional *Port* is possible - URL connections are not possible.
- Only *BasicAuthentication* with an *Account* (i.e. user name) and optional *Password* is possible: SSH authentication, for example, requires that the SFTP protocol is used.

#### The Remote Target Fragment

It is not necessary to define a fragment for the target part of the operation as this takes place locally and the target directory is specified as part of the transfer profile.

### *Step 2: Profile Configuration Considerations*

The second configuration step is to define the *Profile*, which is operation-specific. For example, a *Profile* for a *Move Operation* requires source and target to be specified where as a *Remove* operation only requires a source.

The operation in the current example is *Copy* for which a source and a target elements (*CopySource* and *CopyTarget* respectively) have to be specified.

The type of fragment to be used for the source part of the transfer - here an *FTPFragmentRef* element - is specified in the *CopySourceFragmentR ef* element.

The FTP fragment itself is then specified in the *FTPFragmentRef* element's *ref* attribute.

In this example, the *CopyTargetFragmentRef* is used to specify that the target is to be local. The target directory is specified in the *Directory* elem ent, which is a sibling of the *CopyTargetFragmentRef* reference element.

The XML for the copy operation from remote to local is structured as follows:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
                - *CopySourceFragmentRef*
                    - *FTPFragment*Ref (with *ref* attribute for *FTPFragment*)
            - *CopyTarget*
                - *CopyTargetFragmentRef*
                    - *LocalTarget*
                - *Directory*

Note that once again the XML hierarchy shown above is simplified through the omission of the majority of optional elements.

### *Step 3: Calling the Configuration Profile*

The configuration profile is specified with two parameters when JADE is called from the command line:

- the name of the XML configuration file and
- the *Profile* to be used.

Example calls for Windows and Unix systems can be found on our Configuring File Transfers with the JADE Client page.

## Further Information

- Configuring File Transfers with the JADE Client
- An overview of the authentication methods that can used with JADE can be found in the Authentication article.
- Parameter Reference

# JADE User Manual - File Transfer with SSH authentication

## Introduction

This article builds on the Simple File Transfer with Basic Authentication article, concentrating on the configuration aspects introduced with use of SSH authentication.

### *Note:*

- On our JADE - XML Configuration - Sample Files page there is a complete configuration file available for download that uses SSH public/private key authentication. This configuration is a working example that comes with the necessary private key authentication file and will download files from the online SOS Demo server to a local file system.
- The transfer configuration is described here uses the XML-based configuration introduced with Release 1.11 of JADE. A similar example using the *Settings* parameters configuration that was implemented for releases before 1.11 can be found our JADE Tutorials along with a configuration download file.

## Features of SSH Authenticated File Transfer

### *Advantages*

SSH authentication is implemented with the SFTP protocol and together provides a significantly higher level of security than basic - i.e. password - authentication using the FTP protocol.

### *Authentication Features*

The main features of SSH authenticated file transfer in JADE are:

- SSH authentication can be carried out with an account name and either password or public/private key verification.
- As mentioned above, SSH authentication requires that SFTP protocol is used.
- SSH authentication is required when a jump host transfer is carried out.

As with all file transfer protocols, any number of SFTP file transfer configurations in the form of *ProtocolFragments* can be preconfigured and selected as required.

If public/private key verification is used, the SSH password parameter can be used to provide additional protection for the authentication file.

### *File Transfer Protocol*

SSH Authentication requires that the SFTP protocol is used for the transfer operation.

## Configuration of SSH Authenticated File Transfer

The configuration of a file transfer with SSH authentication follows the steps already described described in the Simple File Transfer with Basic Authentication article.

There are, however, two differences to the Simple File Transfer with Basic Authentication example:

- the use of an *SSHAuthentication* element instead of *a BasicAuthentication* element
- SFTP protocol is used instead of FTP as FTP cannot support SSH authentication

Both examples are otherwise kept as simple as possible for clarity. More complex file transfer scenarios will be described later.

### *Specification of SFTP File Transfer Elements*

The use of SSH authentication requires that the SFTP protocol (or a jump host) is specified. This done by specifying a *Profile* element that calls a suitably configured *SFTPFragment*. This *Profile* will then be called when JADE is started. The *SFTPFragment* will contain at least the specification of a *BasicConnection* element and the *SSHAuthentication* element.

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
                - *CopySourceFragmentRef*
                    - `Ref -> SFTPFragment (Ref=Name)`
- *Fragments*
    - *ProtocolFragments*
        - *SFTPFragment (*identified by *name* Attribute*)*
            - *BasicConnection*
            - *SSHAuthentication*

More information about the specification of *SFTPFragments* can be found in:

- the *SFTPFragment* Parameter Reference article.

### *Specification of SSH Authentication*

Use of the *SSHAuthentication* element requires specification of:

- an *Account* (i.e. a user name, required)

- either an *AuthenticationMethodPassword* element, which in turn requires a *Password* to be specified
  or an *AuthenticationMethodPublickey* element which requires that an *AuthenticationFile* and optional *Passphrase* are specified

More information about the specification of *SSHAuthentication* can be found in:

- the SSHAuthentication Parameter Reference article
- the Authentication User Manual article.

### Further Information

- Configuring and Running File Transfers
- The XML Editor
- Parameter Reference

# JADE User Manual - Connection Protocols

## Connection Protocols

The JADE Client can use a number of protocols to make connections for the source and target parts of the transfer. These protocols are specified in parameters set in protocol fragment elements in the JADE XML configuration.

- FTP and FTPS
- HTTP and HTTPS
- Jump,  SFTP and SMB
- WebDAV

Each protocol fragment has a number of *required* and *optional* child elements, depending on the properties of the protocol itself.

For example, the FTPS protocol element has the following *required* child elements:

- *BasicConnection*
- *BasicAuthentication*

And the following *optional* child elements:

- *AcceptUntrustedCertificate*
- *FTPSClientSecurity*
- *FTPSProtocol*
- *JumpFragment*
- *ProxyForFTPS*

## Connection types

As noted for the FTPS protocol element example above, each protocol element requires a connection type. The connection types available are:

- *BasicConnection*, with a required *Hostname* and an optional *Password*
  The *BasicConnection* is used for the *following ProtocolFragments*:
    - *FTP*, *FTPS, Jump,*
    - *SOCKS4* and *SOCKS5* proxies
- *URLConnection* with a required *URL*
  The *URLConnection* is used with the following *ProtocolFragments*:
    - *HTTP, HTTPS*
    - *WebDAV*

## Specifying Connection Protocols

Connection protocols are specified in the *Fragments* elements of the XSD Schema. All fragments elements are child elements of the *ProtocolFrag ments* element as shown schematically below:

- *Fragments*
    - *ProtocolFragments*
        - *FTPFragment*

        - *BasicConnection*
        - *BasicAuthentication*
            - *Account*
            - *Password* (optional)
        - Optional fragment elements
    - *FTPSFragment*
        - *BasicConnection*
        - *BasicAuthentication*
            - *Account*
            - *Password* (optional)
        - Optional fragment elements
    - *HTTPFragment*
        - Etc ...
    - Etc ...

As explained in the Configuration 2 - The Fragments Branch article, fragments elements are called from the *Profiles* branch of the configuration and are identified by a *name* attribute specified in the **Fragment* element.

Note that any number of *ProtocolFragments* can be specified within a configuration, allowing a number of authentication methods to be predefined for a particular file transfer operation. The *AlternativeFragments* element can be used to specify a number of alternative connections and authentication methods for a file transfer profile.

### Example Configuration

An example configuration for an FTP connection showing both the *Fragments* and *Profiles* branches can be seen in the JADE User Manual - Simple File Transfer with Basic Authentication article.

# JADE User Manual - Authentication

## Authentication methods

The JADE Client provides a number of authentication methods:

- *BasicAuthentication*
    - an *Account* (i.e. user name) is required
    - AND an optional *Password*
    - *BasicAuthentication* can be used with the following protocol fragments:
        - *FTPFragment* and *FTPSFragment*
        - *HTTPFragment* and *HTTPSFragment*
        - *WebDAVFragment*
    - Note that *BasicAuthentication* requires that the *Password* is stored in the configuration files.
    - The Simple File Transfer with Basic Authentication article describes the considerations behind an example with *BasicAuthentication*. The configuration file for this example is available as a working download.
- *SSHAuthentication*
    - an *Account* (i.e. user name) is required
    - AND either:
        - *AuthenticationMethodPassword* with a (required) *Password*
        - OR *AuthenticationMethodPublickey*
            - with a (required) *AuthenticationFile* location
            - AND an optional *Passphrase* that protects the file
    - *SSHAuthentication* can be used with with the following protocol fragments:
        - *SFTPFragment*
        - *JumpFragment*
    - The File Transfer with SSH authentication article describes the considerations behind an example with *SSHAuthentication*. The configuration file for this example is available as a working download.
- *SMBAuthentication* (in conjunction with *SMBFragment* elements / the SMB protocol and a *Hostname*)
    - an *Account* (i.e. user name) is required
        - AND an optional *Domain*
        - AND an optional *Password*
    - *SMBAuthentication* can only be used with with the SMB protocol fragment.

## Specifying Authentication Methods

Authentication methods are specified in the *Fragments* elements of the XSD Schema at the end of various *ProtocolFragments* branches in the schema. This is because the authentication method(s) that can be used as for a connection depend on the protocol used. This hierarchy is shown schematically below:

- *Fragments*
    - *ProtocolFragments*
        - *FTPFragment*
            - *BasicConnection*
            - *BasicAuthentication*
                - *Account*
                - *Password* (optional)
        - *FTPSFragment*
            - *BasicConnection*
            - *BasicAuthentication*
                - *Account*
                - *Password* (optional)
        - *HTTPFragment*
            - etc ...
        - etc ...

The protocol-dependency of the various authentication methods is integrated into the schema and explains why authentication methods lie below *ProtocolFragments* in the XML hierarchy. This integration in the schema ensures that the only authentication methods that are supported by a protocol can be specified.

Note that any number of *ProtocolFragments* can be specified within a configuration, allowing a number of authentication methods to be predefined for a particular file transfer operation. The *AlternativeFragments* element can be used to specify a number of alternative connections and authentication methods for a file transfer profile.

## Optional Authentication-Related Elements

Note that the following authentication elements can be optionally specified:

### Proxies

Proxies can be specified for each connection in the relevant connection fragment. For example, the *ProxyForFTP* element is specified as a child of the *FTPFragment*.

### Alternative Connections

Alternative protocols and thereby alternative authentication methods can be specified using the *AlternativeFragments* element.

This element is used to define a series of connections that are tried one after the other in the event of a primary connection not being available.

### Strict Host Key Checking

- *StrictHostkeyChecking* can be specified for *SFTPFragments* and *JumpFragments* to provide maximum security against Trojan horse attacks.

### Untrusted Certificates

- *AcceptUntrustedCertificate* can be used to allow self-signed certificates to be accepted with *FTPSFragments*, *HTTPSFregments* and *WebDAVFragments*.

### Authentication using the SOS Credential Store

JADE can use the *Credential Store* to securely access authentication credentials.

# JADE User Manual - File Selection, Handling and Renaming and the Target Directory

**Notes for the editor**
- no xsd-images are used in this page as only one xsd-image per page is allowed

## Introduction

The selection & handling of the objects being transferred can be divided into a number of areas:

- **File selection** - which broadly covers the specification of features that are related to the *source* side of the file transfer operation such as filtering, polling and directives.
- **File handling** - which broadly covers features that are implemented on the *target* side of the operation such as atomic transfer and the compression, checking and appending of files.
- **Renaming** - which is also related to the source side of the file transfer operation and - in contrast with the file selection and file handling features - is protocol-specific.
- **Transfer target specification** - specification of the transfer target directory and the conditions under which files are to be written to it.

JADE can also be configured to check whether file transfer has been successfully completed. See:

- the User Manual *FileTransferOptions* article which contains a description of the *Transactional* parameter used to ensure that transfer of a group of files has been completed.
- the *Atomicity* Parameter Reference article for a description of the *AtomicPrefix* and *AtomicSuffix* parameters which are used to ensure that transfer of individual files has been completed..

## File Selection

### How to specify file selection

File selection is specified in the *Profiles* branch of the configuration and can be applied to for all the *Operations* that can be defined for JADE: *Cop y*, *Move* etc.

File selection is a source side operation, and the JADE file transfer options are all specified as child elements of the *SourceFileOptions* element.

This configuration hierarchy is shown for the *Copy* operation as follows:

- *Profile*
    - *Operation*

            - *Copy*
                - *CopySource*
                    - *CopySourceFragmentRef*
                    - *SourceFileOptions*
                        - *Selection* (required)
                        - optional file selection parameters
                - *CopyTarget*
                    - copy target parameters
            - *Move*
            - etc.

Note that *all* the *optional* file selection parameters apply to *all* files matching the selection criteria.

## File Selection Functions

JADE comes with a comprehensive set of file selection functions:

- ***Select an individual file or folder:***
    - *FilePathSelection*
        - This parameter is used to specify an individual file or a folder from which *all* files will be processed.

- ***Select a dynamic number of files from a folder:***
    - *FileSpecSelection*
        - Selects *all* files from a folder that match a regular expression
        - Can be used to select files *recursively*

- ***Select files from a list:***
    - *FileListSelection*
        - This parameter is used to specify a number of files for processing.

These functions are specified using one of the child elements of the *Selection* element.

The element hierarchy for the *Selection* element when a *Copy* operation is specified is:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
                - *CopySourceFragmentRef*
                    - *SourceFileOptions* (required)
                        - *Selection* (required)
                            - *FilePathSelection*
                                - *FilePath* (required)
                                - *Directory*
                            - *FileSpecSelection*
                                - *FileSpec* (required)
                                - *Directory*
                                - *Recursive*
                            - *FileListSelection*
                                - *FileList* (required)
                                - *Directory*

Exact definitions for each file selection element and the interaction between elements can be found in the *Selection* Parameter Reference.

## Checking files for availability before transfer (CheckSteadyState)

The *CheckSteadyState* option can be used as a method for checking that a file is fully available at the file transfer source, for example, when waiting for the receipt of large files before processing.

The file *size* is measured at regular intervals and until a steady state is determined.

Note that this method of determining whether a file is fully available is not 100% reliable. This is discussed in detail in the *CheckSteadyState* para meter reference along with a description of the three child elements of the *CheckSteadyState* parameter: *CheckSteadyStateInterval*, *CheckStead yStateCount* and *CheckSteadyStateErrorState*.

The *CheckSteadyState* parameter is a child element of *SourceFileOptions* as shown:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*

                        - *CopySourceFragmentRef*
                            - *SourceFileOptions* (required)
                                - *Selection* (required)
                                - *CheckSteadyState* (option)
                                    - *CheckSteadyStateInterval*
                                    - *CheckSteadyStateCount*
                                    - *CheckSteadyStateErrorState*

### Behavior when transfer criteria are not met (Directives)

Directives can be seen as optional parameters that specify how JADE is to generally react at the file transfer source.

The *Directives* element itself is a child of the *SourceFileOptions* elements:

- *SourceFileOptions*
    - *Selection* (required)
    - *Directives* (optional)
        - *DisableErrorOnNoFilesFound*
        - *TransferZeroByteFiles*
    - Other optional source file elements

The directives are:

- ***DisableErrorOnNoFilesFound***
    - The *DisableErrorOnNoFilesFound* parameter allows the default behavior of JADE to be changed.
      Note that the use of this parameter can modify the behavior of JADE when *TransferZeroByteFiles* is set.

- ***TransferZeroByteFiles***
    - The *TransferZeroByteFiles* parameter specifies how JADE is to handle zero byte files - i.e. whether they are to be transferred and processed by subsequent commands.
      See the Parameter Reference for a details about how this parameter interacts with the *Remove* and *DisableErrorOnNoFilesFoun d* parameters.

### Polling for files

JADE can use *Polling* to wait for incoming files. The files to be polled are specified with the *Selection* element and polling can be is used to specify, for example, how often and how long should be polled.

*Polling* only continues until the polling criteria have been met (i.e. one or a minimum number of files have been found).

File transfer is started once polling has stopped.

When JADE is run as a job in the JobScheduler a job node can be specified for execution in case polling is unsuccessful.

A detailed list of the polling parameters that can be set and their definitions can be found in the *Polling* section of the Parameter Reference.

The *Polling* parameter is a child element of *SourceFileOptions* as shown:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
                - *CopySourceFragmentRef*
                    - *SourceFileOptions* (required)
                        - *Selection* (required)
                        - *CheckSteadyState* (option)
                        - *Polling* (option)
                            - *PollInterval*
                            - *PollTimeout*
                            - *etc*
                        - Other optional source file elements

### Restricting the files to be transferred (multiple parameters)

The following file filtering options can be used to filter the files that are to be transferred:

- ***FileAge***
    - *FileAge* sets the minimum and/or maximum file age

- ***FileSize***

        - *FileSize* sets the minimum and/or maximum file size

- ***ResultSet***
    - The *ResultsSet* parameter handles the result set of a *Selection* of files from the source host. The result set can be written into a file or, when JADE is run as a JobScheduler job, checked against an expectation.

- ***SkipFiles***
    - *SkipFile*s allows the first and/or last files found to be omitted from transfer.

- ***MaxFiles***
    - The *MaxFiles* parameter allows the maximum number of files to be transferred to be set. All other files will be ignored.

A detailed description of the file filtering elements listed above can be found in the *SourceFileOptions* Parameter Reference.

This configuration hierarchy relevant for the *SourceFileOptions* parameters where a *Copy* operation is specified is:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
                - *CopySourceFragmentRef*
                    - *SourceFileOptions*
                        - *Selection* (required)
                        - *FileAge*
                        - *FileSize*
                        - *ResultSet*
                        - *SkipFiles*
                        - *MaxFiles*
                        - Other optional file selection elements

## File Handling

File handling is used here to refer to aspects of the transfer that are carried out target side of a transfer operation.

### How to specify file handling

File handling is specified in the *Profiles* branch of the configuration and can be applied to those *Operation* elements that can be defined for JADE: *Copy*, *Move* etc.

File handling is a target side operation, and the JADE file handling options are all specified as child elements of the *TargetFileOptions* element.

This configuration hierarchy is shown for the *Copy* operation as follows:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
                - *CopySourceFragmentRef*
                - *SourceFileOptions*
                    - *Selection* (required)
                    - optional file selection parameters
            - *CopyTarget*
                - *CopyTargetFragmentRef* (required)
                - *Directory* (required)
                - *TargetFileOptions* (option)
                    - *AppendFiles* (option)
                    - *Atomicity* (option)
                    - etc.
        - *Move*
        - etc.

Note that *all* the optional file selection parameters apply to *all* files matching the selection criteria.

### The file handling options

#### *Appending one file to another (AppendFiles)*

The *AppendFiles* element can be used to specify whether files are to be appended to other files (i.e. added to the end of a file already existing in the target directory)

*AppendFiles* is specified as a child element of the *TargetFileOptions* element, which itself can be a child element of the *CopyTarget* or *MoveTarge t* elements as shown schematically in the How to specify file handling section above:

Files are appended to files with the same name.

Note that if the *AppendFiles* element parameter is set to *true* then *DisableOverwriteFiles* (another optional child of *TargetFileOptions*) will be ignored.

### Functioning

If we have two files to be transferred: *file_a* and file_b and a *file_a* already exists in the target directory, then the contents of the *file_a* being transferred will be appended to the already existing *file_a*. *file_b* will be written to the target directory as usual.

### Atomicity

The *Atomicity* parameter is used to mask the presence of a file being transferred to a directory that may be monitored until the file has been fully transferred.

See the *parameter reference* for a detailed description of this parameter and its interaction with other elements.

### Ensure that file transfer is complete:

- ***CheckSize***

  When specified, the *CheckSize* parameter causes the size (i.e. the number of bytes) of the files in the source and target directories to be compared.

  This method is not as accurate as *CheckIntegrityHash*.

- ***CheckIntegrityHash***

  The *CheckIntegrityHash* element causes integrity hash sums to be calculated for the files at source and target after transfer. If the hashes are not equal then the transfer will be rolled back.
  This method is more accurate than the *CheckSize* method.

### Compress files (CompressFiles)

JADE can compress individual files before writing them to the target directory.

The *CompressFiles* element specifies whether files should be compressed with a zip algorithm before saving to the target system.

Note: JADE cannot combine files at the source to a single compressed file before transfer (see the relevant change Issue for details).

### Configure error handling (DisableErrorOnNoFilesFound)

The default configuration for JADE is that an error will be logged if a transfer should take place but no files are found.

This behavior can be altered with the *DisableErrorOnNoFilesFound* element, which is a child of the *Directives* element and which is itself a child of the *SourceFileOptions* elements.

See the *DisableErrorOnNoFilesFound* parameter reference article for more detailed information about this element, in particular on its interaction with the *FileSpec*, *FilePath* and *FileList* elements.

### CumulateFiles

The *CumulateFiles* element specifies whether the contents of the transferred file(s) should be appended to the contents of an already existing file. All files being transferred will be written to the one file.

### Compress files at the target (CompressFiles)

The *CompressFiles* element causes files to be compressed after transfer but before being written to the target file system.

Files are compressed individually, the possibility of adding files to a compressed archive either before or after transmission is not available.

### KeepModificationDate

The *KeepModificationDate* element specifies whether the modification date of the source file is to be applied to the target.

This option is protocol-dependent and requires that the necessary permissions are available.

See the *TargetFileOptions* Parameter Reference for more detailed information about this parameter and its functioning with different protocols.

#### *DisableMakeDirectories*

The *DisableMakeDirectories* element specifies whether JADE is to create a target directory if the directory specified in the *Directory* parameter is not available.

This option is protocol-dependent and requires that the necessary permissions are available.

See the *TargetFileOptions* Parameter Reference for more detailed information about this parameter and its functioning with the *Recursive* parame ter and different protocols.

#### *DisableOverwriteFiles*

The *DisableOverwriteFiles* element specifies whether files can be overwritten.

## File Renaming

File renaming is specified using the *Rename* element, and can be applied at either the source or target sides of a file transfer.

Renaming can be applied to operations that are carried out using all protocols JADE can currently use (Release 1.11.x).

File renaming is protocol-specific and is specified as a child element of *FragmentRef* elements such as *FTPFragmentRef*, *SMBFragmentRef* and *LocalSource* and *LocalTarget* elements in *Profiles*.

The element hierarchy related to a Rename parameter specified for a copy to a local target operation is:

- Copy
    - *CopySource*
        - child elements ...
    - *CopyTarget*
        - *CopyTargetFragmentRef*
            - *LocalTarget*
                - *Rename* (optional)
                    - *ReplaceWhat*
                    - *ReplaceWith*

The *Rename* element requires that two child elements are defined:

- *ReplaceWhat* - a regular expression that allows selection of parts of file names.
- *ReplaceWith* - complex replacement patterns using substitution masks and capturing groups can be used.

A detailed description of the possible file name replacement patterns can be found in the *Rename* element parameter reference.

See also *Atomicity*.

## The Target Directory

### The Directory Element

The *Directory* element defines the target folder for a file transfer and is required for the *Copy* and *Move* operations.

For a *Copy* operation, the *Directory* element is specified as a sibling element of the *CopyTargetFragmentRef* and *TargetFileOptions* elements as shown schematically below.

- Copy
    - *CopySource*
        - Child elements
    - *CopyTarget*
        - *CopyTargetFragmentRef* (required)
        - *Directory* (required)
        - *TargetFileOptions* (optional)

### Writing files

- #### ***DisableMakeDirectories***

- #### ***DisableOverwriteFiles***

These elements are children of the *TargetFileOptions* element and are described in the File Handling section above.

# JADE User Manual - The Operation - What is to be done?

## Introduction

The *Operation* element specifies the type of file transfer that is to be carried out. The *Operation* is a child element of the *Profile* and a *required* ele ment.

## *Operation* - the action JADE is to carry out

The *Operation* elements that the JADE Client can carry out are defined as follows:

- Copy - duplicate the files from one location to another
- *Move* - duplicate the files from one location to another and then delete the original files
- *Remove* - delete files from a server
- *GetList* - obtain a file containing a list of file names

## Specifying the Operation Element

An *Operation* element is required for each file transfer configuration.

In the JADE XSD schema, the *Operation* is a child of the *Profile* element as is shown in the hierarchy below:

- *Profile*
    - *Operation*
        - *Copy*
            - *CopySource*
            - *CopyTarget*
        - *Move*
            - *MoveSource*
            - *MoveTarget*
        - *Remove*
            - RemoveSource
        - *GetList*
            - *GetListSource*
    - Client (optional)
    - other optional siblings ...

The hierarchy also shows how the *Operation* element that is selected defines whether source and target elements require to be called. These in turn define the transfer options that may be available.

The *Operation* element has a number of siblings (*Client*, *JobScheduler*, *Notifications* and *CredentialStore*) which are optional children of the *Profil e* element and are described in the *Profile* parameter reference article.

When the SOS XML Editor us used, selection of an *Operation* will cause the Editor to automatically generate the *required* descendants necessary to specify the *Operation*.

For the *Copy Operation* these would be:

- *Copy*
    - *CopySource*
        - *CopySourceFragmentRef*
            - **FragmentRef (contains the reference to the Fragment to be used, where * defines the protocol to be used)*
        - *SourceFileOptions*
            - *Selection*
    - *CopyTarget*
        - *CopyTargetFragmentRef*
        - *Directory*

## Example Configuration using the *Copy* Operation

An example configuration using the *Copy* Operation was shown on the Configuring File Transfers with the JADE Client page.

## Example Configuration using the *GetList* Operation

An example configuration using the *GetList* Operation is shown on the Example JADE Configuration using the GetList Operation page.

## Further Information

- We recommend using the Editor for XML Configuration to configure file transfers for JADE. The editor uses the JADE XSD schema to guide users through the configuration process and validate configurations.
- The *Operation* parameter reference page.

# JADE User Manual - File Transfer Options

## File Transfer Options

File *TransferOptions* are all optional and are used to:

- optimize file transfer (*BufferSize*, *ConcurrentTransfer*) and
- ensure transfer has been completed and handle errors (*TransactionalTransfer*)

File transfer options are specified as child elements of the file transfer operations - e.g. *Copy* and *Move* - and apply to both the source and target parts of the transfer operation.

### Specification of File Transfer Options

File Transfer Options are defined using the *FileTransferOptions* elements. *FileTransferOptions* elements are specified as part of a file transfer *Pro file*, as child elements of the *Copy* or *Move* elements as shown in the following hierarchy:

- *Profile*
  - *Operation*
    - *Copy*
      - *CopySource*
        - Child elements ...
      - *CopyTarget*
        - Child elements ...
      - *TransferOptions*
    - *Move*
      - *MoveSource*
        - Child elements ...
      - *MoveTarget*
        - Child elements ...
      - *TransferOptions*

The JADE file transfer options are independent of the transfer protocol. As they apply to the *transfer,* it follows that they are independent of the source /target and are specified as sibling elements of - in the cases of a *Move* operation - the *MoveSource* and *MoveTarget* elements.

The *TransferOptions* element can have three children as listed below. The use of these child elements is described in the next sections:

- *TransferOptions*
  - *BufferSize*
  - *ConcurrentTransfer*
    - *MaxConcurrentTransfers*
  - *Transactional*

### Transfer Optimization 1 - Buffer Size

The *BufferSize* parameter element defines the maximum data block size.

### Transfer Optimization 2 - Concurrent Transfer

The *ConcurrentTransfer* element is used to allow parallel file transfer to occur.

A child element of *ConcurrentTransfer* - *MaxConcurrentTransfers* - can be used to limit the number of parallel transfers.

- *TransferOptions*
  - *ConcurrentTransfer*
    - *MaxConcurrentTransfers*

## Ensuring Transfer has been Completed

### Transactional Transfer

This parameter specifies whether a transfer should be processed as a single transaction, i.e. whether all objects are to be successfully transferred or none at all.

Should an error occur during a transfer operation then transfers already carried out will be rolled back.

Transactional transfer is often used together with the following parameters:

- *Atomicity* (a *TargetFileOption* for the *CopyTarget* and *MoveTarget* elements)
- *DisableOverwriteFiles* (a *TargetFileOption* for the *CopyTarget* and *MoveTarget* elements)

See the *Transactional* parameter reference article for a detailed explanation of the dependencies between these elements.

Note that transactional transfer is not 100% reliable - there are situations where rollback will not be possible.

# JADE User Manual - Pre- & Post-Processing

## Pre- & Post-Processing Options

JADE can carry out both pre- and post-processing at the source and at the target parts of the transfer.

A typical post-processing operation at the source would be to *move* or *rename* a file after it has been copied to the target.

Pre- and post-processing can be triggered for:

- Transfer operations.
- Individual file transfers, so when three files are to be transferred, processing would be carried out three times.

In addition, post-processing can be triggered for:

- File *Rename* operations.

Pre- and post-processing can be carried out using the FTP, SFTP and local file system protocols: they are not possible with HTTP, HTTPS, SMBF or WebDAV.

### Pre- and post-processing commands

The commands that can be executed as part of a pre- or post-processing operation are protocol-dependant:

- FTP commands have to be used with the FTP protocol and
- Shell commands have to be used with the SFTP protocol.

### Pre- and post-processing operations

The pre- and post-processing operations available are:

- Pre-processing operations:
    - *CommandBeforeFile*:
        - commands are executed before each individual file is transferred
    - *CommandBeforeOperation*
        - commands are executed before the transfer operation is started
- Post-processing operations:
    - *CommandAfterFile*:
        - commands are executed after each file transfer has been completed
    - *CommandAfterOperation*
        - commands are executed after the transfer operation has been completed
    - *CommandBeforeRename*
        - commands are executed before each individual file is renamed

Note that special variables such as *$target_dir* and *$date* are available for pre- or post-processing commands. These variables are listed under the Parameter Reference article for the relevant protocol **FragmentRef* element. See the list of links provided in the Parameter reference for pre- and post-processing operations section on this page for more information.

## Specifying pre- and post-processing operations

The commands that can be executed as part of a pre- or post-processing operation are protocol specific - meaning that they are specified as child elements of a protocol fragment element such as *SFTPFragmentRef.*

The XML hierarchy used to specify the pre- and post-processing operations around a typical *Copy* operation using SFTP would be:

- Profile
    - *Operation*
        - CopySource
            - *CopySourceFragmetRef*
                - *SFTPFragmentRef*
                    - *SFTPPreProcessing*
                        - *CommandBeforeFile*
                        - *CommandBeforeOperation*
                    - *SFTPPostProcessing*
                        - *CommandBeforeFile*
                        - *CommandBeforeOperation*
                        - *CommandBeforeRename*
                    - etc. (optional)

## Parameter reference for pre- and post-processing operations

Detailed information about pre- and post-processing operations can be found in the relevant parameter reference article:

- *FTPFragmentRef*
- *LocalSource*
- *LocalTarget*
- *SFTPFragmentRef*

# JADE User Manual - The Use of Proxies

## Introduction

JADE can establish connections that are routed over proxies.

However the type of authentication allowed and the protocol used for the 'main' connection have to be considered when selecting a proxy.

## File Transfer Protocols and Proxy Protocols

The following proxy protocols are available:

- *HTTPProxy*
- *SOCKS4Proxy*
- *SOCKS5Proxy*

Different Proxy protocols can be used for different file transfer protocols. For example:

- HTTP proxies can be used with the FTP, HTTP, HTTPS and WebDAV protocols.
- SOCKS4 proxies can be used with FTP, FTPS and SFTP protocols.
- SOCKS5 proxies can be used with FTP, FTPS and SFTP protocols.
- See the JADE Parameter Reference - Reusable Elements - Proxy for more detailed information.

## Local or remote Proxy Connection and Authentication

A proxy can be installed on any host in the network and is identified by its connection parameters.

Some Proxy protocols - e.g. SOCKS5 - allow authentication credentials to be specified.

Proxies connections are all configured using *BasicConnection* and *BasicAuthentication* elements - *SSHAuthentication* is not possible.

## Specifying Proxy Connections

Connections to source hosts, jump hosts or target systems can be routed by a proxy.

- Proxies are defined for file transfer protocols and connections:
    - In the JADE XML configuration schema proxy elements are children of *ProtocolFragments*. For example,
        - the *ProxyForSFTP* element is a child of the *SFTPFragment*

        - two separate *ProtocolFragments* would need to be configured if it was required to define a direct connection to a server and a connection over a proxy.
- A proxy configuration consists of:
    - a connection to the host where the proxy is located, which is specified with a *BasicConnection* element and
    - credentials for authentication (optional), specified using a *BasicAuthentication* element

The XML element hierarchy used to specify a proxy for - here - an SFTP connecton are:

- *SFTPFragment*
    - *BasicConnection*
    - *SSHAuthentication*
    - *ProxyForSFTP* (optional)
        - *SOCKS4Proxy*
            - *BasicConnection*
                - *Hostname*
                - *Port*
        - *SOCKS5Proxy*
            - *BasicConnection*
                - *Hostname*
                - *Port* (optional)
            - *BasicAuthentication*
                - *Account*
                - *Password* (optional)
    - other optional elements